Artificial Intelligence in Design '96, John Gero and Fay Sudweeks (eds.), Kluwer Academic Publishers, Boston, 1996, pp. 409-428

LEARNING BY SINGLE FUNCTION AGENTS DURING SPRING DESIGN

DAN L. GRECU, DAVID C. BROWN
AI in Design Group
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609, U.S.A.
E-mail: dgrecu@cs.wpi.edu, dcb@cs.wpi.edu
URL: http://cs.wpi.edu/~dgrecu, http://www.wpi.edu/~dcb

1.0 Introduction

This paper reports on some initial experiments on learning in multi-agent design systems. These experiments have several goals. The first is to study the ease with which simple learning techniques fit into the multi-agent paradigm we are using. The second is to determine the performance of these techniques. The third is to study the application of the multi-agent paradigm we use to "real" problems, as its development has mostly been concerned with a more theoretical view.

The design system we use is built from small knowledge-based (expert) systems that we call **Single Function Agents** (SiFAs) [Victor & Brown 1995], [Dunskus et al 1995], [Berker & Brown 1995]. There are a small set of types of agents, each of which has restricted capabilities. SiFAs will be explained in more detail below.

1.1 THE TYPE OF DESIGN PROBLEM

The type of design problem addressed is Parametric design. This occurs when the topology of the artifact being designed is already decided, and the design is to be completed by determining values for a set of parameters that specify the remaining details, such as color, surface finish, or geometry. This is a knowledge-based process, where the amount of available theoretical and experiential knowledge (e.g., heuristics or relevant design history) can vary from problem to problem.

We are concerned with non-routine or near-routine situations that have many constraints present, and perhaps tangled dependencies between parameters. Parametric design is not necessarily routine or simple.

Deciding a parameter's value can require various types of reasoning using different kinds of knowledge. At least three factors make the decision difficult, even though the range of choices may not be that large. These factors are the different design requirements, a multitude of attributes that characterize the parameter, and the dependencies between parameters.

The lack of a fixed, known order of parameters for which to decide values makes the search for a good set of values even more complex. Under these cir-

cumstances it is virtually impossible to avoid failures due to constraint violations. This leads to conflicts between the agent that decided a value and the constraint that rejected it.

To reflect the complexity involved in parameter decisions, we allow multiple design agents to have the same "job" (i.e., to provide a value for parameter X), but to have different points of view (e.g., cost, strength). The agents will often produce different values for X, leading to a conflict. To resolve conflicts, trade-offs have to be made via negotiations, with an exchange of information between the agents involved.

1.2 WHY USE LEARNING?

A multi-agent design system includes many agents with lots of potential interactions between them. Conflicts considerably increase the number of interactions needed during design. The more serious the conflict (i.e., the more difficult it is to resolve), the more messages are exchanged to find an acceptable solution. As the ratio of messages per design decision increases, the efficiency of the system decreases. High conflict situations, such as the ones we are addressing, provide a strong motivation for any technique that leads to reduced overhead. Learning can reduce this kind of overhead, by reducing the number of conflicts and/or by reducing the number of messages during interactions.

In this paper, after briefly describing other related research we will present Single Function Agents. The domain of material selection in Spring design is described next, along with its mapping to SiFAs. A discussion of conflicts in material selection motivates the description of the method of learning and its implementation. The paper concludes after a presentation of the results of our experiments.

2.0 Related work

Agents are becoming increasingly important in Artificial Intelligence and Computer Science [Wooldridge & Jennings 1994]. Design, as well as other complex tasks, have parallel decompositions that remove problems resulting from forced pre-execution serialization. Parallel decompositions allow opportunistic collaboration. These map well to agents. This has led to a slow but steady growth in the amount of research into multi-agent design systems. Examples include [Klein 1991], [Kuokka et al 1993], [Lander & Lesser 1991], [Sycara 1990], [Taleb-Bendiab & Oh 1993], [Victor et al 1993], [Werkman & Barone 1991].

The study of Conflict Resolution and Negotiation is also growing. Klein's [1991] model of conflict resolution uses a hierarchy of conflicts with the most abstract conflicts at the top and most concrete conflicts at the leaves. A corresponding hierarchy of resolution strategies allows conflicts to be mapped to resolution strategies. Klein's agents have both design and conflict resolution knowledge.

Negotiation is a common approach to conflict resolution in the design domain, and is the process by which resolution of inconsistencies is achieved in order to arrive at a coherent set of design decisions [Sycara 1990]. Negotiation proceeds with generation of a proposal, then a counter proposal based on feedback from dissenting agents, and communication of justifications and supporting evidence.

As the Single Function Agent approach is relatively new only three systems have been developed so far. I3D is a system that integrates part design and manufacturing plan production for Powder Processing applications [Victor et al 1993]. In I3D the agents were not allowed to conflict, but were in I3D+ [Victor & Brown 1994]. The SNEAKERS system was built to train users in Concurrent Engineering. The user interacted with agents that had different functions and points of view [Douglas et al. 1993].

Recently, researchers have started to work on the application of Machine Learning algorithms to multi-agent systems [Sen 1995], and design researchers have investigated learning in design systems [Maher, Brown & Duffy 1994]. However, there is still relatively little work on learning in multi-agent design systems [NagendraPrasad 1995].

3.0 Single Function Agents

A SiFA is a small knowledge-based, expert agent. It performs a single **Function**, on a single **Target**, from a single **Point of view**. For example, the selection of a material from the point of view of reliability.

A SiFA's type is determined by its *function*. The function describes what kind of information it processes and what kind of result it produces. There are a limited number of types currently allowed. The types are intended to be combinable to have the problem-solving power to do (at least) Parametric design.

The SiFA types used in previous work are:

1. *Selector:* A selector selects a value for a parameter, by picking a value from prestored or calculated possible values, according to some preferences.
2. *Estimator:* An estimator produces an estimate of the value of a parameter. Unlike selectors, estimators can work quickly and with insufficient information, so that the values they produce are just estimates of what the final value should be. They may also be imprecise, producing a range of possible values.
3. *Evaluator:* An evaluator gives a quality rating for the value of a parameter, producing a measure of goodness for that value, usually represented as a percentage or as a symbol (e.g., "good").
4. *Critic:* A critic criticizes the value of a parameter by pointing out constraints or quality requirements that are not met by the current value.
5. *Praiser:* A praiser praises values of parameters by pointing out why the value is desirable.

Each SiFA has a single *target*. The main type of target of a SiFA is a value of a single parameter of the design. But as the critiques, praises, estimations and evaluations are all treated as "first class objects", any of these entities can be the target of an agent [Berker & Brown 1995].

The *point of view* of an agent is some direction or aspect of the design that the agent considers while doing its work – often a goal that the agent is trying to achieve. Examples of points of view are cost, strength, style, weight, reliability and availability. The points of view partition the knowledge about the target.

SiFAs communicate by sending each other messages. Each agent can communicate directly with any other agent. The communication language used is based on KQML (Knowledge Query and Manipulation Language) [Finin et al 93]. The current state of the design is accessible to all agents.

Although a scheduling mechanism is required for any implementation of SiFA-based systems, no particular method is assumed by the SiFA model. SiFAs are assumed to be able to be "triggered" by satisfied preconditions. It is likely that Selectors should be given priority, and that conflict detection and resolution is more important than carrying on with the design.

3.1 WHY STUDY SIFAS?

SiFAs were originally conceived from experiences with building expert systems for Concurrent Engineering support systems (for example, see [Douglas et al. 1993] and [Victor et al. 1993]). Since that time they have gradually been refined.

They are not principally seen as a new way of building design systems. Rather, they are considered to be, and have been used as, a tool for investigating knowledge, problem-solving and conflict resolution.

They provide a context for precisely conceptualizing, generating, clarifying and categorizing:

- types of knowledge (e.g., conflict identification [Berker & Brown 1995]);
- types of reasoning (e.g., conflict detection [Berker & Brown 1995]);
- types of conflicts (e.g., Estimator-Critic [Dunskus et al 1995]);
- trade-offs in the design process (e.g., material processing vs. cost);
- what might be learned in multi-agent design systems (e.g., selection preferences [Grecu & Brown 1995]);
- knowledge for knowledge acquisition (e.g., the selection list [Currier 1995]).

3.2 WHAT SIFAS AREN'T

SiFAs are not intended to be a realistic simulation of a team of designers (such as in Concurrent Engineering). They are too fine-grained for that. At present, SiFA research has little to offer the study of multi-agent systems built from legacy code (such as [Kuokka et al. 1993]), or other large-grained design systems (such as

[Lander & Lesser 1991] and [Werkman & Barone 1991]). However, the detailed study of conflict management that SiFAs facilitate should yield general results [Dunskus et al 1995].

SiFAs are also not “overhead free”. For the same overall functionality, having many small agents, rather than fewer large agents, leads to increased overhead from inter-agent communication. Currently, the research benefits of SiFAs outweigh this disadvantage. Learning, however, can help reduce this overhead. The small grain size increases the number of potential locations where learning can occur. However, it remains to be seen if SiFAs will ever become a fully viable design system building tool, in addition to being a research tool.

4.0 Material Selection in Spring Design

A variety of types of springs are used in mechanical engineering design. Each type corresponds to a basic spring configuration. After a specific configuration is chosen, the designer has to reason about the values of the design parameters which describe the configuration. The parameter set is not necessarily the same for all the spring configurations.

Our experiments were limited to helical compression springs. The most important parameters which define these springs are the spring material, the wire diameter, the mean coil diameter and the number of coils. These are primary design parameters, meaning that they are not computed based on other spring parameters. Note that the primary design parameters are not totally independent, as they are related through design constraints. The non-primary spring parameters, such as the spring index (the ratio of the mean coil diameter of a spring to the wire diameter), are derived from the primary design parameters.

All the single function agents we used to test agent interactions targeted one single design parameter: the spring material. Material selection requires the designer to take into account a wide diversity of attributes, such as stress, electrical conductivity and cost, and to decide which of the possible materials satisfies the design requirements. The choice influences the decisions on the other spring parameters. For example, the deflection of a spring coil depends on material elasticity and will be used in deciding the number of coils. This is an example of parameter dependency.

There are about 30 materials most commonly used in spring design [Machinery 1982]. Based on their composition they are grouped in 7 categories with related physical properties. The selection of a material is determined by these factors:

1. the *temperature range,* where the material has its normal physical properties;
2. the *tensile strength,* dependent on the manufacturing process;
3. the *resistance under various shock loading conditions*;
4. the *resistance under various impact loading conditions*;

5. the *allowable working stress* given the intended service of the spring;
6. the *modulus of elasticity;*
7. the *fatigue life* – the time after which the spring fails, far below its normal elastic limit, due to continuous deflection;
8. the *endurance limit* – the highest stress, or range of stress, that can be repeated indefinitely without causing spring failure;
9. the *hardness* value that can be achieved through treatments;
10. the *electrical conductivity;*
11. the *magnetic properties;*
12. the *corrosion resistance;*
13. the *shape* and *diameter* of the wire section, from the manufacturer;
14. the *cost* of manufacturing the material.

The selection of the spring material is usually the first step in parametric spring design. As such, it influences many of the subsequent design decisions. Poor choices can lead to the need to reconsider the entire design process. Therefore, it is desirable to consider as many of the design requirements as possible when deciding the spring material.

A spring design problem does not necessarily have requirements given for all of the previously enumerated factors. Requirements impose thresholds for the admissible values of these factors. Ideally one would like to achieve an optimal value for each factor. However, this is rarely the case. The goal of the design will be to find materials which *satisfy* all the requirements. Optimality is a different issue as it is relative to various criteria. A global measure of optimality is not available. If the user needs higher standards of quality for some of the material properties, (s)he adds the corresponding constraints to the problem specification.

The material properties are not independent. For example, additional hardness can be obtained by special treatments, but these procedures raise the cost of the spring material. In order to find satisfactory solutions, designers use only some of the properties as selection guides. The rest of the material properties are used to verify that design requirements are not violated.

The knowledge about these properties is neither complete, nor uniform across the entire range of materials. Some materials simply do not exhibit one property or another (e.g., electrical conductivity, or magnetic properties). Sometimes, even when the property is known to exist, the knowledge describing it can be available in different amounts and under various representations (e.g., physical laws, graphs, or experimental data stored in tables). Therefore, given varying degrees of completeness and uniformity, analyzing a material's suitability for a design is a matter of expertise.

5.0 Material Selection with SiFAs

We approached the problem of material selection for spring design by defining a set of selectors, critics, and praisers. All of them have the same target – a material value. The points of view were chosen from the criteria enumerated in the previous section. Not all of them were included in these experiments. But for every point of view we decided to use, there was at least one agent defined.

The important questions we had to address at this point were:

- What type of agents should be defined for each point of view?
- How do we partition the available knowledge among the agents?
- Do we need to use every type of SiFA?

Since we intend to choose a material value we need to define at least one selector. Selectors are the only agents allowed to propose parameter values. Selectors are defined only for those points of view for which there exists knowledge about acceptability, as well as preferences for all parameter values. In other words, a selector should be able to distinguish, from its point of view, between *any* two materials in the range of the application.

For example, every material has a cost. Therefore, we can order all the material values from the point of view of cost, and select the most preferable value. But not all materials have magnetic properties. It isn't possible to prefer one of those materials over another from that point of view, as they don't have any.

Our experiments used two selectors:

1. The first selector proposes materials from the point of view of the working temperature range. Materials are ordered by how well they cover the temperature interval in which the spring is required to work.
2. The second selector chooses materials from the point of view of cost.

The critics are used to express objections to the choices made by the selectors. A critic is not supposed to have knowledge about all the possible choices for the parameter value. It is only supposed to point out those material values that are not acceptable given its point of view. The acceptability of parameter values may depend not only on the design knowledge of the critic, but also on the design requirements. In fact, from its point of view, a critic is capable of detecting the entire set of unacceptable values in the current design context.

We are currently using critics from the following points of view: tensile strength, resistance to shock loading, resistance to impact loading, fatigue life, and electrical conductivity.

These critics use points of view independent of the other spring parameters. As the design experiments will be extended to include all the spring parameters, critics will also be responsible for the dependencies between parameters. For

example, an availability critic for wire diameters will restrict the materials to those for which the desired wire diameters are provided.

Praisers are the complementary agents to critics. Their role is to highlight every selection which has a very good rating from their point of view. They are not required to know the entire set of high rating selections in the current design context – as opposed to critics, which do have to know all the 'bad' choices. A praiser's duty is just to mention whether the current proposal for a parameter value surpasses its internal standard. The information provided by the praiser is not critical. That means that neglecting a praiser's observation will not lead to constraint violations and failures. Praisers are useful in negotiations. Whether a proposal is praised or not can influence which of the agents in conflict should reconsider (e.g., relax) its proposal. The current version of the system uses material praisers from the following points of view: stress, endurance, corrosion resistance, and hardness.

Advisers and estimators have also been implemented. Advisers help selectors decide between values which have very similar support. The adviser that assists will have the same point of view as the selector. Since they are not critical for the illustration of the following experiments, advisers are not further mentioned.

Estimators are used in the selection process for parameter values that are connected by constraints with other parameters. Assuming that the other parameter values in the constraint have not been decided yet, they can be estimated in order to make a more informed selection for the current design parameter. As this paper focuses on the decisions associated with one single parameter, estimators are not used either.

Agents from the same functional class (e.g., selectors, critics, etc.) have heterogeneous domain knowledge representation and reasoning. The tensile strength critic uses functions extracted from plotted graphs on which to base its decisions, while the electrical conductivity critic uses data available in tables. The temperature selector uses qualitative interval matching as one of its techniques, while the cost selector simply orders the current costs.

As the target and point of view of each agent are domain dependent, they provide the interface of the agents with the design domain. The functionality of the agents (e.g., selection, critique, praising, etc.) creates a general framework of interaction among the agents independent of the underlying domain.

Finally, another important distinction which differentiates agents with the same functionality is how context sensitive they are. A material selector from the point of view of cost will use a list of preferences which is independent of the current design problem, since the costs of the materials are given and can be viewed as independent of the design specifications. In contrast, the material selector from the point of view of the temperature range is context sensitive. Its preferences vary depending on the working temperature range which needs to be

covered. The working temperature range of each available material will overlap the required range differently, thus producing a different preference rating for each material. The issue of context sensitivity will prove to be important when we discuss what agents can learn about each other.

6.0 System Design and Implementation

The spring design system is composed of several modules: the single function agents and the design board (figure 1). The design board is a module which is vis-

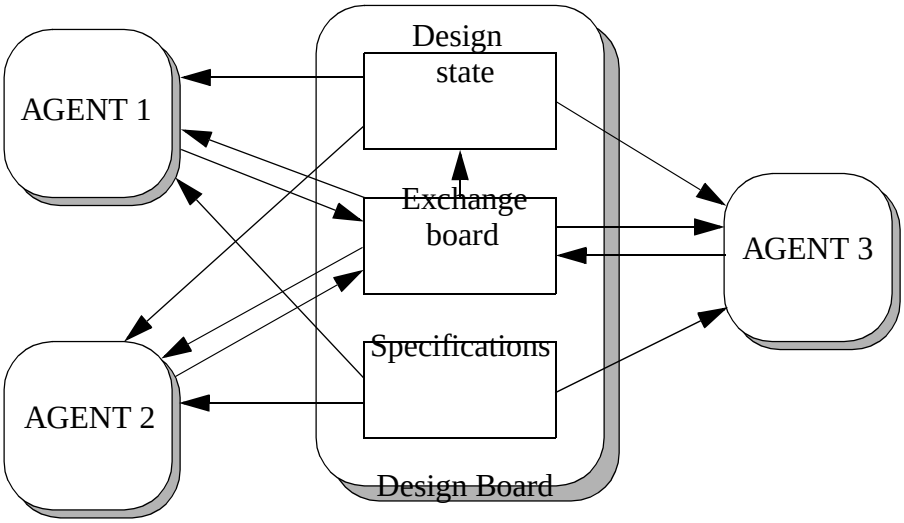

Figure 1. Architecture of the Spring design system

ible to all the agents participating in design. It is subdivided in three parts:

- the *design specifications*, including all the requirements provided by the user for the current design;
- the *design state*, which describes the design and records the parameter values decided so far;
- the *exchange board*, where agents make their proposals, engage in negotiations and reach agreements.

The agents encapsulate the domain knowledge about the corresponding target and point of view. Most of it is represented as rules and facts. Additional specialized routines carry out numerical computations associated with reasoning based on equations and physical laws. Selectors use such routines to establish their lists of preferences, and critics use them to compute the ranges of admissibility.

Every agent can see the information describing the design state and the specifications. The exchange board is used by selectors to make proposals, by critics to post objections, and by praisers to announce the superiority of a proposal, as seen from their point of view. In addition, agents can communicate directly.

An agent is activated by a set of preconditions, confirming that the information for carrying out its task is available and that the result of the task is needed. The control of its activity belongs to each agent and is not imposed from the outside. The way in which design agents work together can be easily altered through these preconditions.

The system is implemented in CLIPS [Giarratano & Riley 1994]. Each agent is a rule-based system, with the addition of specialized C functions. The design state is described in an object oriented manner, allowing uniform parameter handling.

7.0 Agent Conflicts in Material Selection

Choosing an acceptable parameter value involves the following stages:

1. *The selectors negotiate a common acceptable value for the design parameter.* Each selector computes a range of possible values and an order of preferences for these values. Selectors start negotiation with the most preferred value from their point of view.

   One of the selectors, let's call it *A*, makes a first proposal for the parameter value. The other selector (*B*) accepts the value only if it is currently the highest ranking value in its ordering of values. Otherwise, a conflict is detected and a negotiation session starts. *B* will make a counterproposal. *A* will proceed in the same manner as *B* did. The process continues until one agent makes a proposal that is also the best one for the other agent at this stage. Negotiation, also stops if the counterproposal to be sent by one selector has been previously posted by the other agent.

   Given the following material preferences of the two selectors (temperature and cost):

SELECTOR	**PREFERENCES**
Temperature	J I D E F B
Cost	B C A F E D G I

the following negotiation would take place:

SELECTOR	**PROPOSALS**
Temperature	J I D E F *agreement on F*
Cost	B C A F

The negotiation sequence can be altered by a praiser. Assume a praiser *P* praises a proposal *v* made by an agent (say *A*), and *B* does not consider the value as being the best one. Considering that *v* is claimed as a benefit from more than one point of view (*A* and *P*), *B* will match *v* against *several* preferences before making a counterproposal. Currently an agent is required to match a proposal against its 3 next best values, if the proposal is reinforced by a praiser.

For example, consider the same preferences of the same two agents. Assume that the praiser from the point of view of hardness praises material I, while the praiser from the point of view of stress praises material E. Praised values are marked in bold face. The additional values considered after a praised proposal are included in parentheses.

SELECTOR	**PROPOSALS**
Temperature	J **I** D **E**
Cost	B C (A, F) A F (E, D) *agreement on E*

The outcome is different than in the first case. Proposal I is not accepted, since the selector from the point of view of cost cannot find it among its top 3 preferences at that moment. However, proposal E is accepted, as it is found in the required range. In this situation, one selector relaxes its preferences in the light of evidence from other agents.

2. *The critics post their objections to the agreement reached by the selectors*. A critic which rejects a parameter value, will post the entire set of values that are not acceptable under the current conditions. This operation is computationally expensive and is carried out only by those critics which are not satisfied with the current decision. If at least one objection arises, a critic-selector conflict is signalled and a third stage becomes necessary.
3. *The selectors start a new negotiation round, during which they avoid use of the values considered not to be acceptable by the critics*.

   If, for example, the design problem requires high resistance to shock loading, the corresponding critic would object to the material value *E*, and point out that the values *A*, *B* and *C* are also unacceptable. The new preference table for the selectors would be:

SELECTOR	**PREFERENCES**
Temperature	J I D F
Cost	F D G I

Knowing, that I is a praised value, the new negotiation round would run as follows:

SELECTOR	PROPOSALS
Temperature	J **I**
Cost	F D (G, I) agreement

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

One of the advantages of the method proposed is that it allows multi-partite negotiation. Considering the number of agents, any method approaching conflicts in a pairwise manner would have to cope with a serious overhead and with convergence problems.

In designing our experiments we have considered two types of negotiation, as possible versions of the previous strategy:

I) *Point-to-point negotiations:* This technique is the method described above. Each agent proposes a single value at a given moment. When analyzing a proposal made by another agent, the current agent matches it against a single value – the most preferred value from the local point of view. Proposals are made and analyzed one value (point) at a time.

II) *Range-to-range negotiations:* This strategy generalizes point-to-point negotiations in two directions. An agent (*A*) proposes its preferred value at that moment and also posts a set of alternative options. The other agent (*B*) matches the proposal against its best value. In case of mismatch, *B* compares the alternatives with its own next best set (range) of values. If the two sets intersect, a value is chosen as an agreement. Otherwise, *B* will prepare its own counterproposal as a new preferred value followed by a set of alternatives.

Considering the initial preferences of the two selectors:

SELECTOR	PREFERENCES
Temperature	J I D E F B
Cost	B C A F E D G I

a negotiation session (without praisers) in which an agent posts two alternatives to its best proposal and compares an incoming proposal with its first three best values, would run as follows:

SELECTOR	NEGOTIATION
Temperature	prop: J (I, D)
Cost	comp to: B (C, A) prop: F (E, D) *agreement on D*

While this strategy is computationally more costly for both agents, it requires fewer interactions and fewer proposals by each side.

8.0 Learning in SiFAs

SiFAs generate a significant number of interactions while deciding the value of a parameter. This overhead arises because each single function agent is responsible for only for a portion of a decision, and the final decision has to gain the approval from all points of view. The number of single function agents involved in a parameter decision is not trivial. We have used 11 agents just for the material selection. This number can easily be increased if we take additional points of view into account.

The thorough exploration of the choices for the design parameter's value justifies this overhead. Designers usually prune the large number of comparisons necessary for an informed choice, by using subranges for their evaluations. Due to their specialization, SiFAs are much more powerful, provided they quickly learn to cope with situations which occur frequently.

As interactions consume a large portion of the computational effort expended on value selection, the primary goal of learning is to improve the knowledge which the agents have about each other. The learning experiments investigate:

- how difficult it is to learn about the other agents;
- how good the prediction of the behavior of the other agents will be;
- how much learning contributes to reducing the interaction overhead.

The learning results from agents interacting, and is aimed at predicting the future behavior of the interaction partners. Since agents act based on their functionality, learning can take advantage of this knowledge. Therefore, in our experiments we have made the learning strategies dependent on the type of the agent whose behavior is to be predicted.

The agents that learn are the selectors. They search in the space of values to find an acceptable solution. The other agents will 'encourage' or 'discourage' their search. It is to a selector's advantage if it can predict the feedback, and already take it into account when it evaluates alternatives, without exploring unproductive search paths. Critics' and praisers' design opinions will not be influenced by the actions of the other agents.

The generic strategy used by an agent *A* to learn about another agent *B* is:

1. *Create a case from the interaction with agent B.* Cases are indexed by the design requirements relevant for the point of view of the agent one learns about. A case records the sequence of responses of the other agent. For example, assume the learning refers to the selector from the point of view of temperature. The case will be indexed by the working temperature range required for the spring and will reflect a decreasing preference sequence of

material values of the selector from the point of temperature. The sequence is recorded only up to the point where an agreement is reached in that negotiation session.

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 agent *B* under as many conditions as possible.

The learning strategy attempts to create a model of another agent's behavior that is closely tied to the specific design conditions rather than to the details of the other agent's domain reasoning. Two arguments favor this approach:

- Single function agents are very specialized. It is virtually impossible to assume that an agent can understand another agent's domain reasoning.
- An agent is much better off quickly learning small things about the other agents. There are many agents to learn about, therefore it makes sense to look for knowledge that can be applied easy and which reduces interactions.

It is important to mention that agents only discover the other agents through the "signed" proposals, criticisms and praises posted on the design board.

8.1 WHAT CAN BE LEARNED ABOUT EACH AGENT TYPE?

The learning approach used is concept formation ([Gennari et al. 1990], [Michalski 1983], [Mitchell 1982]). Concepts reflect the connection between design conditions and the decisions based on those conditions. Whenever an agent posts a design proposal, criticism or praise, it also posts the elements from the design context on which the decision was based. If no conditions are posted, it is assumed that the decision is valid independent of any particular circumstances.

Each case recorded about an agent *A*'s behavior represents a training instance in developing a conceptual description of *A*. The concept features are the design conditions which led to *A*'s proposal, expressed as allowable design ranges and thresholds. The contents of the proposal determines the class partitioning of the concept instances.

As the heterogeneity of the domains which have to be covered by the inductive learning is extremely diverse, no assumption can be made about the continuity of the concept representations in the feature domain. For generality we have used disjunctive induction methods [Michalski et al 1986].

The dependency of the learned data on the learning of another agent raises additional issues which are as important as the accuracy of the classification. When learning about selectors the learning agent has to learn an evolving description of the other agent's behavior, as the training instances get refined in time. Even though the internal preferences of an agent repeat themselves under identical conditions, the way they are perceived from the outside varies depending on the other agents. The external perception is used to construct the model of the agent.

The classification features are the same for all the agent types, since they originate in the design. However, what is learned about an agent depends on its type:

1. Learning about praisers. Praisers point out parameter values which are particularly suitable from their point of view. Whenever a praiser praises a parameter value, the information is recorded by selectors. In the current experiments, only context-insensitive praisers were used. For example, a material is considered excellent from the point of view of hardness regardless of any other conditions or design values. Thus learning is merely storing praised values associated with praisers. If context-sensitive praisers are used the inductive technique used for critics applies to praisers too.

Praisers effectively provide only one learnable item at a time, and can only act when a selector proposes a value. This can occur during a negotiation. Consequently, knowledge about the praisers is acquired only to the extent to which proposals are praised during negotiations.

2. Learning about critics. Critics can be context-insensitive, or context-sensitive. The values they do not accept can depend on the design requirements; e.g., the resistance of a spring material to impact loading can be good or bad, depending on the conditions under which the spring is required to work. The information recorded about a critic is the set of values it considered unacceptable and the design conditions that the critic used to make those decisions. The classes correspond, in this case, to the individual materials a critic objects to. For any particular requirement(s), the information learned about the critic is complete (i.e. the critic makes all its objections known).

3. Learning about selectors. Selector behavior is the most difficult to predict, since, for a given design context/state, a selector can use different methods or sets of prestored values. Even if the set of values remains constant, the preferences among those values can be variable. The classes that partition a selector's behavior are defined by sets of parameter value preferences.

Assume that selector *A* encounters selector *B* several times, with the same design requirements for *B* each time. *A* still has reason to record *B*'s responses every time, as they might offer sparse sequences of *B*'s preferences. The sparsity can be due to *B*'s knowledge that some values will be unacceptable to critics and are therefore left out of its proposals. However, the critics may have different requirements each time and thus the 'holes' in *B*'s sequence of proposals will be different. In addition, the recording is incomplete because it stops when an agreement is reached. *A* will integrate the current response sequence of *B* with the sequence compiled from the previous encounters under the same conditions.

The main issue is that, while being refined, concepts can 'migrate'. Different sequences of responses can become identical after several interaction sessions,

merging into a single concept. Alternatively, concepts can be split if under some particular subconditions the refined responses become different.

8.2 HOW DO AGENTS USE THE LEARNED KNOWLEDGE?

The knowledge learned by an agent about the other agents is used differently, depending on the type of the agent to which it refers:

Knowledge about selectors. During a negotiation a selector will decide on its next proposal. Before posting its proposal the selector will see whether it has knowledge about the behavior of the other selectors under these design conditions. If so, the selector will anticipate their proposals and prepare a new proposal. A new evaluation of the next-best proposals of the other selectors will be used to determine their responses. The predictive reasoning continues until either one of the following conditions is fulfilled:

- a value is found which is likely to be agreed on by all the selectors;
- for one of the other selectors no further preferences are known.

The selector will do predictive reasoning only if it knows how *all* the other selectors will respond. This condition is not imposed on other agent types.

If the proposal a selector submits after predictive reasoning is not known to be an agreement, it will be accompanied by the last value taken into account for each of the other selectors. This will allow the other selectors to know where to continue with their proposal evaluation.

The incompleteness of the knowledge about a selector can cause an undesired phenomenon: *A* will assume that the responses learned from *B* are consecutive preferences. This may cause possible solutions to be overlooked, as *A* will unknowingly skip them in its prediction of *B*'s behavior. We have assumed that selectors can always reach an agreement, unless either *A* or *B* have insufficient knowledge about each other. Lack of agreement initiates a new negotiation in which they will not use the knowledge they have about each other. The new sequence of responses is used by each agent to correct its knowledge about the other selector.

Knowledge about critics. Knowledge about the critics is used to eliminate proposals known to be unacceptable. This holds true for the proposals which the current selector intends to submit, as well as for the proposals it anticipates from the other agents. Even if there is no knowledge about how a critic will respond, the selector continues its anticipatory reasoning. The critic will eventually 'protest' about an unacceptable agreement of the selectors.

Knowledge about praisers. The knowledge about praisers is used in a very similar manner to that of the critics. During the anticipation of the other selectors' proposals, praiser opinions will be taken into account to find out which agent has to revise its counterproposal, assuming that no agreement is anticipated.

These methods attempt to ensure that what has been learned is used effectively, and that information is exchanged in a cooperative fashion.

9.0 Experimental Results

Our experiments used eleven SiFAs (see Section 5). The two selectors, five critics and four praisers encoded knowledge about the 20 materials that are considered representative for helical compression springs. This makes the material choice problem non-trivial, due to the many comparisons from various points of view.

The design problems used included design constraints for each selector and for each critic. This made their proposals and responses context-sensitive. In these experiments the measure used for evaluation was the *number of interactions* needed to generate a material value accepted by the selectors as well as the critics. By an "interaction" we mean a proposal posted by a selector, an objection posted by a critic, or a praise posted by a praiser.

The first type of analysis required running the system through a sequence of 22 generated design problems. One of the problems was considered a reference problem, while each of the other 21 problems introduced a change in the requirements affecting the reasoning of one selector or one critic, relative to the preceding problem. A run through all the 22 problems is called an "experiment". Several experiments were carried out, with the same changes in the requirements ordered differently. During each experiment an agent encountered three changes in the design conditions affecting its proposals.

At first, the changes in the design problems in an experiment were made in random order. There were 22 experiments in all. Each experiment was run with and without learning capabilities. Table 1 summarizes the average results for the set of experiments.

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

Type of analysis	Average number of interactions (rounded to closest integer) after			
	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 explained by the fact that the selectors do not use predictions if they have no information about the behavior of the other selector in the new situations. The initial decrease in the number of interactions is due mainly to the information learned about the critics.

Another type of analysis involved scheduling the changes affecting selectors first, and then making the other changes in random order. The results of this set of

runs are summarized in Table 2. The numbers in parentheses represent the interactions due to selectors.

The major improvement can be seen towards the end of this run. The learning of selectors about selectors happened mostly during the first six experiments and without any changes in the behavior of critics and praisers, as their requirements did not change during that time. As a consequence, interactions during the next five experiments were reduced mostly due to shorter negotiations between selectors. Interactions due to praisers are reduced only to the extent that the negotiations between selectors become shorter and the praised material values occur less often on the average.

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

Type of experiment	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)

For all of the previous experiments, the changes in requirements were close to the ones in the reference design problem. The primary goal was to test the decrease in the number of interactions under the assumption that the variations can be captured relatively fast. 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. Even though the reduction of the amount of interaction was smaller than in the case where learning extended to all the agents, accuracy of prediction proved to be better.

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 explanation relies on the fact that learning without interference of critics generates an accurate partitioning of the design requirements into concepts. Adding the critics does not change the conceptual partitioning, but causes the concepts to be refined.

The experiments described so far used point-to-point negotiations. The first type of experiment was also done using range-to-range negotiations. An agent considered two additional options besides its most preferred one. Table 3 reflects the learning results for the range-to-range interactions.

The design problems were randomly ordered. The learning rate was slower than in the point-to-point negotiations, as a much larger amount of negotiation

was needed to capture correct information. This is mostly due to the 'hidden' preferences of a selector (due to the look-ahead), which were not seen by the other selector. The number of failures (no agreement reached and renegotiation

TABLE 3. Decrease of interactions in range-to-range negotiations (random problem ordering)

Type of analysis	Average number of interactions (rounded to closest integer) after			
	6 expts	12 expts	17 expts	22 expts
without learning	21	22	21	21
with learning	19	17	16	15

without using the learned knowledge) was higher (5 cases vs. 2 cases in point-to-point negotiations). However, rerunning the system over a total of 50 experiments, with the same changes in the requirements affecting the agents, but combined in different ways, led to interaction reductions that compared well with the point-to-point case.

10.0 Conclusions

We want a system to learn to reduce the design overhead in the application range needed by a specific user. Designers will most often use the system only in a particular region of the design space. Pre-training the system is not really possible, as it is hard to foresee the class of problems explored by a particular user. Therefore it is important to have the system learn from applications.

Although the order of presentation of design problems can make the learning faster, note that similar final results were obtained by a random ordering of the design problems seen by the system. This is important, and a positive result.

A factor that makes experimentation hard, and clear conclusions difficult, is that the learning processes are not independent. One agent learn something, can delay the occurrence of situations leading to learning by another agent. The range of phenomena of this type has yet to be sufficiently explored.

We have dealt with interactions between agents having the same target. There will be many additional aspects to be considered for the entire parameter set for spring design. For example, the knowledge representation will have to be extended to handle domain types specific to various parameters.

Advisors and estimators will also require the model to be enhanced. Advisors raise the interesting issue that their behavior is modelled in conjunction with the behavior of the agent they advise. Predicting estimator behavior will be more complex as it is an example of a situation where an agent attached to one parameter is needed in decisions related to another design parameter.

The general conclusion of this work is that SiFAs, coupled with relatively simple negotiation schemes, provide a good basis for interesting experiments in learning in multi-agent design systems, but that much more remains to be done.

References

Berker, I. and Brown, D.C.: 1996, Conflicts and Negotiations in Single Function Agent Based Design Systems, submitted to: *Concurrent Engineering: Research and Applications*, special issue on Multi-Agent Systems in Concurrent Engineering, D. C. Brown, S. Lander and C. Petrie (eds.).

Currier, P.M.: 1995, *SiFAKA: Knowledge Acquisition for Single Function Agents*, Major Qualifying Project, Comp. Science Dept., WPI, Worcester, MA.

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

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

Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzon, R., McKay, D., McGuire, J., Pelavin, R., Shapiro, S. and Beck, C.: 1993, *DRAFT Specification of the KQML Agent-Communication Language*, The DARPA Knowledge Sharing Initiative External Interfaces WorkingGroup.

Gennari, J.C., Langley, P. and Fisher, D.: 1990, Models of Incremental Concept Formation, *in* J. Carbonell (ed.), *Machine Learning – Paradigms and Methods*, MIT Press, Cambridge, MA, pp.11-61

Giarratano, J. C. and Riley, G.: 1994, *Expert Systems: Principles and Programming*, 2nd ed., PWSx Publishing Co., Boston, MA.

Grecu, D. L. and Brown, D. C.: 1995, Design Agents That Learn, *AI EDAM: Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, special issue: "Machine Learning in Design", A. H. B. Duffy, M. L. Maher and D. C. Brown (eds.), Cambridge University Press, to appear.

Klein, M.: 1991: Supporting Conflict Resolution in Cooperative Design Systems, *IEEE Transactions on Systems, Man, and Cybernetics* **21**(6), pp.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**, pp. 137-146.

Lander, S. E. and Lesser, V. R.: 1991, Customizing Distributed Search Among Agents with Heterogeneous Knowledge, *Proc. 5th Int. Symp. on AI Applications in Manufacturing and Robotics*, Cancun, Mexico.

Machinery's Handbook, 1982, revised 21st ed., Industrial Press Inc., New York, NY.

Maher, M. L., Brown, D. C. and Duffy, A.H.B. (eds.): 1994, *AI EDAM: Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, **8**(2), special issue on Machine Learning in Design.

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

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

Mitchell, T. M.: 1982, Generalization as Search, *Artificial Intelligence*, **18**, pp. 203-266.

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

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

Sycara, K. P.: 1990, Cooperative Negotiation in Concurrent Engineering Design, in *Cooperative Engineering Design*, Springer Verlag Publications, pp. 269-297.

Taleb-Bendiab, A. and Oh, V.: 1993, Speech-Act based Communication Protocol to Support Multi-Agent Cooperative Design Systems, *Proc. 1993 AI in Engineering Conf.*, CMI Publ.

Victor, S. K., Brown, D. C., Bausch, J. J., Zenger, D. C., Ludwig, R. and Sisson, R. D.: 1993, Using Multiple Expert Systems With Distinct Roles in a Concurrent Engineering System for Powder Ceramic Components, *in* G. Rzevski, J. Pastor and R. A. Adey (eds.), *Applications of AI in Engineering VIII*, vol. 1, *Design, Methods and Techniques*, Proc. 8th Int. AI in Engineering Conf., Toulouse, France, Elsevier Press, pp. 83-96.

Victor, S. K. and Brown, D. C.: 1994, Designing with Negotiation Using Single Function Agents, *in* G. Rzevski, R. A. Adey and D. W. Russell (eds.), *Applications of Artificial Intelligence in Engineering IX*, Proc. 9th Int. AI in Engineering Conf., PA, USA, July 1994, CMI Publ., pp. 173-179.

Werkman, K. J. and Barone, M.: 1991, Evaluating Alternative Connection Designs Through Multi-agent Negotiation, *in* D. Sriram, R. Logcher and S. Fukuda (eds.), *Computer Aided Cooperative Product Development*, Lecture Notes Series, No. 492, Springer Verlag, pp. 298-333.

Wooldridge, M. and Jennings, N. R.: 1995, Intelligent Agents: Theory and Practice, *Knowledge Engineering Review*, **10**(2), pp. 115-152.