Negotiated Search: A Framework for Cooperative Design

Susan E. Lander and Victor R. Lesser
Computer Science Department
University of Massachusetts

COINS Technical Report 91-79
November 1991

Abstract

An important aspect of cooperative design is the resolution of conflicts among the multiple, heterogeneous agents that contribute to a design. Existing approaches to conflict resolution view it as occurring outside of the cooperative search process of the agents. The appropriateness of specific conflict resolution strategies and the time when these strategies should be invoked are related to both the local and composite search processes of agents. A framework, *negotiated search*, is introduced that treats conflict resolution as integral part of cooperative search. This framework is described through a multi-agent system testbed, TEAM, that is configured for the design of steam condensers. Experimental results are presented that show communication and aggregation of information by agents about their local search results and the causes of conflict have a demonstrable impact on both processing time and quality of solutions. Further, we demonstrate that varying the control policies for relaxation of solution requirements in response to conflict effects solution quality and processing time.

*This research was supported by DARPA under Contract #N00014-89-J-1877 and by a University Research Initiative Grant, Contract #N00014-86-K-0764.

Negotiated Search: A Framework for Cooperative Design

Susan E. Lander and Victor R. Lesser

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003
Email: lander@cs.umass.edu

Theme: Coordination as Search

Abstract

An important aspect of cooperative design is the resolution of conflicts among the multiple, heterogeneous agents that contribute to a design. Existing approaches to conflict resolution view this resolution as occurring outside of the cooperative search process of the agents. The appropriateness of specific conflict resolution strategies and the time when these strategies are invoked are related to both the local and composite search processes of agents. A framework, *negotiated search*, is introduced that treats conflict resolution as integral part of cooperative search. This framework is described through a multi-agent system testbed, TEAM, that is configured for the design of steam condensers. Experimental results are presented that show communication and incremental aggregation of information by agents about their local search results and the causes of conflict have a demonstrable impact on both processing time and quality of solutions. Further, we demonstrate that varying the control policies for relaxation of solution requirements in response to conflict effects solution quality and processing time.

1 Introduction

In this paper, we are concerned with how to resolve conflicts occurring in cooperative multi-agent problem solving. Conflict resolution is often discussed as a separate topic in AI literature [4, 7, 12, 13]. However, multi-agent conflict resolution does not naturally occur in a vacuum; it is always called upon as part of a larger process that we call *negotiated search*, the process of finding mutually-acceptable solutions. In negotiated search, conflict resolution is an integral part of multi-agent problem solving. Other researchers are also looking into integrated approaches to conflict resolution and problem-solving [5, 9, 10].

We assume that agents are heterogeneous but share a common goal of finding the best global solution possible. Agents are *reusable* in the sense that they are not designed to work within a predefined agent set. Instead an agent is developed to represent an area of expertise. It can then be incorporated into a dynamically-formed set of agents to work on any problem that requires that expertise. Conflicts occur not as a result of hostile behavior but rather because they are inherent in the nature of the agent set. Conflicts can be due to:

This research was supported by DARPA under Contract #N00014-89-J-1877 and by a University Research Initiative Grant, Contract #N00014-86-K-0764.

- inconsistent knowledge among agents;
- incomplete knowledge and/or incorrect assumptions;
- different problem-solving techniques;
- different criteria for evaluating solutions.

Given this model, it is not possible to anticipate and engineer out all potential conflicts at development time since it is not known what knowledge will be contained in the complete system [3]. Instead of dealing with conflict through knowledge engineering, the agent must consider conflict to be an integral part of the problem-solving process and address it explicitly as it occurs.

There are two basic ways to characterize conflict resolution in negotiated search: *search* and *relaxation*. The first, *search*, "sidesteps" a conflict by extending the local searches of one or more agents until a solution is found that does not conflict. Search methods are used when it is believed that a better solution can be developed if one or more agents continue to examine their local solution spaces for additional solutions. A good example of a search strategy is presented by Conry et. al. [1]. A formal model is described for incrementally developing an understanding of the global search space using communication of local information among agents. Through ongoing interaction, non-conflicting solutions will eventually be found if they exist.

The second negotiated search strategy, *relaxation*, occurs when one or more agents relax some requirement on a solution, thereby expanding the local search space (and possibly making a previously developed solution that was judged unacceptable now become acceptable). Relaxation methods are utilized when it seems likely that the global solution space is overconstrained or when the expense of further local search is unjustified. There are many different methods for achieving these ends that can be applied in specific situations. We describe some of these in detail in Section 3.

The two primary questions that must be answered in negotiated search are:

1. What strategies can be used to develop mutually-acceptable solutions?
2. When should these strategies be applied?

The ultimate goal of our research is to develop a theory that answers these questions based on the characteristics of the local search space of agents and of the composite of these spaces (the global search space). As a first step towards developing this theory, we have been empirically exploring these questions through a multi-agent system testbed, TEAM, that is configured for the design of steam condensers.

In TEAM various search and relaxation strategies can be enabled or disabled by individual agents and thresholds can be set to change the control behavior of the agents. This allows us to examine both the functionality of various strategies and the effect of different control plans for those strategies. The major focus of experiments discussed in this paper is to show that communication among agents about their local search results and the causes of conflicts has a demonstrable impact on both processing time and quality of solutions. In addition, no matter how sophisticated an agent's search and communication capabilities are, there are cases in which no mutually-acceptable solution exists. In these cases, relaxation methods are required. We demonstrate that varying the control policies for relaxation of solution requirements in response to conflict effects solution quality and processing time. In this paper, we use a simple relaxation strategy (described in Section 4.3): other work [2, 6, 10] describes more sophisticated operators for relaxing solution requirements.

Section 2 introduces TEAM. Section 3 describes how local proposals are generated and merged in the TEAM system and how this process realizes a negotiated search. Specific search and relaxation strategies that are currently implemented for TEAM are also presented in Section 3. Section 4

discusses the setup and results of experiments on the use of various negotiated search strategies and control policies for relaxation. Finally, Section 5 explores the implications of the behavior we see in the experiments and presents some ideas about the long-term contributions of this work.

2 The TEAM Domain: Multi-Agent Parametric Design of a Steam Condenser

TEAM is an application system that performs parametric design of steam condensers under a set of user-defined specifications. In parametric design, the general form of the artifact being designed is known, but the designer must find values for a set of variable parameters. Much of the application knowledge in TEAM was originally developed by Meunier [8] for use in an *iterative respecification* system. The original agents were designed to work in a hierarchically structured environment and did not include any mechanisms for explicit peer negotiation.

Figure 1 shows the general form of a steam condenser, comprising a pump, heat exchanger,

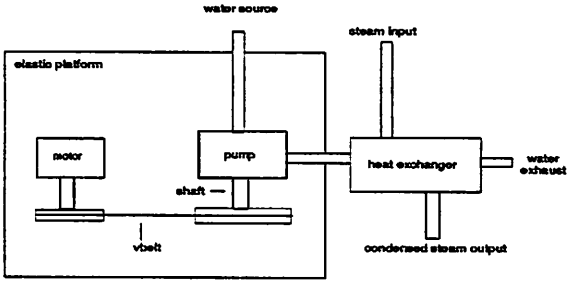

Figure 1: A Steam Condenser

motor, platform, shaft and v-belt. TEAM agents produce *proposals* where each proposal is a design for a single component of a condenser. Each component is designed by a separate agent, for example, PUMP AGENT produces pump proposals. The components are independent except for shared parameters which represent the interface points of the design. The values for these parameters must be acceptable to all agents that use them. For example, the shared parameters of a pump component include `water-flow-rate` and `power`. The pump and heat exchanger share the `water-flow-rate` parameter (water flows between the pump and heat exchanger) and the pump and motor components share `power` (the motor must deliver sufficient power to run the pump).

The TEAM architecture is shown in Figure 2. Agents are distinct entities that can view the shared blackboards. An agent cannot write directly on the design blackboard, but can send messages to the framework monitor asking that a design change or addition be made. A set of framework knowledge sources (FKSs) make required changes and do bookkeeping tasks on the blackboard objects (for example, totalling the cost of a design when a new component is added). During processing, each agent is given the opportunity to execute, then the FKSs execute, updating the shared blackboards.

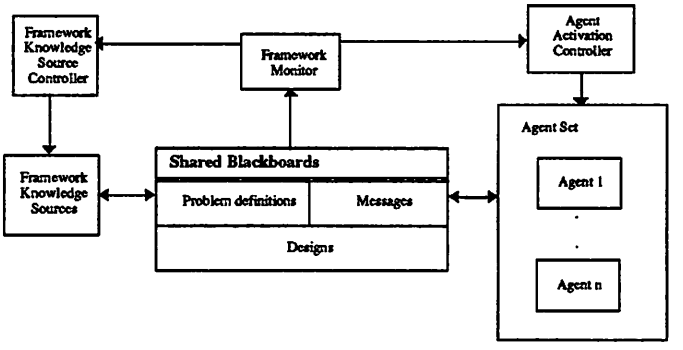

Figure 2: The TEAM Architecture

3 The Structure of Search in TEAM

3.1 Generating Anchor Designs

Problem solving begins in TEAM when a problem definition is specified by the user. All agents receive the problem definition and choose whether or not to respond to it. In general, if the problem is too underconstrained by the triggering object for an agent to develop a reasonable proposal given its specific requirements, the agent may choose to wait until a later point in the processing. New constraints will be added by other proposed components and will eventually provide the constraining information desired. Figure 3 shows the internal task structure of an agent in response to a problem definition. Each responding agent uses a default local search strategy, *generate-best-proposal*, to

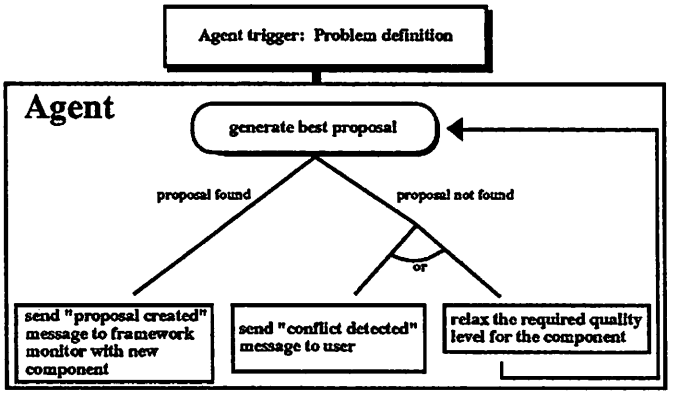

Figure 3: Responding to a Problem Definition

generate a component proposal that fulfills the user-defined specifications. *Generate-best-proposal* is a general strategy to search an agent's local solution space. The boundaries of the local search space are defined by the agent's local constraints and by the attribute values specified in the problem definition. The actual type of search used is left to the agent implementor based on the domain characteristics of the agent. The agent generates a proposal that meets or exceeds the current required quality level if it can find one. Required quality level is set to *excellent* at the start of processing. Its value can change dynamically when necessary, e.g., if no solutions exist at the current required value. When this occurs, the agent must choose whether to inform the user that the problem definition is problematic or to immediately lower its own required quality

level. In the experiments we present, the agent lowers its required quality level and again applies *generate-best-proposal*. If no feasible proposal can be found at any level, the agent must inform the user that the problem definition is infeasible.

The initial proposal generated at this point in processing reflects the agent's selfish perspective: it represents an agent's "ideal" component under the constraints imposed by the user-defined problem definition. Once an initial proposal is found, it is sent to the framework monitor which creates an "anchor" condenser design using the specified attribute values of shared parameters. Figure 4 illustrates the anchor designs created from the component proposals of PUMP AGENT and HEATX AGENT in response to a problem definition. These represent different starting points for designing a condenser, each the preferred starting point of its originating agent.

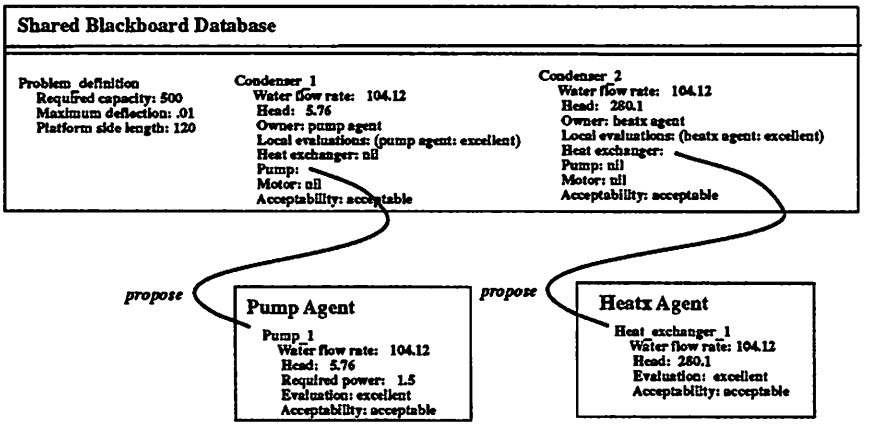

Figure 4: Initial Anchor Proposals

3.2 Responding to Anchor Designs

As anchor designs are generated on the shared blackboards, they are available to all agents. Any agent that can respond to an anchor design by adding a component initiates a task to do so. For example, in the problem above `Condenser_1` is owned by PUMP AGENT (meaning that the original proposal that triggered the creation of `Condenser_1` was generated by PUMP AGENT). It still requires a heat exchanger component, so HEATX AGENT initiates a task to generate a component that will "fit" `Condenser_1`.

The negotiated search strategy used to generate components that are constrained by specific values of shared parameters is *generate-matching-proposal*. An agent will return a proposal with the specified parameter values if a feasible one exists. The quality level of the proposal will be as high as possible, but need not meet the required quality level of its agent. In other words, the responding agent will generate the most highly-rated proposal it can find that matches the attribute values of the anchor design.

The internal task structure of an agent in response to an anchor design is shown in Figure 5. As shown, there are three possible results to the *generate-matching-proposal* strategy: 1) a matching proposal is generated that meets the required quality level of its owner (the agent that creates the proposal) ; 2) a matching proposal is generated that does not meet the required quality level of its owner; and 3) no proposal can be generated that matches the values of the shared attributes specified in the anchor design.

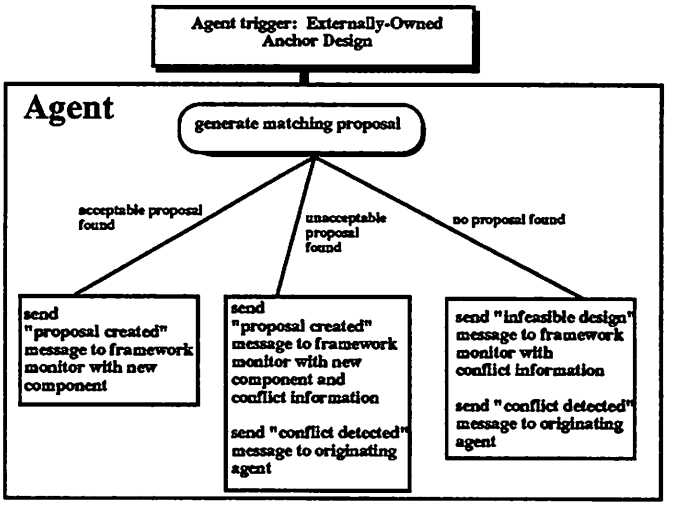

Figure 5: Responding to An Anchor Design

In the first case, the acceptable proposal is sent to the framework monitor which creates an extended version of the anchor design. The original anchor also remains available, but is made inactive for the time being. If the extended design is complete (it contains all of the components that can be generated under the current configuration of agents), and all components are locally acceptable to their owners, and the global utility value of the complete design exceeds some user-specified threshold, then this condenser is considered an acceptable design. The number of acceptable designs that will be presented to the user is controlled by a user-specified variable. For our experiments, we used a default value of 3. When an acceptable design is found, the framework monitor checks to see if the number-of-acceptable-designs requirement is fulfilled and, if so, processing ends and the finished designs are presented to the user. If there are not enough acceptable designs yet, processing continues. If an agent has no other work to do on a cycle, it will create a new anchor proposal using the *generate-best-proposal* strategy.

The second possible result to the *generate-matching-proposal* strategy is that a feasible matching proposal is generated that does not meet the required quality level of the responding agent. In other words, the best proposal that matches the specified attribute values is unacceptable to its creator. In this case, if the responding agent can "explain" the lower quality rating, it will attach that information to its response. For example, in Figure 6, the attribute value for head in `Condenser_2` is 280.1. PUMP AGENT has a constraint:

```
Constraint_1:
    predicate:  (head <= 145)
    flexibility:  5
    quality:  excellent
```

The best pump component design found by PUMP AGENT violates `Constraint_1`. PUMP AGENT responds to `Condenser_2` with a message to HEATX AGENT that a conflict was detected and a message to the framework monitor containing both the generated pump design and the violated constraint. It isn't always the case that an agent can send information that explicitly pinpoints the cause of the conflict. When this happens, the agent sends a message that a conflict was detected to the owner of the anchor design and sends its (unacceptable) proposal to the framework monitor which adds the proposal to the anchor design. For example, in Figure 6, HEATX AGENT sends a

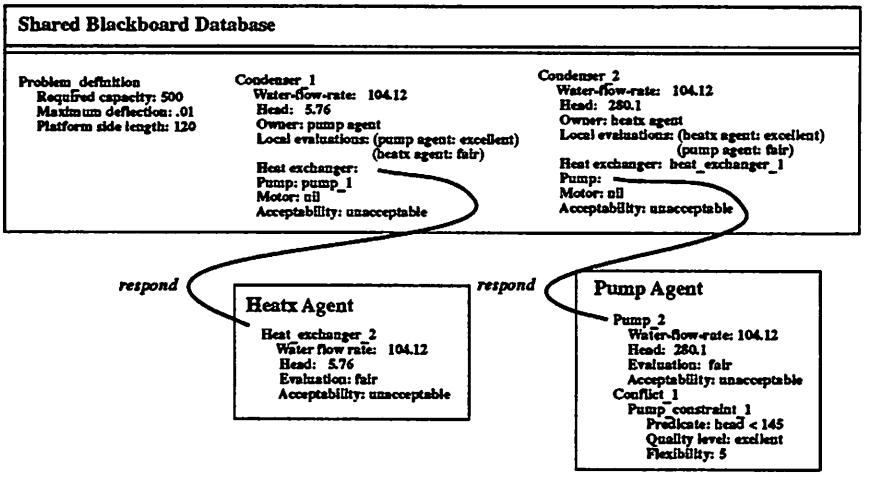

Figure 6: Matching Proposal Generation

conflict message to PUMP AGENT but does not send a conflicting constraint. Note that although this anchor is currently considered an unacceptable design, it has the potential to become acceptable at some future time if the required quality levels of the agent or agents that have given it an unacceptable rating change.

The third possible result to the *generate-matching-proposal* strategy is one in which the responding agent cannot generate a feasible proposal given the specified values of input parameters. In this case, the responding agent sends a message to the owner of the anchor design and the framework monitor containing its violated constraints (when possible). The framework monitor marks the anchor design as *infeasible*.

3.3 Responding to Conflict Detected by Another Agent

In Section 4, we will present two sets of experiments that have different negotiated search strategies enabled. One set does not take advantage of conflict information that is returned by responding agents (conflict messages are ignored) while the other uses that information to guide further local search. We hypothesized that incorporating constraining information from external agents would narrow the boundaries of the local search space of the receiving agent, thereby focusing the search and resulting in faster and more effective processing. Sycara [11] describes a model of distributed constrained heuristic search in a job-shop scheduling domain that uses communicated information about local requirements to develop a shared characterization of resource utilization. This characterization is then used to focus local search on critical resources and activities. Although the system is very different from TEAM, in both domain characteristics and problem-solving paradigm, it has the same flavor of building a local view of shared resources that can be used to guide local search.

To test the value of incorporating nonlocal information into local search, we developed a negotiated search strategy, *generate-bounded-proposal*, that takes advantage of communicated external constraints. When this strategy is enabled, an agent gathers constraining information from responses to some anchor design it initiated. This information is analyzed and incorporated into the agent's local search if possible. The response of an agent to a conflict detection from another agent is shown in Figure 7.

When the agent begins processing a conflict detection, it first looks to see if any new information has been sent. In some cases, the detecting agent is unable to provide any constraining information,

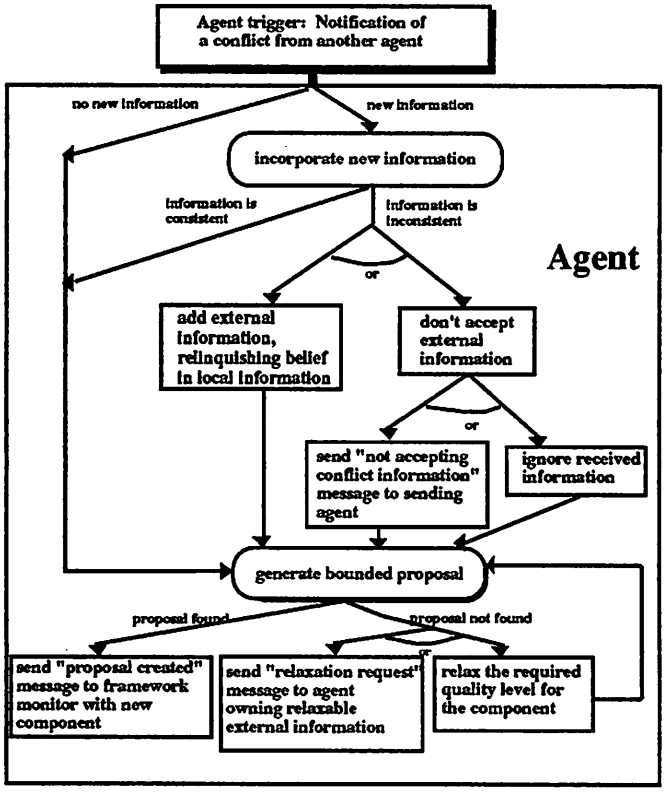

Figure 7: Response to a Conflict Detected by Another Agent

in others, the detecting agent will send constraining information that is already known to the receiving agent, and in some cases, the constraining information is new to the receiving agent. If no new information is received pertaining to a conflict, the receiving agent applies *generate-bounded-proposal* to its existing search space. An agent keeps a history of proposals it has previously submitted and will attempt to find a different solution under the same bounding conditions.

When new information is received, that information is incorporated into the agent's search space boundaries. In the simplest case, the information does not conflict with any existing internal boundaries. For example, suppose HEATX AGENT receives `Constraint_1` from PUMP AGENT as in Figure 6. `Constraint_1`'s predicate is `(head <= 145 )`. The most stringent constraints HEATX AGENT currently has on the value of head are:

```
Constraint_H1:                    Constraint_H2:
   predicate:  (head <= 809)         predicate:  (head >= 0.1)
   flexibility:  0                   flexibility:  0
   quality:  >=poor                  quality:  >=poor
```

Incorporating the constraint from PUMP AGENT narrows the set of potential values of head without conflicting with the current constraints. HEATX AGENT can then generate a new anchor proposal within the new boundaries.

In some situations, external constraining information cannot be reconciled with existing constraints. If an incoming constraint has less flexibility than a conflicting internal constraint, the internal constraint is relaxed. Otherwise, the external constraint is ignored. A more appropriate response would be to inform the agent that sent the conflicting constraint of the conflict, thereby

enabling another level of negotiation. This response capability is being implemented at the current time.

Assuming that no conflict occurred when the external constraints were analyzed or that a conflict occurred but some reconcilation has been accomplished, the agent tries to generate a design, with previously untried parameter values that fall within the new bounding constraints. This is sometimes straightforward: a new proposal is generated, sent to the TEAM monitor, and a new anchor design is built. The new anchor is more likely to be acceptable to the agents that responded to the initial anchor since it is bounded by the constraining information that was sent. However, it is not necessarily acceptable. The current set of agents in TEAM communicate only constraints on attribute values. A situation that frequently occurs is that there are dependencies among attributes that aren't captured in the constraints. For example, PUMP AGENT calculates the value of head as a function of the value of `water-flow-rate` and this functional relationship cannot be explicitly communicated to another agent. Therefore any anchor design owned by another agent that assigns values to those parameters may not satisfy the required relationship. This relationship could conceivably be communicated in some form, but that would require both agents to have specialized capabilities. The generalized TEAM framework is neutral to the existence of these capabilities: the more information that can be shared, the more it can be used, but it is not required.

The application of *generate-bounded-proposal* can result in no proposal of the current required quality level being found. Again, because of dependencies that are not captured in the constraining information, it may be that all constraints are consistent but that there are no actual solution points for some attribute within the boundaries. The agent must choose whether to relax its internal quality or to get another agent to relax some constraint.

4 Problem-Solving in TEAM: Experiments

We describe a set of two-agent experiments. In each experiment, the system is required to produce at least three designs that are considered mutually acceptable since, in a design domain, it is worthwhile to present the user with multiple potential designs representing different trade-offs. These designs are rated by TEAM and presented in order of their rating. In reporting the results of experiments, we present the global utility value of each acceptable design, and the average global utility of all acceptable designs. We also present the number of "processing cycles" that occurred. During processing, an agent cycle is run in which each agent is given the opportunity to execute, followed by an framework knowledge source (FKS) cycle in which all triggered FKSs execute. Each cycle of agent execution followed by FKS execution is called a *processing cycle*. Finally, we report the required quality level at both agents at the end of processing, i.e., what quality level each agent has agreed to accept for its components.

In the first set of experiments, *experiment-set-300*, the user specification sets the value of the parameter, `required-capacity`, at 300. Under this specification, the solution spaces of HEATX AGENT and PUMP AGENT are depicted in Figure 4. This figure was developed by generating the best solution for each of the agents at sample values for the parameters `water-flow-rate` and head over the entire range of possible values. The solutions were then plotted according to their locally-calculated quality. For example, at the point $(30, 50)$ (points are of the form (`water-flow-rate`,head)), there is both an excellent quality pump and an excellent quality heat exchanger. A solution space plot is presented for four different experiment sets in which the solution spaces were changed by changing the value of the required-capacity parameter.

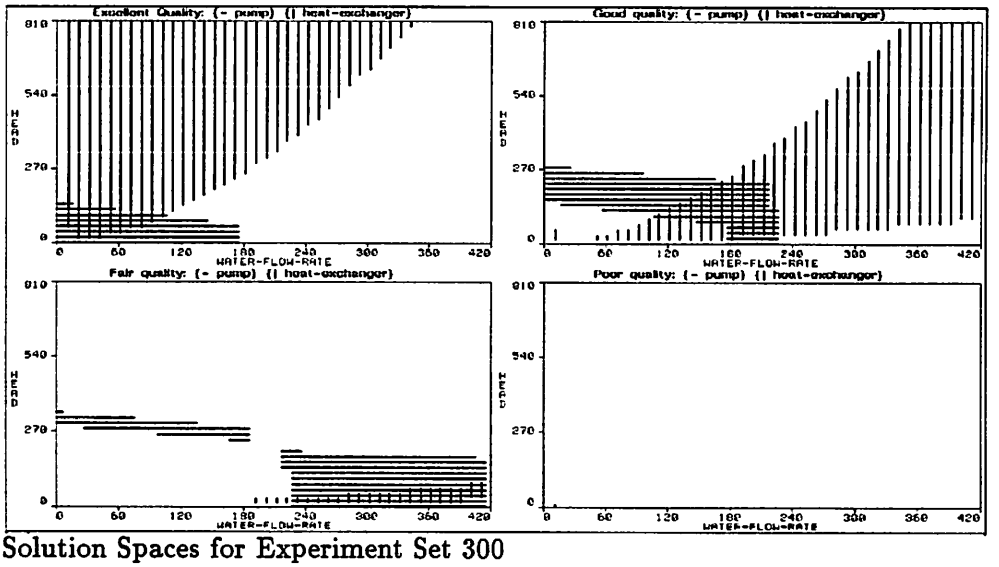

Figure 8: Solution Spaces for Experiment Set 300
Pumps (—) and heat exchangers (|) generated by PUMP AGENT and HEATX AGENT respectively are superimposed on a shared space over the parameters `water-flow-rate` and head. Each window represents components of a particular quality: *excellent*, *good*, *fair*, and *poor*

4.1 Experimental Results

We ran four experiment sets: experiment-set-300 (Figure 4), experiment-set-500 (Figure 9), experiment-

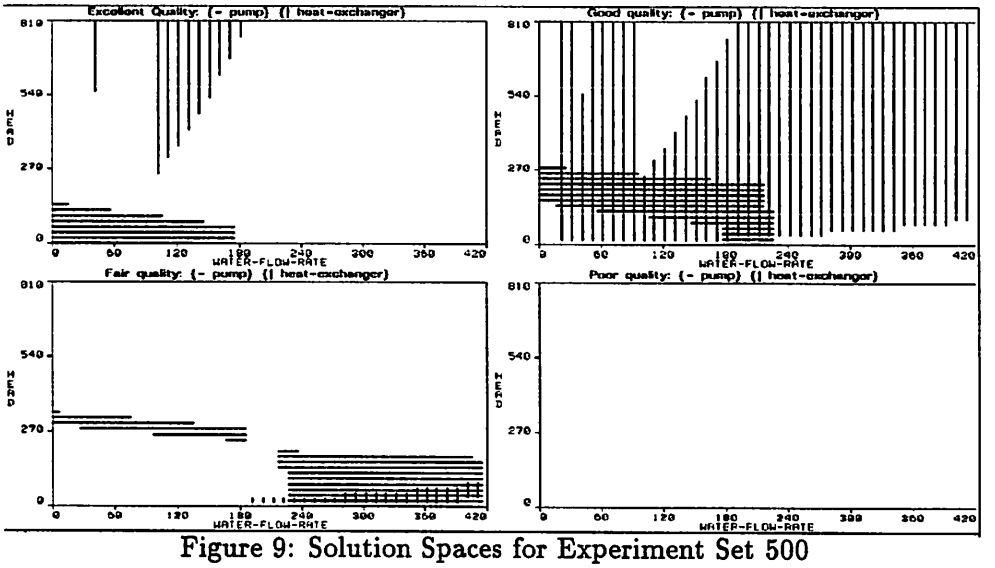

Figure 9: Solution Spaces for Experiment Set 500

set-700 (Figure 10), and experiment-set-900 (Figure 11). Each of these experiment sets is divided into two search classes: *random search* and *limited communication search.*

In the random search sets, the only negotiated search strategies enabled are: *generate-best-proposal*, *generate-matching-proposal*, and *drop-quality-level*. In the limited communication search sets, *generate-bounded-proposal* is also enabled. The character of search in the random sets is that conflict information is ignored. On each processing cycle, an agent either produces a new anchor proposal or it produces a matching proposal for some other agent's anchor design. Each new anchor proposal will be distinct from any existing proposals, but will not take into account any responses

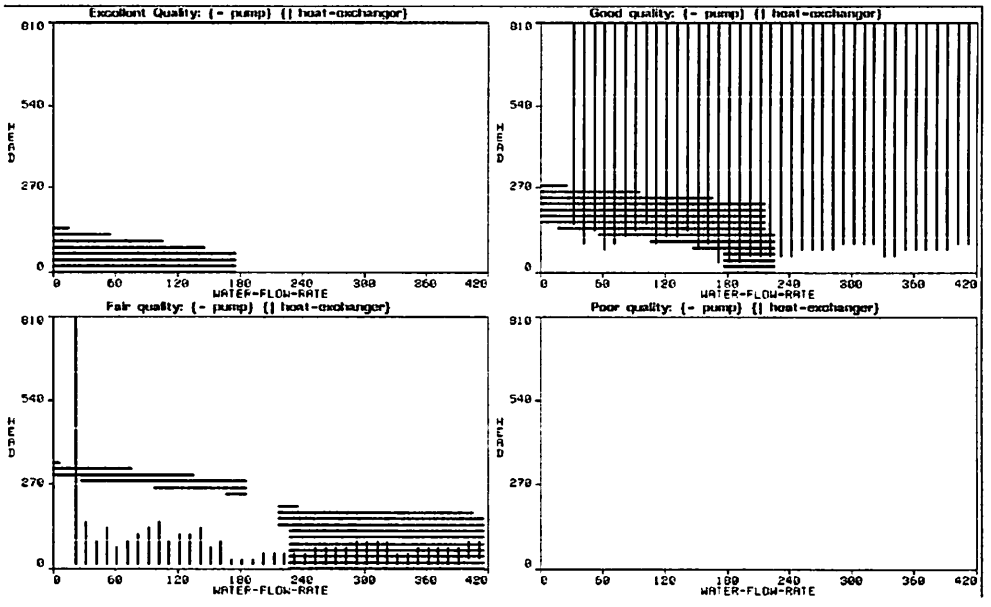

Figure 10: Solution Spaces for Experiment Set 700

from other agents to earlier proposals .

In the limited communication sets, agents will incorporate information from responses to earlier proposals. The term "limited communication" signifies that an agent is able to incorporate communicated information into its local search structure. However, it cannot respond to the sending agent about whether or not received information was successfully integrated. This response capability is currently being implemented.

The results of the random and limited communication search experiments are reported in Table 1.

4.2 Analysis of the Effect of Varying Negotiated Search Strategies

In Experiment Sets 300 and 500, the limited communication negotiation strategies did better than the random search strategies, both in terms of the processing cycles required to produce at least 3 acceptable designs and in terms of the (globally-evaluated) cost of the solutions produced. In Experiment Set 700, the number of processing cycles were the same for both search classes, but the solutions were less costly in the limited communication case. In Experiment Set 900, the type of search strategy used made no difference in either processing cycles or quality of solutions. These results are explained below.

In Experiments Sets 300 and 500, the processing cycles were lower in the limited communication class partially because there were constraints communicated from PUMP AGENT to HEATX AGENT that helped to narrow HEATX AGENT's search space. This resulted in fewer unacceptable solutions being generated by HEATX AGENT. Since each unacceptable solution takes a processing cycle to produce, there were fewer processing cycles used. In Experiment Set 300, this is the only factor that differentiates the number of processing cycles. In Experiment Set 500, there is another factor that will be discussed later. However, 27 of 33 solutions produced through limited communication in Experiment Set 500 were acceptable, and this is primarily due to communicated information. The reason the communication factor is more apparent in Experiment 500 than Experiment 300 is that it takes several cycles of unacceptable solutions being generated for the appropriate constraints to be

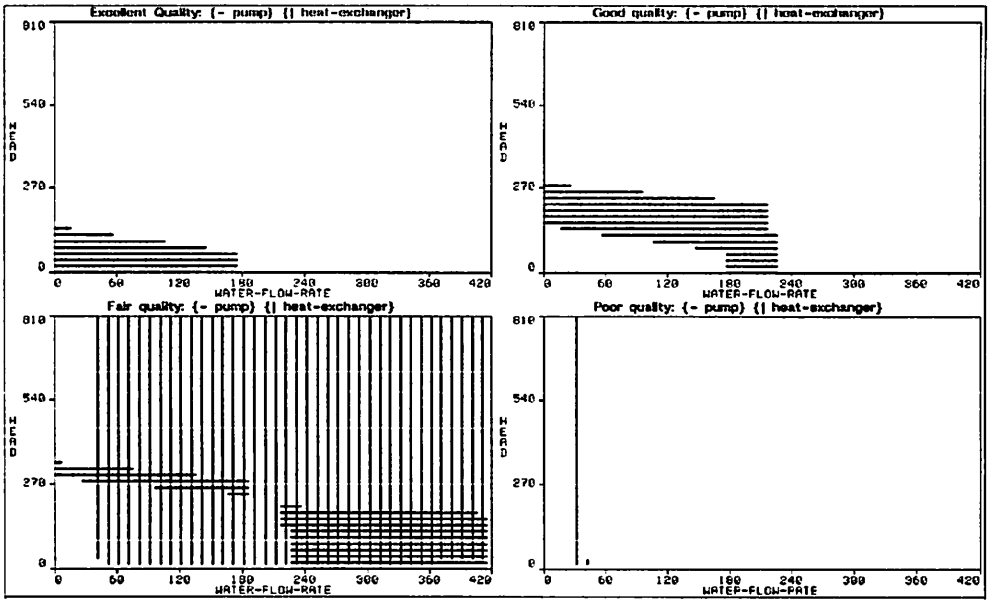

Figure 11: Solution Spaces for Experiment Set 900

Experiment Set 300		
	Random Search	Limited Communication
Processing cycles	8	6
Number of acceptable solutions	3	3
Solution costs: min, max, average	955 1017 996	955 1017 976
Pump final required quality level	excellent	excellent
Heatx final required quality level	excellent	excellent

Experiment Set 500		
	Random Search	Limited Communication
Processing cycles	41	33
Number of acceptable solutions	19	27
Solution costs: min, max, average	1298 1703 1426	1219 1703 1379
Pump final required quality level	fair	fair
Heatx final required quality level	good	good

Experiment Set 700		
	Random Search	Limited Communication
Processing cycles	17	17
Number of acceptable solutions	5	7
Solution costs: min, max, average	1760 1925 1855	1760 1924 1837
Pump final required quality level	good	good
Heatx final required quality level	good	good

Experiment Set 900		
	Random Search	Limited Communication
Processing cycles	6	6
Number of acceptable solutions	3	3
Solution costs: min, max, average	2510 3036 2688	2510 3036 2688
Pump final required quality level	excellent	excellent
Heatx final required quality level	fair	fair

Table 1: Experimental Results: Varying Available Search Strategies

communicated. For example, a conflict may occur over `water-flow-rate` and a pertinent constraint is communicated. However, once this constraint is incorporated, another conflict occurs over head and now pertinent head constraints must be communicated. There is a tradeoff here between sending all constraints and sending only those constraints that are directly relevant to the current conflict with the caveat that further constraint communication may be required later. TEAM agents communicate only directly relevant constraints on the principle that irrelevant information adds significant overhead, both for the sending agent and the receiving agent. Relevant constraints are quickly identified through iterative generation of anchor designs and, once the constraints are known, they are used in the generation of all ensuing proposals. This means that there is some "setup" time at the beginning of a run in which problematic constraints are exchanged. In short runs, like those in Experiment Set 300, the benefits of the limited communication strategy are less pronounced than in longer runs.

Experiment Set 700 resulted in the same number of processing cycles for each search class, but better solutions in the limited communication class. Looking at Figure 10, no *excellent* quality solutions are available to HEATX AGENT at this `required-capacity` value. However, at the *good* level, there are many solutions and a large overlap with the pump solutions. Better solutions were found because the communication of constraints resulted in limiting the local search space of HEATX AGENT, as described for Experiment Sets 300 and 500. The number of processing cycles is the same, however, because many of the matching proposals that were produced by PUMP AGENT had quality level *good*. These proposals were not acceptable at the time they were produced since the starting required quality level of PUMP AGENT was *excellent*. Once PUMP AGENT dropped its quality level to *good* (as explained in Section 4.3), all of the previously developed matching solutions became acceptable. Processing stopped at that point because more than 3 mutually-acceptable designs now existed. It was necessary in both the random search and limited communication sets for the PUMP AGENT quality to drop before solutions were found.

Experiment Set 900 is interesting because almost all of HEATX AGENT's solutions lie in the *fair* class (see Figure 11). That class has a very large overlap with *excellent* pump components. Once HEATX AGENT has dropped its quality level to *fair*, because it couldn't find any more highly-rated solutions, this experiment set degenerates to an "easier" set than Experiment Set 300. Even random search performs well because the overlap between the *excellent* pumps and *fair* heat exchangers covers many possible combinations, making it easy to find mutually-acceptable solutions.

4.3 Dropping the Required Quality Level of an Agent

In Experiment Set 300, the final required quality level of both PUMP AGENT and HEATX AGENT are *excellent*, meaning that neither agent had to drop its required quality level at all. Looking at Figure 4, it is apparent that there is a fairly large overlap of excellent quality solutions between PUMP AGENT and HEATX AGENT. Dropping the required quality level in this situation would be misguided, since the result would be to miss excellent solutions. In contrast, Figure 9 shows that there is no overlap in excellent quality components between HEATX AGENT and PUMP AGENT in Experiment Set 500. At least one agent must drop its required quality level to reach any mutually-acceptable solution.

The representation of the shared solution spaces in Figures 4–11 makes it easy to see the potential overlap of local solutions. This information is invaluable in making control decisions about when it is necessary to relax local requirements. Unfortunately, an agent can't generate its entire solution space during problem solving since this is prohibitively expensive. However, each time an agent performs a local search some information about its local space is developed and

each time it receives information from other agents, some information about the shared space can be inferred. By collecting data from local searches and communicated information, a dynamic characterization of the local and shared solution spaces can be incrementally produced. Salient attributes of this characterization include the size, density, uniformity of both local and shared solution spaces and overlap of local spaces in the shared space. The characterization can be used to guide the invocation of heuristic control policies for negotiated search strategies, resulting in an increasingly appropriate set of control decisions as problem-solving progresses. We have not yet implemented the mechanisms required to use this information as part of the control process and instead use a much less sophisticated decision process. However, this dynamic characterization capability is one of the long-range goals of the project.

Currently, the relaxation control policy is dictated by a parameter at each agent that controls the number of anchor proposals that should be produced at the current quality level before dropping the level. Anchor proposals are those that used as starting points for designs, as opposed to matching proposals, that are produced in response to a specific set of attributes of an existing design. An agent produces an anchor proposal either through *generate-best-proposal* or *generate-bounded-proposal*.

This relaxation control policy implements the general tenant that if no mutually-acceptable solutions can be found after some reasonable amount of search, it may be that there are noncommunicable constraints on some agent's solutions that cannot be fulfilled. If that is the case, further search at that level will not be productive. For example, in Figure 9, HEATX AGENT has some *excellent* solutions and PUMP AGENT has some *excellent* solutions. However, they are not overlapping. HEATX AGENT does not know the boundaries of the area in which its excellent solutions lie due to the nature of its knowledge representation. It cannot determine that there is no overlap, and since PUMP AGENT has no information about HEATX AGENT's search space, PUMP AGENT cannot determine this either. Therefore, there must be some heuristic control policy that curtails non-productive search at a particular quality level.

Quality level can be relaxed either in response to not finding mutually-acceptable solutions or because no local solutions can be found. For example, if HEATX AGENT is executing the negotiated search strategy *generate-best-proposal* and finds that it cannot generate a proposal at the current quality level, it will immediately drop that level in order to produce a proposal (as in Figure 3).

4.3.1 Varying the Control of Relaxation at an Agent

We have run experiments under five different relaxation classes that vary according to the number of anchor proposals allowed to be generated at a particular level by each agent. A relaxation class is represented as (`pump-anchor-designs,heatx-anchor-designs`) where `pump-anchor-designs` represents the number of anchor designs produced by PUMP AGENT before dropping its required quality level and `heatx-anchor-designs` represents the HEATX AGENT designs. The five representative classes are (5,2),(2,5), (5,5), (8,20), and (20,20). Table 2 shows the results from the different relaxation classes. For the analysis of negotiated search strategies in Section 4.2 as summarized in Table 1, we used the results from Class 4: (8,20). This class biases the experiments in favor of HEATX AGENT: HEATX AGENT will go through 20 processing cycles before agreeing to drop its quality level, whereas PUMP AGENT will go through only 8. Under this structure, HEATX AGENT is more likely to end up with high-quality components than PUMP AGENT. In the two-agent experiments presented here, the heat-exchanger component represents a significantly larger proportion of the total cost of a condenser than a pump component, therefore, it is better (from a cost perspective) for PUMP AGENT to drop its required quality level. In experiments with three or more agents, pump components have a larger ripple effect on other components which tends to balance

	Class 1 (2,5)	Class 2 (5,2)	Class 3 (5,5)	Class 4 (8,20)	Class 5 (20,20)
Experiment Set 300: Random Search					
Processing cycles	8	6	8	8	8
Number of acceptable solutions	3	3	3	3	3
Average solution cost	996	1045	996	996	996
Experiment Set 300: Limited Communication Search					
Processing cycles	6	6	6	6	6
Number of acceptable solutions	3	4	3	3	3
Average solution cost	976	1098	976	976	976
Experiment Set 500: Random Search					
Processing cycles	11	8	11	41	41
Number of acceptable solutions	5	3	4	19	19
Average solution cost	1439	1475	1474	1426	1483
Experiment Set 500: Limited Communication Search					
Processing cycles	11	6	11	33	41
Number of acceptable solutions	5	3	4	27	20
Average solution cost	1439	1566	1474	1379	1470
Experiment Set 700: Random Search					
Processing cycles	9	8	11	17	41
Number of acceptable solutions	3	3	6	5	23
Average solution cost	2007	2092	1994	1855	1981
Experiment Set 700: Limited Communication Search					
Processing cycles	8	8	11	17	36
Number of acceptable solutions	3	3	8	7	3
Average solution cost	1817	2092	1945	1837	1857
Experiment Set 900: Random Search					
Processing cycles	6	5	6	6	6
Number of acceptable solutions	3	3	3	3	3
Average solution cost	2688	3112	2688	2688	2688
Experiment Set 900: Limited Communication Search					
Processing cycles	6	5	6	6	6
Number of acceptable solutions	3	3	3	3	3
Average solution cost	2688	3112	2688	2688	2688

Table 2: Experimental Results: Varying Relaxation Classes

the cost of the heat exchanger component. Notice that both Class 1 and Class 4 tend to perform well and that Class 2 performs poorly. These results can be attributed to the bias of those classes toward high-quality components for a particular agent. They indicate that quality biases should be considered when putting together agent sets since they have a large effect. There is some potential for agents to "learn" appropriate biases based on feedback from the global evaluations of different agents' designs.

In the following discussion, the Class 2 results are ignored for two reasons. First, it is believed that the biasing effect is the primary factor in solution cost for Class 2. Secondly, the results cannot be directly compared to any of the other classes since this is the only experiment in which PUMP AGENT is dominant. When comparing results, the most pertinent comparisons are pairwise between Classes 1 and 4 (HEATX AGENT is dominant) and Classes 2 and 5 (no agent is dominant).

In Experiment Sets 300 and 900, there are no differnces in results between relaxation classes (ignoring Class 2) for either random or limited communication search. These two experiment sets can be classified as "easy" problems (once HEATX AGENT's required quality level is adjusted in Experiment Set 900 as explained in Section 4.2). An easy problem is one in which there are many mutually-acceptable solutions and no quality level adjustment that is attributable to lack

of mutual satisfiability is needed by any agent. When mutually-acceptable solutions are readily available, relaxation strategies should not be a factor in problem-solving, and indeed, our results support this.

In Experiment Sets 500 and 700, we see the general trend that requiring more search before dropping quality level results in better solutions. Table 2 is somewhat misleading because solution costs are not directly comparable. For example the average solution cost in Experiment Set 500, random search, is 1474 for Class 3 and 1483 for Class 5. However, the Class 3 average is over 4 acceptable solutions and the Class 5 average is over 19 solutions. A more direct comparison of solution costs is shown in Table 3. Here, it is seen that the Class 3 average over 4 solutions above is 1474, but the Class 5 average over the 4 best solutions (rather than all 19 solutions) is 1334, showing great improvement.

	Class 1 (2,5)	Class 4 (8,20)	Class 3 (5,5)	Class 5 (20,20)
Experiment Set 500: Random Search				
Average solution cost (over 4 best solutions)	1373	1318	1474	1334
Experiment Set 500: Limited Communication Search				
Average solution cost (over 4 best solutions)	1373	1258	1474	1298
Experiment Set 700: Random Search				
Average solution cost (over 3 best solutions)	2007	1809	1994	1789
Experiment Set 700: Limited Communication Search				
Average solution cost (over 3 best solutions)	1817	1803	1803	1857

Table 3: Experimental Results: Direct Comparisons between Relaxation Classes

The anomaly in Experiment Set 700, limited communication, where the Class 5 result is higher than the Class 3 result, occurs because 3 solutions are found in Class 5 before PUMP AGENT drops it required quality level. HEATX AGENT has dropped its level unilaterally because it could not find any *excellent* solutions. Actually several solutions had already been found that had lower costs but were not acceptable to PUMP AGENT. If problem solving had continued until the required pump quality dropped to *good*, the average solution cost over the 3 best designs would have been <= 1786, as calculated by looking at all solutions, rather than just the acceptable solutions.

Although it is reasonable to expect that more search at a given level would result in better solutions (as we see in the results), it is also expected that there is a break-even point where more search will not significantly improve solutions. The tradeoff between processing time and solution quality becomes more critical as search progresses. We believe that it is possible to use a dynamic characterization of local and global search spaces to enable an agent to change its relaxation policies responsively as the problem-solving picture evolves.

5 Conclusions

Negotiated search is a comprehensive model of multi-agent cooperation. This model views conflict resolution as an integral part of agent cooperation and clarifies the role of conflict resolution in terms of how the local and composite search spaces of agents are changed. This perspective on cooperative multi-agent search also provides intuition about how to answer the questions of what type and when conflict resolution strategies should be applied. The applicability of this model has been described and evaluated in the context of a multi-agent system testbed, TEAM, configured for

the design of steam condensers. In TEAM, we have framed the negotiated search model in terms of a centralized framework monitor that holds and integrates the proposals of agents. However, we feel that distributed protocols can be developed to implement the framework monitor thus making our model of agent interaction completely asynchronous and distributed.

Experimental results are presented that show communication among agents about their local search results and the causes of conflict improves both processing time and quality of solutions. We believe that more abstract information about the local and composite search spaces can be valuable in deciding how to control negotiated search. Our next step is to try to extract this information, both statically at system initialization time and dynamically at run time, to guide control decisions. Additionally, we plan on expanding our framework to allow for more dialogue among agents in deciding which agent in a conflict should relax constraints and deciding the number of alternative designs that are actively pursued. The experiments presented here have only dealt with two agents. The framework supports multi-agent cooperation however, and future work will present issues that arise when more agents are involved in the cooperative search.

References

[1] S.E. Conry, K. Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage negotiation in distributed constraint satisfaction. *IEEE Transactions on Systems, Man and Cybernetics—Special Issue on Distributed Artificial Intelligence*, January 1992.

[2] Mark S. Fox. *Constraint-Directed Search: A Case Study of Job Shop Scheduling.* PhD thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1983.

[3] Carl Hewitt. Offices are open systems. *ACM Transactions on Office Information Systems*, 4(3):271–287, July 1986.

[4] Mark Klein. Supporting conflict resolution in cooperative design systems. In *Proceedings of the 10th Workshop on Distributed Artificial Intelligence*, Bandera, Texas, October 1990.

[5] B. Laasri, H. Laasri, and V.R. Lesser. Negotiation and its role in cooperative distributed problem solving. In *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, Bandera, Texas, 1990.

[6] Susan Lander, Victor R. Lesser, and Margaret E. Connell. Conflict resolution strategies for cooperating expert agents. In S.M. Deen, editor, *Cooperating Knowledge Based Systems 1990*, pages 183–198. Springer-Verlag, 1991.

[7] Susan E. Lander, Victor R. Lesser, and Margaret E. Connell. Knowledge-based conflict resolution for cooperation among expert agents. In D. Sriram, R. Logher, and S. Fukuda, editors, *Computer-Aided Cooperative Product Development*, pages 253–268. Springer-Verlag, 1991.

[8] Kenneth L. Meunier. *Iterative Respecification: A Computational Model for Automating Parametric Mechanical System Design.* PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, February 1988.

[9] T. Moehlman and V.R. Lesser. Cooperative planning and decentralized negotiation in multi-fireboss phoenix. In *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, Texas, November 1990.

[10] Arvind Sathi and Mark S. Fox. Constraint-directed negotiation of resource reallocations. In Les Gasser and Michael Huhns, editors, *Distributed Artificial Intelligence, Volume 2*, pages 163–193. Pitman, Morgan Kaufmann Publishers, 1989.

[11] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man and Cybernetics*, Fall 1991.

[12] Katia Sycara. Arguments of persuasion in labour mediation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 294–296, Los Angeles, California, 1985.

[13] Keith J. Werkman. *Multiagent Cooperative Problem-Solving through Negotiation and Sharing of Perspectives*. PhD thesis, Lehigh University, May 1990.