# Knowledge-Based Systems as Cooperating Experts

Susan E. Lander
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 91-28
March 1989

## Abstract

In this report, we introduce the Cooperating Experts Problem Solving (CEPS) paradigm, a problem-solving approach that supports the cooperation of multiple, independent, heterogeneous agents. We discuss the issues that arise in developing a suitable architecture and illustrate the approach through a detailed example. Because CEPS agents are heterogeneous, it is not possible to guarantee consistency of knowledge and methodology across the set of agents. Conflicts inevitably arise among the agents; therefore, conflict resolution is a integral part of the CEPS paradigm. The research described is the first step toward the implementation of a generic framework for developing application programs within the CEPS paradigm.

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

# Contents

# Chapter 1

# Introduction

## 1.1 The Cooperating Experts Paradigm

Many complex problem-solving tasks require diverse expertise to generate comprehensive solutions. Managing diverse expertise implies the ability to work out solutions in the face of conflicting goals, knowledge, and viewpoints [17]. Examples of the integration of expertise through cooperation are seen in human problem-solving tasks such as design, research, business management, and human relations. We present the Cooperating Experts (CE) paradigm as a model for achieving cooperation among sets of knowledge-based computer systems.

Consider a team of human experts who are cooperating in an office task, for instance, choosing a telephone company. The team consists of a manager and an accountant. They have the shared goal of selecting an appropriate system for their office, but each individual would like to insure that her own priorities receive top consideration. Unfortunately, many of the individuals' local goals and priorities are conflicting from a global viewpoint. For example, the accountant would like to try Company 1, a company with excellent long-distance rates, to save money. The manager is concerned about the quality of service and would rather choose Company 2, a company with a known high level of quality.

In this situation, it is very difficult to judge what the "correct" solution is since it depends on the criteria used to decide. In truth, there is no right or wrong answer: the company may need the level of service provided by Company 2, but it may also need to save on costs through Company 1. The problem is further complicated because there is incomplete information available about the quality of service of Company 1—it might prove to be quite adequate for the needs of the office. The two officers must reconcile their differences somehow and reach a decision. There is no guarantee that the decision they reach will be a good one but at least they will have considered all factors. Since solution correctness is an elusive concept, they strive instead for balance. A solution which is acceptable to all experts, though possibly not ideal for any one, is the best that can be hoped for when there are no global criteria for evaluation.

When specialists cooperate, they bring together multiple disciplines or multiple viewpoints on a single problem. Bringing together diverse knowledge is a source of robustness and balance which is extremely important in many real-world situations: a structural engineer and an architect work together to design and build a safe and attractive building, or a pediatrician and a cardiac specialist consult to help an infant with a heart problem. The team can solve problems that are beyond the scope of any of the individual experts and the solutions are generated from a rich and varied body of knowledge, providing the potential for creativity and innovation.

Although diversity is beneficial in some respects, there are also difficulties with handling the conflicts that arise from trying to merge multiple goals, priorities, and evaluation criteria for a common good. How can the conflicts be resolved? Resolution usually occurs through the exchange of information among the participants. How to exchange, what to exchange, when to exchange, and who to exchange it with are questions that are examined in this proposal.

Who will be in charge of the effort to resolve? A mediator can be called in to act as a buffer between the parties, one of the disputants can be given the power to take control of the conflict, or the parties can attempt to come to agreement themselves. Within the CE paradigm, the experts involved in building a solution are considered to be the most qualified agents to resolve conflicts which arise. They know the reasons behind their decisions and are able to anticipate the impact of various revisions. A mediator or a single agent from the team won't have enough detailed knowledge about the problem to be able to make good decisions outside of its own expertise. Therefore, we present a form of expert interaction in which conflicts are resolved through the direct communication of constraints and goals rather than through the use of a mediator or controller.

Given the problem characteristics described above, we propose an overall architecture for sets of CE agents. In Chapter 2, we describe related work which has been done in the areas of cooperation and coordination among knowledge-based systems and human and machine models of negotiation. In Chapter 3, we present ideas which serve as the foundations of the current research. Chapter 4 gives an example of problem-solving within the CE framework.

## 1.2 Strengths and Weaknesses of the Cooperating Experts Paradigm

A set of knowledge-based systems which share expertise has both advantages and disadvantages compared to a single, monolithic system designed to accomplish the same task(s). Because there has been very little practical experience to date with the cooperating experts approach to complex problem-solving, it is impossible to state with certainty what problems will arise. However, we can make informed predictions about both the strengths and weaknesses of the approach and present a rationale for performing research in this area.

The most compelling argument for building small, modular expert agents is the ease of building, maintaining, and debugging small systems as compared to larger, monolithic systems. Given a flexible framework which can support the integration of multiple agents, knowledge-based systems can be built and tested separately. The CE framework supports diversity among its agents: they need not be consistent in form, function, or knowledge. The framework provides a kernel of procedures for communication, conflict resolution, and scheduling and a language for shared information. Once a system has been adequately tested outside of the framework, integration can be tested separately from the problem-solving.

A CE framework can achieve flexibility in its problem-solving by allowing the addition and deletion of systems with minimal effort. Different sets of agents can be combined to handle different problems. Agents can be varied according to expertise, desired inferencing techniques, or based on the amount of computational overhead that can be supported.

CE sets are expected to be robust and show graceful performance degradation in situations where not all agents are operating. Since agents can work independently, the loss of an agent results in a loss of its particular expertise or viewpoint. Depending on the role of the agent in the problem-solving, the impact on the overall solution can be large or small. However, the other agents will continue processing without interruption.

Expected disadvantages of the approach are linked to the complexity of the cooperating experts approach. First, extra computing power must be applied to managing interactions, negotiation of conflicts, and synchronization of the various experts. The result may be that the problem-solving process may be too slow for certain domains. Second, negotiated solutions may be worse in some cases than those that would be generated by a single expert. If a single expert actually has the best perspective on a particular problem, forcing that expert into a cooperative problem-solving situation will result in a poorer quality solution.

Problems with developing applications within the CE paradigm include a lack of guidelines for building coordinated systems, difficulty in coordinating the systems to work together effectively, and the inability to merge overlapping solutions. We address these issues in the proposed framework.

# Chapter 2

# Related Research

## 2.1 Cooperative Distributed Problem-Solving

Cooperative Distributed Problem-Solving (CDPS) is the study of cooperation and coordination among multiple machine agents in distributed problem-solving tasks. Cooperating Experts is a subfield of CDPS with the following distinguishing characteristics:

- Agents are heterogeneous in problem-solving methods and knowledge;
- Long-term knowledge is potentially inconsistent or incompatible among agents;
- Local goals, priorities, and evaluation criteria may be conflicting among agents.
- Agents are largely independent and form liasons dynamically around interacting subproblems.

We briefly describe the most relevant research.

Georgeff [16] discusses a restricted class of problems in which agents are able to cooperate without explicit communication. One requirement of Georgeff's strategy is that each agent must have complete knowledge of other agents' utilities and payoffs. This limitation is too severe for the majority of real-world problems and is certainly too constraining for CE problems. One of the compelling reasons for developing a CE framework is to allow agents to negotiate solutions when they don't have good models of each others' utilities and payoffs.

Hewitt [18] describes a "knowledge processing architecture" which addresses the issues of dealing with conflict and negotiation in large-scale organizations. Knowledge processing has many similarities to the CE paradigm but with a more human quality. Agent interactions are based on very complex and poorly defined motivations and the agents themselves are not well understood. A characteristic of knowledge processing is that the computer organizations described are expected to be compatible with human understanding of how and why they form commitments, cooperate, and negotiate conflicts. The CE framework that we describe is much less committed to modeling human performance.

Hearsay-II [14] provides an early model of multiple specialists working together to solve a single, complex problem; namely, the interpretation of spoken utterances. In Hearsay-II, knowledge is divided among specialists called knowledge sources (KSs). KSs share information through a global structure called a blackboard (BB). Cooperation occurs implicitly through the incremental extension of globally available hypotheses. Conflicts are not resolved explicitly, instead competing hypotheses coexist and vie for processing resources to improve their believability.

The expertise of a traditional blackboard system is represented in KSs which parallel the knowledge-based systems of a CE set. Traditionally, KSs have some or all of the following attributes:

1. they are instantiated in response to a particular pattern on the blackboard;
2. they have no understanding of the problem outside of their own area of expertise;
3. they cannot be suspended during execution or reinstantiated once execution has ended;
4. they keep no history of their actions.

CE agents, on the other hand, are fully functional systems which have the property of persistence throughout the problem-solving process. They have at least a general understanding of the entire problem and can generate at least partial solutions independently. They have private data, knowledge, and histories of their own actions. They can reexamine earlier decisions and change results. They can suspend work on a particular task at any time and resume it when the situation seems more suitable.

An important difference between the Hearsay-II and CE frameworks is that control is distributed in the CE framework. Hearsay-II had a centralized scheduler which made control decisions based on the global situation represented on the blackboard. Each CE agent is responsible for scheduling its own tasks based on both its internal problem-solving state and the information available through the global database.

## 2.2 Human Negotiation and Creative Problem-Solving

Pruitt [21] discusses negotiation and conflict in a general way and details specific negotiation tactics. He defines integrative negotiation and describes a set of negotiation operators. His work comes from a background of social psychology and is grounded in domains such as international relations. Human negotiation models can't be directly applied to machine cooperation. Human motivation is more complex than the state of our current understanding; often, even the disputants in a conflict don't know what they really want. However, the study of human negotiation offers tremendous insight into possible strategies for dealing with conflict and for the beneficial use of conflict as a platform for creativity.

DeBono [10] defines *lateral thinking* as a mechanism for generating innovative solutions in human problem-solving. Lateral thinking is a method in which different approaches to a problem are explicitly sought through strategies which are designed to elicit unusual connections between problem elements. Although DeBono does not speak directly to the issue of negotiation of conflicts, his work offers insight into the type of creative problem-solving that can be used to form innovative solutions.

## 2.3 Computer Models Of Negotiation for Task Allocation

Davis and Smith [9], Conry, Myers, and Lesser [6], and Durfee and Lesser [11] discuss negotiation among multiple agents as a control mechanism for task allocation. Davis and Smith suggest the use of a bidding/response scheme for allocating subtasks to the most suitable available agent. Conry et. al extend this framework to allow for iterative propagation of resource and goal conflicts throughout a node network and provides a mechanism for retracting commitments when a conflict is detected and trying alternate routes. Durfee and Lesser describe a problem-solving framework, *partial global plans*, for coordinating the actions of distributed problem-solvers. PGP's provide a uniform representation for plans and estimates of their costs and effects. This enables sophisticated reasoning about what local actions will best serve the global goals of a set of agents. Negotiation in these projects is used primarily for the coordination of tasks among a set of distributed agents. Coordination schemes suppress direct conflict by minimizing redundancy and therefore minimizing the opportunity for directly competing beliefs to clash.

In task allocation models, the agents are generally assumed to be motivated by a network goal—that of finding the best match of agents, tasks, and resources for a particular problem. An agent will accept work if it has the appropriate resources for the task at hand and is not overburdened by its processing load. In the more general case of conflict resolution among cooperating experts, an agent expects some personal payoff for accepting another agent's proposal. The payoff may actually enrich the agent by providing resources such as money or power or it may simply avoid some worse alternative. The issue is really one of motivation: cooperating agents cooperate to fulfill their own goals while coordinated agents cooperate to fulfill network goals.

## 2.4 Computer Models of Negotiation for Conflict Resolution

Sycara [26, 27] presents a model of negotiation in which the system (PERSUADER) acts as a mediator in union/management labor disputes. PERSUADER attempts to find an equitable solution for a set of conflicting goals within the context of industry standards. PERSUADER's preferred reasoning method is precedent-based reasoning. A case memory is searched for cases which are similar to the current dispute based on a set of salient features. If one is found, the values used in that case are adjusted according to domain heuristics and presented to the disputants as a possible solution. In most cases, the proposals presented by the disputants are ignored by the mediator in developing a compromise solution. This may be reasonable in domains where there is an accessible database of standard cases, such as law or labor relations. It doesn't work in domains where problems require unique solutions or where there isn't any easily accessible compilation of standards. In the framework we are developing, conflict resolution begins with a direct comparison of the conflicting proposals and revisions are based on the differences found.

In PERSUADER, precedent-based reasoning breaks down when no appropriate precedents exist. Preference analysis, a simplified derivative of multi-attribute decision theory, is used to generate a solution based on the disputants' utilities when this happens. Preference analysis relies on the existence of a central mediator with access to the solution evaluation criteria of all agents. This is unrealistic in cooperating expert frameworks since the knowledge represented is so diverse. Even if it were possible to gather all the necessary data, the amount of effort required to do so would not justify the results.

The disputants in Sycara's work are human organizations with complex underlying motivations that are not directly related to the task at hand. For example, a company and union are unlikely to divulge their true utilities to each other or to a mediator. A union may ask for a 20% wage increase, but in fact, would be satisfied with a 7% increase. It is unwilling to state that 7% would be okay because it would then have nothing left to bargain with. In contrast, the parties in the project proposed here are cooperative machine agents who have nothing to lose by stating their "bottom lines" explicitly. Therefore, we are able to use a more straightforward approach to communicating goals and constraints.

A characteristic of this model (and Sathi's below) is that negotiation is the main task performed by the system. The systems perform conflict resolution on disputes that are provided by outside agents. We note that our own work views negotiation as an integral part of a general problem-solving process rather than as a separate task.

Sathi [25, 24] describes a constraint-directed negotiation strategy. The problem domain that Sathi has chosen is the reallocation of sets of resources among consumers in response to changes in resource requirements. He uses a negotiation strategy which automates several formal negotiation operators: logrolling, reconfiguration, and bridging. The problem solver acts as the mediator in a group problem solving process. All negotiation takes place within the group except for the initial evaluations. The mediator has access to local and global information about the situation and about the negotiation participants.

Our work will also build negotiation strategies based upon constraint-directed reasoning [15]. However, Sathi concentrates his work on the negotiation of solutions between agents with differing sets of constraints. There are no coordination issues involved and problem-solving is controlled by a mediator. This is in contrast to our own view of negotiation as a tool for integrating a set of knowledge-based agents.

Adler et. al. [1] describe a set of strategies intended to facilitate cooperative behavior between multiple planning agents. The strategies range from a priori protocols for avoiding conflict situations to arbitration of conflicts to direct agent-to-agent negotiation. Two negotiation protocols are described: *conflict-driven plan merging*, a bottom-up approach to resolving a conflict that has already occurred, and *shared plan development*, a top-down approach to avoiding conflicts as plans are developed and refined. This work is relevant to the CE framework, but is still very preliminary. The strategies are not presented in enough detail to make any direct comparison to the model we are developing.

# Chapter 3

# Research Foundations: Cooperating Experts in Design

## 3.1 Cooperating Experts in Design

Design is the process of constructing an artifact description that satisfies a functional specification and criteria on the form of the artifact, can be realized in its target medium and within the restrictions of the design process, and which meets performance requirements [20]. Some of the characteristics which make the design process particularly amenable to a CE approach are described in [19]:

- the cost, scale, and complexity of many design problems exceed the resources of a single individual or organization;
- the design process may involve diverse resources that in turn are managed by different agents.
- the process of design has been studied extensively thereby providing a basis for the implementation of knowledge-based design systems;
- design problems can be characterized as routine, creative, or innovative with each type of problem requiring different design methodologies.

## 3.2 Knowledge-Based Systems as Cooperating Experts

Each of the agents in a cooperating expert set is a fully functional knowledge-based system. Throughout this report, we use the terms *agent*, *expert*, and *system* interchangeably. We do not believe it will be possible to take "off-the-shelf" knowledge-based systems, make a few minor modifications, and have them work together effectively. Some of the major reasons that this can't be done are:

- there is no shared language for communication;
- most knowledge-based systems cannot tolerate inconsistency and would require extra mechanisms to handle the beliefs of other agents;
- in general, internal knowledge representations do not capture sufficient goal and history information to allow for cooperative solution revision;
- there is no provision for sharing information in a timely manner during problem-solving;
- there are no mechanisms for negotiating the settlement of conflicts;
- the solution space overlap may be random and non-productive.

We have identified several phases of problem-solving that must be captured in a system design in order to allow CE-style processing: solution generation, evaluation, negotiation, and revision. We will attempt to provide guidelines for modularizing systems that will enable cooperating experts to have partial solutions available at appropriate times. By following general rules for system design, a knowledge-based system will be much more likely to be extensible to a cooperating experts component.

In the following sections, we present preliminary ideas about designing systems to be integrated into groups of cooperating experts and the interaction of agents in CE sets.

## 3.3 Knowledge Representation Issues

There are several important issues that must be addressed in the area of knowledge representation when building a system intended to act as a CE. The most obvious is the need for systems to understand each other. This implies that the systems must either use a common language or have translation capabilities for shared entities. Private information need not be accessible or understandable to other systems. For example, an algorithm used to compute a particular result can be represented procedurally in a C program even though the other systems have no understanding of C. However, if there is reason to believe that the other systems will need to understand how that particular result was derived, then some other representation would be more suitable.

In order to enable the negotiation process, certain types of information must be made explicit. This is true regardless of the choice of architecture and inference mechanism. Constraints and goals will have to be communicated and compared and therefore must be represented declaratively in a public language.

There must be a history database that records how goals are expanded, why particular expansions are chosen, what problems are encountered, and what revisions are made. A preference model for constraints and goals is needed to help evaluate choices during solution generation and revision. Because negotiation requires the reexamination of decisions that were made during the original solution generation, it must be decided whether or not some symbolic representation of decision attributes must be retained. A number which represents the rating of a particular path during search does not carry enough information to reevaluate that path when circumstances change [5]. However, it is an open question for our implementation whether to pay the cost of storing a potentially large amount of information in case it is needed or to regenerate the information when the need arises.

## 3.4 Interaction among Experts

Cooperating experts are peers. There is no agent with the power to coerce others to accept a solution. Agents can try to persuade others, pay them off, trade something valuable, or reason with them, but they can't force acceptance of a particular solution. Further, we are considering only sets of experts in which the component systems are not hostile and will not intentionally mislead other agents. The systems must share some high-level goal of cooperation although they can work autonomously as long as their solutions do not overlap.

The systems may have mutually inconsistent or incomplete long-term knowledge, different knowledge representations, different problem-solving architectures, and different biases. It may be impossible for the systems to share information except through formal protocols because of the lack of a common language or understanding of terms. The systems may be working on different problems and overlap in the solution space is incidental or they may be working on the same problem and have different criteria for generating and evaluating solutions. In any case, we are concerned with how to find solutions that are acceptable to all agents given their differing priorities and interests. Because the systems potentially have different internal problem-solving mechanisms, they can't accurately predict the actions or preferences of others–an agent may not even know what other agents exist in the environment and what their interests are. Therefore some formal mechanism must be provided to enable the generation of mutually acceptable agreements. The agents must be able to share solutions and negotiate compromises when conflicts occur.

There are several ways in which agents can interact to generate or revise proposals:

1. one agent can generate a solution independently, all interested agents evaluate and criticize it, and the agent who proposed it then revises it.

2. the agents involved can work together as a group to propose, evaluate, and revise one or more solutions;

3. all agents generate solutions independently, all interested agents evaluate and criticize all the proposed solutions, and all agents independently revise their solutions;
4. a mixed strategy can be used where some work is done independently and some within a group.

Having a central agent generate and revise solutions with other agents evaluating and suggesting repairs is a *critic* approach [23]. This approach is inappropriate for CE processing because it removes the diversity of potential solutions which is one of the primary strengths of Cooperating Experts. A scheme of polling the agents to decide which one was most likely to contribute salient hypotheses given the current state was used in Hearsay-I [22]. A dominant KS would be chosen to present initial hypotheses which were then evaluated by the other KSs. If this did not lead to an acceptable solution, another KS would generate a set of hypotheses for evaluation. There was, therefore, some potential for different experts to present proposals. However the proposals were never directly compared and no attempt was made to understand the reasons for the differences.

Because we are viewing the processes of solution generation and revision as schedulable activities that must compete for processing with a number of other on-going activities, group problem-solving is an expensive activity. Interested agents must be identified, notified of the need for a "meeting", and a time must be scheduled for it. All other processing must stop while the meeting is in progress. This type of problem-solving may prove to be cost-effective at times but isn't required in most situations. Consider a human organization: if all work had to be done in meetings, very little work would get done.

A third means of interaction is for all agents to independently propose solutions, and then have each agent evaluate and criticize all the solutions and revise its own proposed solution. This is again a computationally expensive approach to problem-solving. There is too much redundancy in the revision process. It would guarantee thorough coverage of the solution space (within the limits of expertise represented by the systems in the CE set), but at a high cost. We believe that we can achieve a performance level which is close to this at a much lower cost by limiting the revision of proposals.

We are suggesting a strategy of mixed interaction. Diversity is encouraged in initial proposals by having all agents interested in a particular area submit a potential solution (except under certain conditions which are discussed further in 3.9.7). All participant agents will then evaluate all proposals. When an agent evaluates another's proposal, it logs information about which of its own constraints are violated in the proposal. Some pre-specified number of proposals (possibly one) are then selected for further analysis and revision based on overall ratings. The constraint violations uncovered during the evaluations as well as information about the type and severity of the conflict, and the number of competing proposals is used to decide which negotiation strategy is most suitable for this situation. Agents are free to submit compromise proposals according to their own state of problem-solving. The cycle of evaluation, comparison, and revision is continued until some threshold level of acceptance is reached by one or more proposals.

## 3.5 Conflict Resolution

Conflict resolution is a pertinent topic within the cooperating expert paradigm because conflicts are inevitable and because good solutions require powerful and flexible techniques for handling those conflicts. No agent in the CE set has a global view of problem-solving activity and no agent has the power to make global decisions without the consent of the other agents involved. When there are several proposed solutions, the agents must agree to choose one, cooperatively revise one, or search for a new solution that will be mutually acceptable.

Each agent has a top-level goal of finding a solution that will be acceptable at the global level. However each agent also has local goals and constraints which are potentially in conflict with those of other agents. Because the problems are usually over-constrained by the full set of agents involved, it is unlikely that all the goals can be fulfilled.

Conflict may be either direct or indirect. A direct conflict occurs when two or more agents have beliefs that are explicitly inconsistent. Taking an example from the kitchen design domain described in Section 3.8, we illustrate the two types of conflict. One agent has a constraint that says a sink center should have a 36" counter and another agent has a constraint that specifies a 42" counter. This type of inconsistency exists because of the uncertainty inherent in the domain. There is no absolute rule shared by all experts, the rules are shaped through an agent's experience or through the experience of the human who builds a machine agent.

Indirect conflict happens when agents have constraints that do not directly address each other but which independently constrain a shared object. For example, a kitchen designer says that the sink must be in front of a window and the plumber says the sink must be within 2 feet of the water line to the second floor. This is a problem if the kitchen's only window is 4 feet from the water line. The inconsistency here is not inherent in the knowledge, rather it is due to a particular configuration of objects.

In simple cases of indirect conflict, there may be only two agents involved. However, there may be multiple agents involved and the conflict may not be resolvable by the agents that notice it. For example, a design expert has a constraint that the sink must be in front of a window. The window is far from the waterline to the upstairs, but the plumber accepts the designer's proposal because he can compensate for the distance by using a more expensive piping system. The cost agent does not accept the plumber's proposal however because it's too expensive. The only way for the plumber to reduce costs is to move the sink closer to the waterline, but the designer is the one responsible for that action. All three agents must cooperate to find an acceptable solution.

Some or all of the agents involved must settle for partial goal satisfaction. It may be that if the goals can be slightly relaxed or modified, all goals can be fulfilled. On the other hand, it may be that some goals must be completely relinquished in order to find a solution. The conflict resolution process must incorporate procedures for making decisions about how to focus on appropriate goals and localize revisions as much as possible.

We have identified several stages of conflict resolution that must be addressed in a CE framework:

- detection of conflict;
- solution comparison to isolate inconsistencies between sets of local beliefs;
- evaluation of the type and degree of conflict;
- solution revision based on information collected during evaluation and comparison;
- generation of new proposals when solution revision does not appear to be promising.

We will describe strategies for handling each of these stages in later sections.

Pruitt [21] has identified two types of negotiation which are used by expert human negotiators to seek acceptable solutions. These two approaches can also be applied to networks of knowledge-based systems although it is necessary to take into account the differences in motivation between humans and computer systems. Much human negotiation involves working around personal greed and fears about loss of status, power, and public image. Cooperation is often forced by circumstances rather than by any real desire to produce a mutually satisfactory solution. In our computer networks, we are able to include cooperation as an integral part of the overall problem solving strategy. This, in turn, leads to negotiation protocols which are more straightforward than those used by humans. We describe the two major approaches to negotiation below.

The first approach, compromise negotiation, involves having each party concede some of it own requirements in order to satisfy others' requirements. We consider it to be a method of *fine-tuning* a proposal that is close to being acceptable. Compromise negotiation is suitable for negotiations in which there are a small number of parameters under dispute and well-understood mechanisms for making revisions that move the proposals closer together.

The second approach, integrative negotiation, involves identifying the most important requirements of each party from the initial proposals and using these to form the foundation of a new solution. Secondary requirements are relaxed or relinquished as needed to build the new solution. Integrative negotiation is an exciting alternative to constraint relaxation techniques for two reasons. First, solutions can sometimes be found using integrative negotiation when compromise fails and second, the solutions found using integrative negotiation are potentially more satisfying because they emphasize the fulfillment of primary goals.

## 3.6 Scheduling Cooperative Tasks

We are working with sets of systems that have ongoing work to perform and that cooperate primarily through negotiation of conflicting interests. When a potential conflict is detected, the job of evaluating and resolving that conflict becomes one of the tasks to be performed, but not necessarily the only one. There may be work in progress on several different problems or on different parts of the same problem. Therefore, there is no guarantee that a conflict can be dealt with as soon as it occurs. It must be scheduled like any other job and, because there may be iterative communication involved, it may be necessary to schedule many different subtasks related to the conflict at different times.

In the CE model, control is distributed among the agents. Each agent is responsible for deciding where to focus its attention at any particular time. We present some initial ideas about scheduling conflict resolution tasks in Section 3.9.7.

## 3.7 Design Goals

In order to function as members of a set of cooperating experts the systems must incorporate the following design features:

- solution generation components which record decisions in a history database;
- evaluation procedures for local proposed solutions and those of other agents;
- communication protocols to advertise proposals and identify the participants in multi-agent problem solving;
- negotiation protocols for communicating constraint and goal information and for making mutual decisions;
- explicit constraint representation and constraint attributes which will enable relaxation and analysis during negotiation;

- explicit goal representation and goal attributes to enable decisions about whether partial goal satisfaction is necessary and, if so, how to maximize all agents' satisfaction.

- solution attributes to enable a choice of negotiation strategies.

## 3.8 A Description of the Domain

The application domain we have chosen to demonstrate the CE framework is kitchen design. A well-designed kitchen encompasses different areas of expertise such as space and traffic planning, cost control, and interior design. Some of the areas have no dominant preferred method for deriving solutions. For example, space planning is a search problem that can be approached through different algorithms classified as *sequencing by design unit*, *priority solution method*, or *constraint-directed heuristic search* [13, 2]. The CE paradigm can support multiple agents with similar expertise but different local problem-solving methodologies.

A kitchen design must provide an appropriate set of appliances, storage space, and counter space. These components should be arranged to support the various tasks associated with food preparation and cleanup as conveniently as possible under the constraints imposed by the physical environment. Some of the major goals are:

- to minimize the amount of walking required during the normal course of preparing a meal;

- to ensure that there are no major traffic patterns or other obstacles that intersect the work path;

- to provide enough counters so food preparation can take place without having to continuously clear space;

- to provide cabinet space so items will be at hand when they're needed;

- to customize appliances and other components according to the size of the house and kitchen and the demands of the owner;

- to maximize efficiency and convenience for the most basic kitchen functions.

Unfortunately kitchens never come in ideal sizes and shapes and, even when they do, there are no absolute rules for how to make the most of the space that's available. Usually, the kitchen is divided into work centers that mirror the functionality of a set of components. There are three basic centers: the sink center, range center, and refrigerator center. These can be subdivided in various ways; the sink center is often divided into food-preparation and cleanup centers. There are standard constraints that assist in placing the work centers. One that is widely used is the ***work triangle*** constraint. The work triangle is formed by drawing a line between the midpoints of the stove, refrigerator, and sink. The total length of the line is generally preferred to be greater than 12 feet and less than 23 feet. Even this basic constraint, however, can be manipulated to meet the situation. For example, the "stove" in the work triangle is typically a standard kitchen range. In some situations, however, the range can be moved outside of the triangle and a microwave oven or cooktop can be substituted.

Kitchen design can certainly be accomplished within a single system. Although the knowledge about design is diverse and uncertain and although there are various methods for approaching the problem, the amount of knowledge needed to generate reasonably good designs is manageable. In fact, designs for most spaces can be handled by fairly standard methods using standard components. So why choose this domain? The answer lies in the practical need to be able to build a small prototype that can be implemented and evaluated. Other domains, such as large-scale architectural design or project management, are much more complex and knowledge-intensive, but acquiring the knowledge and building the expert systems to represent it is beyond the scope of a prototype project. In many domains, evaluating a solution is almost as problematic as generating a solution. This makes it difficult to adequately evaluate system performance and limits the number of experiments that can be done. For these reasons, we have chosen a domain in which the knowledge needed is manageable, and in which there are readily available experts to assist with development and evaluation.

## 3.9 Proposed Architecture of the Problem-Solving Framework

Figure 3.1 describes the architecture of an agent within the framework and Figure 3.2 illustrates the overall framework. In the following sections, we discuss some of the features we propose for the framework and bring out some of the questions which need to be addressed.

We will be implementing the CE framework in GBB, a generic blackboard development system that provides a comprehensive package of blackboard tools for application programs [8, 7]. GBB offers flexibility, ease of implementation, and efficient execution of application systems.

We have chosen to use a blackboard system as basis for the CE framework because:

- communication between experts is facilitated by the normal blackboard communication mechanisms, all interaction takes place through the blackboard;
- the blackboard levels and units define a language that can be shared by all component systems;

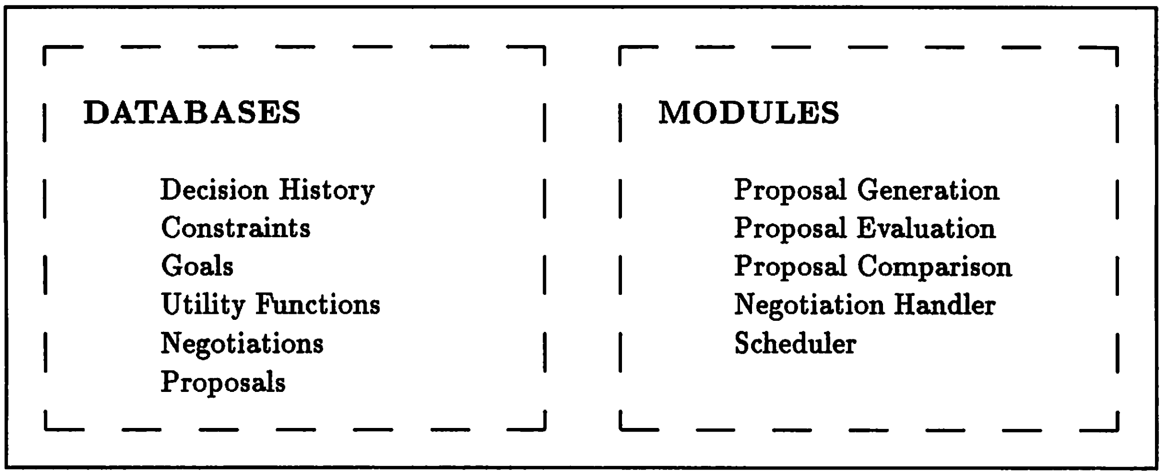

Figure 3.1: Description of an Agent in a CE Framework

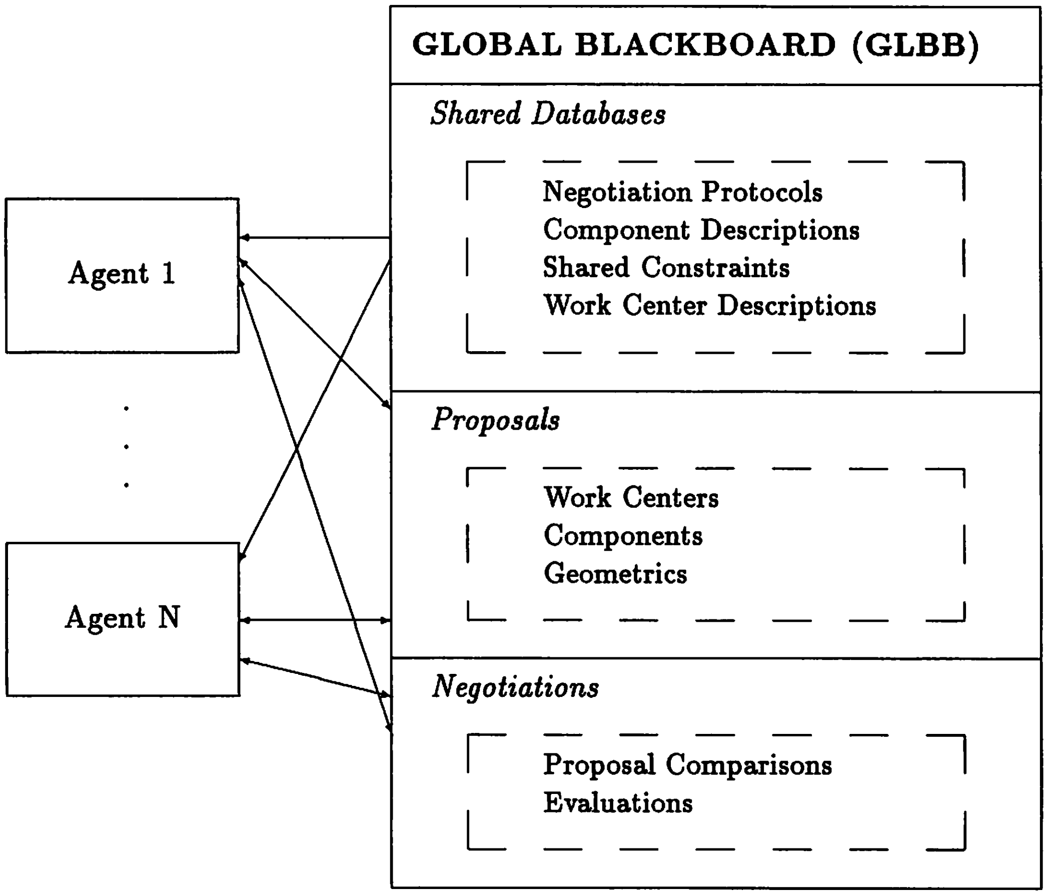

Figure 3.2: The CE Framework

- control flow is opportunistic and allows the necessary autonomy on the part of the component systems;
- the philosophy of blackboard systems as they have evolved has been similar to that of CEs.

The blackboard tools provided by GBB are sufficient to provide the underlying basis for the CE framework and flexible enough to support the required changes to the traditional blackboard model. They also enable efficient and modular implementation of the framework. We expect that using this generic development tool will speed the implementation process considerably.

### 3.9.1 Knowledge representation and storage

Although the ability to display information in a global database is not a requirement for a cooperating expert framework, we have chosen to implement that framework in such a way that there is a public repository, the GLobal BlackBoard (GLBB), available to all agents. This gives us the ability to store "global" information although the information can only be used locally by any of the agents. Alternatively, it would be possible to communicate information directly through point-to-point communication channels.

A shared language is defined by the spaces, levels, and unit types of GLBB. Examples of static global information are a component catalog (refrigerators, ranges, cabinets, etc.) and general constraints which are widely known and applied in kitchen design. This information is placed in the *shared databases* of GLBB. Dynamic global information includes partial and complete solutions at several layers of abstraction. These partial solutions are developed privately by component systems and become globally accessible when placed on GLBB on the *proposal* space. Comparisons of proposals and other aspects of negotiation are stored on a separate space, the *negotiations* space.

Local knowledge is represented in whatever internal language is desired and cannot be accessed by any agent except its owner. If the internal language is not the shared language, translation procedures must be provided for results that are to be communicated to other agents and for other agents' proposals. Constraints and goals must be explicitly represented since they are communicated to other systems as part of the negotiation process. A history of design decisions must be kept to enable the revision process.

### 3.9.2 Constraint-Directed Search

Baykan and Fox [3] describe WRIGHT, a single-agent system which uses an opportunistic constraint-directed search method for space planning, specifically kitchen design.

Since WRIGHT's application domain is the same as the work described here, the two frameworks share much domain knowledge and have similar knowledge representation formats.

WRIGHT uses a uniform constraint representation which includes a set of attributes:

- *importance*–how important is it to satisfy a constraint relative to other constraints?
- *reliance*–are there multiple ways of satisfying a constraint?
- *looseness*–how tightly bounded is the set of possible values that can satisfy a constraint?
- *contention*–how many constraints are attempting to assign values to the same variables?

These attributes define the "texture" of the search space. Decisions about what path to pursue are made using measures of the attributes that define the richness of the space along that path, the certainty that a solution must intersect a path, and the amount of pruning possible.

This work provides guidelines for how to use constraints effectively to limit search in a design domain. We need to do this when conflict forces either the revision of existing solutions or the generation of new solutions in the face of previously unknown constraints.

### 3.9.3 Communication protocols

All communication takes place asynchronously through GLBB in our framework. Agents place proposals on the blackboard at several different levels of abstraction. Using a blackboard allows us to notify relevant agents of a potentially interesting proposal without giving each agent a model of the other systems and their areas of expertise. When an expert enters the framework, it submits a list of areas to the GLBB monitor. The monitor keeps track of incoming proposals and matches them to the experts according to these lists. Whenever a proposal is submitted in an area of interest to a particular agent, it is notified of a potential match. It can then decide whether or not this particular proposal is relevant. An agent who submits a proposal to GLBB has no way of knowing what other agents will see it but can assume that interested parties will be notified.

Another method for deciding who should receive notification of a specific proposal would be for each agent to keep a model of the complete set of experts locally, including their areas of interest. When a proposal is generated in one of those areas, the relevant experts could be notified directly. There are some advantages to this approach since the number of potential responses is known. It is easier to coordinate negotiation tasks when the number of participants is known in advance. The major disadvantage is the difficulty of adding and deleting agents from the framework. Modifying the set of experts would require updating the models of each of the remaining agents.

### 3.9.4 Conflict Detection

Conflicts occur as a result of interacting subproblems. The fact that there is interaction does not become apparent until some expert advertises a result that constrains the work of another. We don't attempt to avoid potential interaction through problem analysis and intervention because subproblem interaction is exactly what gives this paradigm its balance and robustness.

In the CE framework, conflict detection is enabled through the division of the solution space into subspaces. Each agent declares an interest in some set of subspaces and is notified when a proposal is placed in that subspace. The notified system is then responsible for deciding whether or not to respond to that proposal. Each subspace that has more than one system declaring an interest is potentially open to conflict. The subspaces in effect act as a first filtering mechanism. However, the systems can invoke detailed precondition procedures to decide whether or not a particular proposal is truly relevant to their current state. Based on the results of a precondition procedure, an agent may choose to delay action on a proposal or even to ignore it.

### 3.9.5 Solution Evaluation

Solution evaluation takes place whenever a proposal is generated or revised. A system evaluates its own proposals, usually as part of the solution generation process. The evaluation results in a numerical representation of the "goodness" of a solution with respect to the local criteria used to judge. A system will also evaluate the proposals generated by other agents that overlap or directly compete with its own. Each proposal therefore carries a set of numerical ratings, where each rating is assigned by an agent interested in the partial solutions residing in a particular subspace.

Utility theory cannot be applied directly in a meaningful way to the global evaluation of proposals though it can be applied locally by individual agents. To apply a utility function globally would require the existence of some central agent with access to the utilities of all the involved systems and access to the data on which to apply the functions. No such agent exists however due to the nature of CE sets. There is no agent which has knowledge which covers all the evaluation criteria, in fact, the criteria may vary from agent to agent. Further, the agents do not have access to each other's private knowledge or data and don't have good models of their utilities or inferencing capabilities. When we must compare the global quality of solutions, we add the individual ratings and use the sum. This does not measure the real quality of a solution, but it does allow us to order the set of solutions. If each agent uses the same evaluation scale, addition of the ratings gives each agent an equal voice in the overall rating.

When a solution is evaluated by another agent, there is a great deal of information available that can't be captured by a numerical rating. That information will be needed to help focus the revision process appropriately. In the CE framework, we collect pertinent constraints and goals during evaluation and place them in a structure called a *proposal-comparison*. This information is available later during the revision process.

### 3.9.6 Conflict Resolution

When a conflict is detected, the participant systems must find an acceptable solution by exchanging enough information to be able to understand each other's "point of view". We call this *negotiation* and define it as an iterative exchange of constraints, goals, and proposals with the intent of developing a mutually acceptable solution. Because the systems are non-hostile and cooperative, they make no attempt to deceive other agents or hide relevant information.

Pruitt [21] described two major types of negotiation used in human problem-solving. We base our negotiation protocols loosely on the distinctions he identified. The two protocols and the mechanisms for choosing a protocol are described below.

#### Compromise negotiation

Compromise negotiation is a general form of negotiation in which a solution is iteratively revised by sliding a value or set of values along some dimension until a point is found that is mutually acceptable. In some sense, compromise acts to *fine-tune* a proposal that is close to acceptable. A common example of compromise is practiced in real estate transfers: a buyer and seller iteratively suggest prices, thereby sliding a value along the monetary dimension until the two proposals converge. Compromise negotiation has certain requirements that must be true in order for it to be effective:

- there should be a small number of dimensions involved;
- there should be some method for evaluating whether the proposed values are moving toward each other along the dimension;
- the proposals shouldn't be too far apart.

Compromise is expected to be the most efficient form of negotiation if the situation is such that it can be applied. Compromise is based on the revision of an existing proposal and therefore does not require a new cycle of solution generation. Because the systems are willing to share their "bottom line" constraints, the major problem is identifying the salient constraints. Once the necessary information is communicated, an equitable "middle ground" compromise can be found immediately or it can be seen that there is no acceptable compromise. The systems must be able to quickly focus on what the source of contention is and communicate just the necessary information. Too much information will cause confusion and distraction from the major issues while too little information will cause excessive revision cycles.

### Integrative negotiation

Integrative negotiation is a second type of negotiation which is useful for finding solutions in problems that are not appropriate for compromise negotiation or in situations where novel solutions are desirable. The focal point of integrative negotiation is to identify the most important goals of each participant and to find a solution which fulfills the merger of these goals. If the goals are incompatible, it may be necessary to decouple and abandon some of the subgoals. It is also necessary to abandon the current proposals since they are tied to the current sets of goals and distract from the merged goals.

To illustrate, consider a salary negotiation: a company (TheCompany) is trying to induce a prospective employee (EE) to work for them. The job proposed by TheCompany requires extensive travel and long hours but offers an excellent package of salary, benefits, and career opportunities. EE is interested in the position and is highly qualified. She likes to travel and appreciates the benefit package which includes the highest salary that any potential employer has offered. However, she feels strongly that she cannot work any more than an average of 45 hours per week and that the job in its current form would require more than that. Compromise can't be used to resolve the conflict over hours because EE is unwilling to change her position and TheCompany can't reduce the hours required to get the job done. Although they are at a stalemate on the surface, both parties agree that they are very interested in finding a mutually satisfactory arrangement.

TheCompany examines the underlying reasons for the long hours and decides that, given the duties that are assigned to the job, it cannot be done in less time. However they also find that about 30% of the workload involves secretarial and administrative duties that could be handled by someone else. They consider hiring a part-time administrative assistant to reduce the workload, but cannot offer the same salary to EE under the new conditions. They submit a new offer with a lower salary, an assistant, and a 40-hour week. The new salary proposal is less than EE could earn elsewhere but she will still fulfill her primary goals of travel, career advancement, and manageable hours. It is now up to her whether to accept or reject this offer or to try to compromise on salary. At any rate, by focusing on the important issue of hours, TheCompany was able to generate an innovative proposal that satisfies the primary goals of each of the parties.

**Choice of negotiation styles:**

We have said that compromise negotiation is a fine-tuning technique while integrative negotiation offers the opportunity to look for innovative solutions when the current proposals appear to be inadequate. The choice of which type of negotiation to use must be based on on information contained in the evaluations of negotiating agents. We believe that compromise is an easier and more efficient way of handling conflict if the situation is appropriate. We must define the attributes of proposed solutions that will allow us to make decisions about how close two agents are to a mutually acceptable solution. If we can define a set of attributes and an appropriate set of threshold values for those attributes, we will be able to reason about when to try to compromise. Conversely, we will be able to judge when the current situation is hopeless and try for the more difficult, but potentially satisfying, integrative solution. Some of the solution attributes which we feel will assist in making the decision about which proposals to pursue are:

- The number of agents involved in the conflict.
- The number of solution dimensions which are in direct conflict: the fewer the number of dimensions, the more likely that a good compromise can be found.
- The type of solution dimension: for example, it is easier to compromise over a continuously-valued numeric dimension (e.g., money) than over an enumerated or nominal dimension such as cabinet style. The problem with enumerated dimensions is that there may not be a global ordering which can be imposed to indicate how the proposals relate.
- The relationships between competing proposals and other proposals. How many competing proposals are there? How many supporting proposals are there for each of them?
- If the conflicting dimensions are numeric, the distance between proposed values can be considered.
- The number of constraints that have been violated: a large number of violated constraints makes it difficult to find appropriate values for all variables.
- The flexibility of the violated constraints: the more flexible the constraints are, the more likely a compromise can be effected.

Broverman et. al. [4] discuss exception types for planning environments which are analogous to the following conflict types which might occur in a design domain:

- different objects in the proposals;
- missing objects in some proposals;
- the same objects arranged in a different order;
- the same objects in the same arrangements but with different instantiation parameters.

The type of conflict which has occurred can be used as an element of the decision about which style of negotiation is most appropriate. For example if the objects and the arrangements in two proposals are the same, but the x-coordinates of some of the objects is different, it is likely that some compromise of that parameter can be worked out. However, if there are completely different sets of objects in the proposals, the conflict is at a fundamentally different level and fine-tuning is not likely to be an appropriate approach.

### 3.9.7 System coordination

The general area of system cordination and cooperation is beyond the scope of this project. Durfee and Lesser [12] describe a generalized framework (Partial Global Plans) for the coordination of groups of autonomous nodes that plan locally to fulfill global goals. We confine our discussion to the evaluation of overlapping solutions and resolution of conflicts in CE sets. The systems in our implementation work autonomously until an overlap is discovered through the blackboard mechanisms for identifying potential conflicts.

In previous work that has been done on negotiation, conflict resolution occurs as a group problem-solving process. All agents suspend their other activities to focus exclusively on negotiating a settlement. In some circumstances, the importance of quickly finding an acceptable solution outweighs the expense of halting other operations so it is not unreasonable to do so. However, we believe that in the general case there is no reason for all activity to come to a halt because of a conflict. We describe general coordination principles which treat negotiation tasks as schedulable activities within a larger problem-solving context. These tasks are performed asynchronously by the systems involved in a negotiation.

Each system has some criteria for deciding at what level of completion to communicate its results; we call any shared partial solution a *proposal.* In the blackboard approach, each system monitors the database for proposals that overlap its problem areas. When a proposal is made in an area of interest, the system checks to see if it has already generated a competing proposal. If so, it links the two proposals and notifies the proposal originator of its intent to critique the new proposal. It then schedules an evaluation of the proposal and makes sure that this is given a higher priority than any of its own pending tasks which depend on its competing proposal. These dependent tasks may include negotiations with other systems about its proposal. There is no point in resolving a conflict with one system when you know that there is another system interested in the same area. It makes more sense to gather information from all the known interested parties before attempting a resolution.

If a proposal is generated in an area of interest to a system which has not yet generated a competing proposal, that system searches its pending tasks for related generation tasks. It checks to see if it has already started working on a proposal or is planning to start. If so, it links the evaluation and generation tasks. It schedules the evaluation of the triggering proposal before its own generation task. If the triggering proposal is acceptable (or close to acceptable), it may not be necessary to generate a separate proposal.

If there are no current plans to work in the area in question, the evaluation task is scheduled and the system which originated the triggering proposal is notified of that. Although the evaluating system may not wish to generate its own proposal in a particular area, it may have requirements that must be met in order for other proposals to be acceptable. For example, a plumber may not want to submit an interior design for a kitchen but must still be given the opportunity to voice constraints about components which will require water lines.

When a system communicates a partial solution, it doesn't have to wait for critiques to come in from all interested parties since this may take a significant amount of time. In fact, it can't be guaranteed that any particular critique will ever arrive since the evaluation task that produces the critique is just one of a set of scheduled tasks and may never get to the top of the execution queue. Once a proposal has been posted however, systems that are interested in critiquing it will send a notification as soon as possible. Monitoring of interest areas is done at every cycle and notification tasks are given highest priority. However, there may be more than one notification task pending (depending on the number of proposals submitted), so it may take several cycles for all notifications to be sent and received. For the purposes of our implementation, we wait a set number of cycles for notifications and then proceed. If one should arrive very late, scheduling decisions involving that proposal may be reevaluated and if work has already been done, the work itself may have to be reevaluated.

## 3.10 Experimentation

In this section we describe the type of performance we expect from the CE implementation, methods and solution features which can be used to measure performance in terms of both efficiency and quality, and a set of parameters which can be varied to simulate different conditions. In the example presented in Chapter 4, we discuss the impact of the various parameters on a specific design problem.

### 3.10.1 Experimental parameters

We believe that one of the strengths of a CE framework will be its ability to perform gracefully in many different situations. Some parameters that can be adjusted to test the effectiveness of the framework in different situations are:

- Solution space coverage. How well is the solution space covered by the complete set of experts? Could each expert produce a good solution to the problem independently? Can all of the experts together produce a good solution or are they lacking some important expertise? Do the experts work at different levels of abstraction and, if so, are there appropriate links between them?
- Diversity of system knowledge. Long-term and short-term system knowledge can range from completely consistent to completely divergent. Is overlapping knowledge compatible? Does each agent have completely separate input data or are they all working on the same input? Does each agent have some unique knowledge? Does each agent have completely unique knowledge?
- Diversity of inferencing techniques. Are the systems separate instantiations of a single program? Do they share some procedures but not all? Do they have totally different algorithms for computing the same type of result? Do they have different algorithms and compute different types of results?
- Solution overlap. How much interaction occurs? Are the agents able to work completely independently? Should they share every intermediate result?
- Design methodology. How do the problem-solving methods used by the agents compare? Are there methodologies for both routine problems and those that require innovative solutions?

By varying the combinations of experts used to solve a particular problem, we can explore the impacts of the various parameters.

### 3.10.2 Solution cost

We can measure solution cost according to system cycle times and compare it to the cost of solution generation by a single expert. We don't feel that it is necessary to record a speed-up in solution time since we acknowledge that there is a significant amount of overhead involved in cooperation. We are more concerned that the solutions are generally high quality and that the system is robust, particularly in atypical situations that are difficult for a single problem-solver.

### 3.10.3 Solution quality

In designing our experiments, we will use both typical and atypical problem situations and measure the quality of the solutions produced. Quality is, of course, a subjective measurement and there aren't any absolutes by which to evaluate a solution. We will use a validation method to rate solutions: we will have two experts in the field of kitchen design evaluate the solutions based on the criteria they feel is appropriate. What we will be looking for basically is an "average" level of performance (when compared to a human expert) for typical scenarios and, for atypical scenarios, we would like to achieve a performance level that is consistently rated as "good". We aren't expecting to surpass human performance in unexceptional circumstances since solution generation is fairly standardized in these cases. We feel the strength of the cooperating experts approach lies in handling uncommon problems and therefore would expect to find better performance in experiments that have some unusual element.

## 3.11 Summary

The integration of sets of cooperating experts is important to the field of AI at the current time because the types of problems that are being encountered require the application of complex and diverse expertise. There has been considerable work on multi-agent coordination and methods for avoiding conflict among agents. As the agents become more complex however, it becomes a more difficult task to engineer knowledge so as to eliminate sources of conflict. At some point, it becomes an impossible task. The fundamental challenge we address is, therefore, to provide a framework that openly supports multi-agent conflict and its resolution. The following list itemizes a set of goals that must be accomplished in order to realize an effective framework.

- Describe a suitable architectural model to support the interaction and conflict resolution techniques required by the cooperating experts paradigm.
  - What are the characteristics of agent interaction within the paradigm that would help to guide the selection of a model?
- Provide guidelines for designing a system that is expected to be used as an agent in a CE set.
  - How can systems be modularized to support effective information sharing and conflict resolution?
  - What demands are placed on the knowledge representation subsystem by the paradigm? How can goals and constraints be represented to support information sharing and conflict resolution?

  - Which problem-solving methods are most appropriate?

- Define a set of conflict resolution protocols to enable the nearly-autonomous problem-solving inherent in the CE paradigm and describe the situations for which they are applicable.
  - Which agent or agents should be involved in modifying or proposing solutions?
  - What information needs to be shared and when should it be communicated during conflict resolution?
  - What are the salient features of proposals and comparisons of proposals which suggest the use of a particular protocol for resolution?
  - What information-sharing mechanisms are required for simple compromise negotiations?
  - What information-sharing mechanisms are required for integrative negotiations?
  - What information about a developing solution must be saved to enable revision of that solution at a later time?

- Describe the control mechanisms used to coordinate the agents' problem-solving, particularly during conflict resolution.
  - How can conflicts be quickly recognized?
  - How can the agents sychronize their actions when a conflict is recognized to resolve the problem before extending the solution locally?
  - Each agent may have numerous jobs underway at the time a conflict occurs. How can a conflict resolution task be synchronized among agents if each agent cannot devote all of its processing resources to that task?

# Chapter 4

# Example of a Cooperating Experts Problem

The following example is taken from the domain of kitchen design and approximates the problem-solving process of the experts that will be used in our implementation. We present a simplified layout problem for a kitchen design and describe the systems and their interactions. We explain the local processing of the two systems and the negotiation process which results in a mutually acceptable solution.

There are two experts involved in this example: COUNTERS and CONSTRAINTS. They are both fully functional knowledge-based systems and can produce detailed layouts for a given problem individually. CONSTRAINTS produces layouts that balance the requirements for counters, appliances, and other components based on general guidelines. COUNTERS, on the other hand, uses very specific knowledge about counter requirements to produce designs that are exacting about counter details and less so about other components. COUNTERS is used to insure that there is adequate counter space in the kitchen since this is often overlooked and proves to be extremely inconvenient over the life of the kitchen.

Both systems represent several layers of abstraction in their world models, specifically work centers, kitchen components, and geometric components. Geometric components such as lines, rectangles, and circles are used to build kitchen components and enable reasoning about adjacency and containment relationships. Kitchen components include such items as sinks, ranges, counters, doors, and windows. Work centers are aggregates of kitchen components that are connected through functional relationships. For example, a counter is required beside the refrigerator for temporary storage while unpacking groceries and while cooking. The refrigerator and counter can be abstracted to a single work center (the food storage center) during most of the layout process.

COUNTERS and CONSTRAINTS both take a (possibly empty) kitchen layout as input along with the set of work centers that are to be placed. They produce completed layout plans which include the specified units. To understand how the two systems fit into the CE framework described earlier, we look at how they compare over solution space coverage, communication, knowledge representation, and inferencing methods:

- The solution space is completely overlapping. Both systems generate complete kitchen layouts.
- Communication takes place asynchronously through GLBB. Each system can view the blackboard and place units on it.
- System inputs are identical. Both systems take a partial layout that includes at least the physical dimensions of the kitchen and the set of components that are to be included in the finished layout.
- They share a common language, the units defined for GLBB.
- Local goals and constraints are maintained autonomously and are overlapping but not identical. Each system has private knowledge represented in its internal databases. Some goals and constraints are mutually incompatible.
- Each agent is capable of generating a complete solution independently, though the individual solutions aren't likely to be identical to a mutually developed solution.
- CONSTRAINTS uses a constraint-directed search technique to generate solutions that are very conservative and traditional. The knowledge employed is quite general.
- COUNTERS uses a heuristic search in which counters are placed first and other components are then filled into available space. Solutions favor counter space over other components.

Problem-solving begins with the placement of an initial kitchen *layout* and a *proposal* on GLBB. Proposals and layouts are types of *units*; units are the language of GBB. Other unit types include constraints, kitchen-components, geometric-components, and work-centers. The layout specifies a set of physical dimensions which represent the shape and size of the kitchen along with fixed components such as doors or windows. The layout is linked to any proposals which are pending for it. A proposal contains specifications for work-centers which are to be placed in the layout. Work-centers can be specified generically or they can be linked to particular kitchen-components. The initial proposal for this example is shown in Figure 4.1.

An initial proposal may have some specified components or it may have only the types of components that are desired in this kitchen. A complete proposal has all of its components specified. There are three proposed work-centers in this example: food preparation, food storage, and cleanup. Several of the components are already specified for the cleanup center. The design is for a single wall of the kitchen as illustrated in Figure 4.2.

```
{proposal-1
   layout: layout-1
   rating: 0
   work-centers:
        {work-center-1
           center-name: food-preparation
           unspecified-components:  food-preparation-counter
           specified-components: nil
           global-constraints: constraint-1, constraint-2, constraint-3}
        {work-center-2
           center-name: food-storage
           unspecified-components: food-storage-counter refrigerator
           specified-components: nil
           global-constraints: constraint-4, constraint-5, constraint-6}
        {work-center-3
           center-name: cleanup
           unspecified-components: nil
           specified-components: sink-1 cleanup-counter-1
           global-constraints: constraint-4}}
```

Figure 4.1: Proposed Work Centers

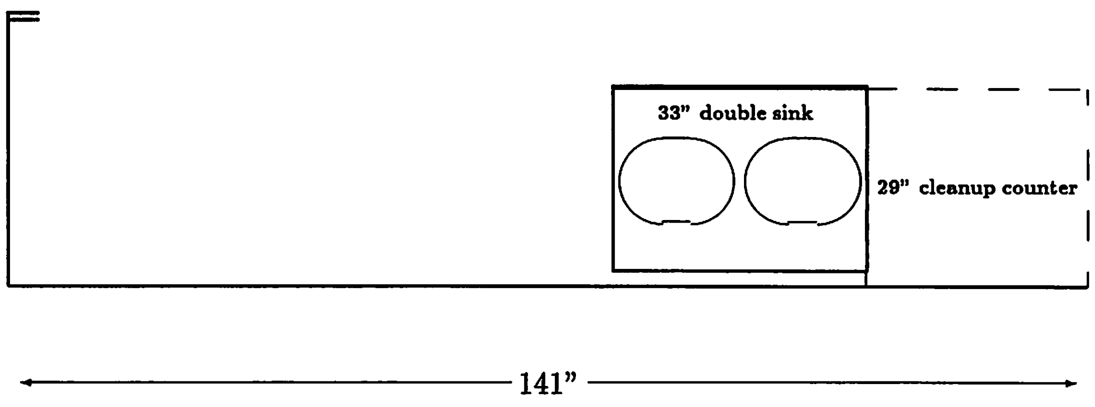

Figure 4.2: The Initial Layout

```
{constraint-1
   constraint-type: dimension-constraint
   component: food-preparation-counter
   contained-in-work-center: work-center-1
   dimension: width
   measurement-unit: inch
   width: 36
     flexibility-measure: 5
     lower-threshold-value: 30
     upper-threshold-value: nil
   fulfilled-goals: goal-1
   importance-measure: 5}
```

Figure 4.3: A Dimension Constraint

When a unit is placed on GLBB, any systems that are interested in the space it is placed on run precondition procedures to decide whether or not it is relevant to their work. If the preconditions judge the unit to be of interest, it is copied to a local database and local processing is scheduled. CONSTRAINTS is interested in any work center or component proposals and COUNTERS is interested in any proposal which includes a counter, so both systems decide to proceed with processing the initial proposal.

There may be global constraints (located in the global constraint database) that are relevant to a proposal. An example constraint (`constraint-1` from work-center-1 in Figure 4.1) is shown in Figure 4.3.

Some of the interesting slots associated with constraints are flexibility measures and importance measures. The variable flexibility measure (VFM) determines the stringency of the value of one of the constraint's variables. A flexibility measure of 10 would mean that the value is totally inflexible and must be satisfied without change. A VFM of 0 would mean that the value is approximate and can be readily relaxed within the specified thresholds. The constraint importance measure (CIM) places a partial ordering on the need to satisfy a particular constraint with respect to the other constraints under consideration in a solution. Constraints with little flexibility of values and/or with high importance are given priority in processing since they serve as focus points for the search process [2].

The `fulfilled-goals` slot is used to link constraints to the goals they satisfy. During negotiation, it is sometimes necessary to look for alternate ways of fulfilling goals. This link provides an explicit mechanism for understanding why a constraint

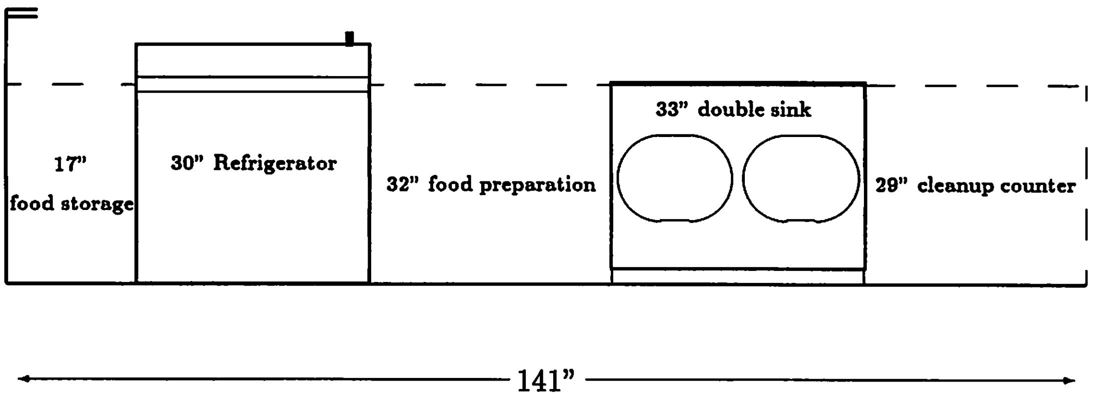

Figure 4.4: CONSTRAINTS' Solution

is being invoked and if there are other possibilities for achieving satisfaction. In this problem, the constraint is linked to a physical-realization goal for the food-preparation-center.

`Constraint-1` defines the ideal width of the food-preparation counter as 36 inches with a lower threshold of 30 inches. `Constraint-2` states that the food-preparation center must be convenient to the cleanup center. The flexibility measure of this constraint is again 5. `Constraint-3` states that the food-preparation center must be located between the refrigerator and sink. `Constraint-4` is shared by the food-storage and cleanup centers and states that the distance between the midpoints of the sink and refrigerator must be between 4 and 7 feet. `Constraint-5` is a dimension constraint which provides for an 18" food-storage counter. `Constraint-6` is also a dimension constraint which defines the ideal width of a refrigerator as 33".

These constraints are located in the global database and are accessible to all systems. Global constraints represent conservative and widely-accepted knowledge. As such, they are often less than accurate for any but the most typical situations and serve primarily as guidelines to follow when more precise information is not available. They are most useful when an agent doesn't have any particular expertise in the area addressed by the constraint. A useful metaphor is to think of them as the knowledge a layman would have about some specialized area.

In this problem, each system generates a solution to the initial layout problem independently, using its local constraints and inferencing techniques. CONSTRAINTS uses a constraint-directed search technique to produce the solution shown in Figure 4.4. COUNTERS produces a solution using a heuristic technique that gives first priority to fulfilling the constraints associated with the counters in the proposal. Once the counters are placed, other components are fit into the remaining space. This solution

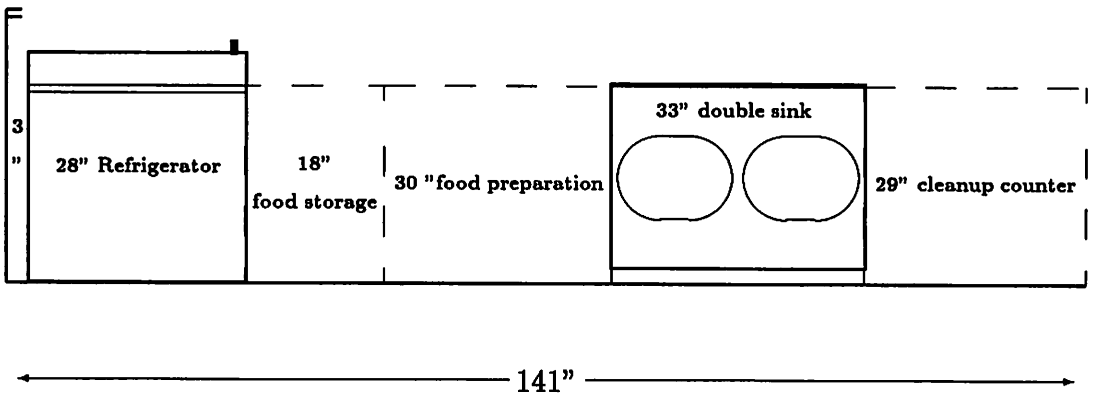

Figure 4.5: COUNTERS' Solution

is shown in Figure 4.5.

It is interesting to note that the solution produced by COUNTERS actually has less counter space than that of CONSTRAINTS. COUNTERS has a constraint which specifies that if the food storage and food preparation counters are adjacent, the ideal total width should be 48". This is 6" less than the sum of the individual ideal total widths. In practice, having the two smaller counters together has a greater utility than having the two larger counters apart. This information is not known by CONSTRAINTS, however, which has less detailed knowledge.

In the COUNTERS solution (Figure 4.5), 3" must be used to move the refrigerator away from the wall so that the door opens easily. Because counters are given priority in placement and size, they are assigned the ideal total width of 48". The refrigerator, on the other hand, is considerably smaller than its 33" ideal.

The CONSTRAINTS solution (Figure 4.4) recognizes that it is impossible to meet the width goals for both counters and appliances and uses a local heuristic to "split the difference". It reduces the total ideal counter width by 5" and the total refrigerator width by 3". (The heuristic takes into account the physical dimensions of available appliances. The 30" refrigerator is the closest available width.)

Each of the proposals has an owner-generated rating associated with it. The two systems employ the same rating scale to enable comparisons during negotiation. For this example, we use a simple numerical scale from 1 to 5 with 1 being poor and 5 excellent. CONSTRAINTS rates its solution at 3 because it was forced to relax the dimension constraints of all of its components considerably. COUNTERS, on the other hand, does fulfill its counter constraints which it considers to be of primary importance. Since the dimension constraints for the other components required extensive relaxation however, it assigns a 4 rating to its proposal.

| Proposal | | Rating | |
|---|---|---|---|
| *Name* | *Created By* | CONSTRAINTS | COUNTERS |
| proposal-2 | CONSTRAINTS | 3 | 2 |
| proposal-3 | COUNTERS | 2 | 4 |

Table 4.1: Proposal Ratings

Solution acceptability has two components: an average rating and the minimum of local ratings. The thresholds for both of these components can be changed to fit the current problem-solving state. An average rating of 3 is considered acceptable for this problem with a minimum local rating of 3. (All agents must rate a proposal at 3 or above for it to be accepted.)

When a new proposal is placed on GLBB, it stimulates activity in other interested systems. CONSTRAINTS and COUNTERS notice each other's proposals and schedule evaluation tasks for them. If a proposal receives high enough evaluations from all systems, it will be accepted without change. In this case, that doesn't happen. CONSTRAINTS gives proposal-3 (the proposal created by COUNTERS) a 2 rating because both the counters and appliances are much smaller than the ideal widths indicated by its constraints. COUNTERS gives proposal-2 (the proposal created by CONSTRAINTS) a 2 rating because the counters are not adjacent and are too small. Table 4.1 shows the evaluation results.

With these evaluations, neither proposal is rated highly enough to be accepted and a decision must be made whether to try a compromise approach or an integrative approach. This decision is based on a comparison of the proposals and attributes of the constraints contained in them. The comparison is prepared by COUNTERS, since it has the most highly rated active proposal, and then placed on GLBB. The comparison can be done by any system since the procedures and knowledge needed are part of the CE kernel. In order to avoid redundant processing, one system should take control of the process at this point. We currently are using a scheme where the owner of the most highly rated proposal is chosen. Another method might be to select the least busy agent, but we currently don't represent this information in a globally accessible way.

Comparisons of the proposals include:

1. Are the objects in the proposals of the same types?

2. Are there the same number of objects of each type?

3. Are the objects arranged in the same order?

4. If similar objects are in conflict, which dimensions are involved?

```
(OR (AND (adjacent food-preparation-center food-storage-center)
         (width food-preparation-center 30)
         (width food-storage-center 18))
    (AND (NOT (adjacent food-preparation-center food-storage-center))
         (width food-preparation-center 36)
         (width food-storage-center 18)))
```

Figure 4.6: COUNTERS' width constraints

5. If similar objects are in conflict, how far apart are the values?

In this case, the objects in the two proposals are the same (although there is a 3" empty space in `proposal-3`, spaces are not considered relevant to comparison of objects). They are arranged in a different order, but can be compared on a one-to-one basis. The only differences in the components themselves occur in the width dimension. This strongly suggests the use of compromise based on the number of conflicting dimensions and the direct comparison of the width values. Constraints related to the width dimension are examined and found to be flexible which also indicates the use of compromise to resolve the conflict. Therefore, a compromise negotiation is initiated between the two systems.

The first task of each system in a compromise negotiation is to isolate which local constraints are violated in competing proposals and communicate this information to the competing system. Since the proposal comparison identified the components and dimensions that were in conflict, this information is used by the individual systems to identify appropriate constraints. CONSTRAINTS notes that the 30" food preparation center and the 28" refrigerator in the COUNTERS' proposal violate its width constraints, `constraint-1` from Figure 4.3 and `constraint-6`. Both of the proposed values are minimally acceptable, but since they affect the rating of the proposal, the constraint is communicated.

COUNTERS notes that the food-preparation and food-storage centers in the CONSTRAINTS' proposal are not adjacent and that the counters are smaller than ideal given that situation. It communicates the knowledge it has about the width values. To simplify presentation, this local constraint is shown in Figure 4 in a generic predicate-logic form.

The constraint communications are posted to the blackboard in the form of proposal *critiques*. Each of the systems looks for critiques posted in reference to its own proposals. COUNTERS picks up the critique posted by CONSTRAINTS and vice versa. Each system must decide how much importance to attach to a constraint received in a critique. For example, the counter width constraints posted by CONSTRAINTS are subsumed by the OR constraint of COUNTERS and therefore can be disregarded by COUNTERS.

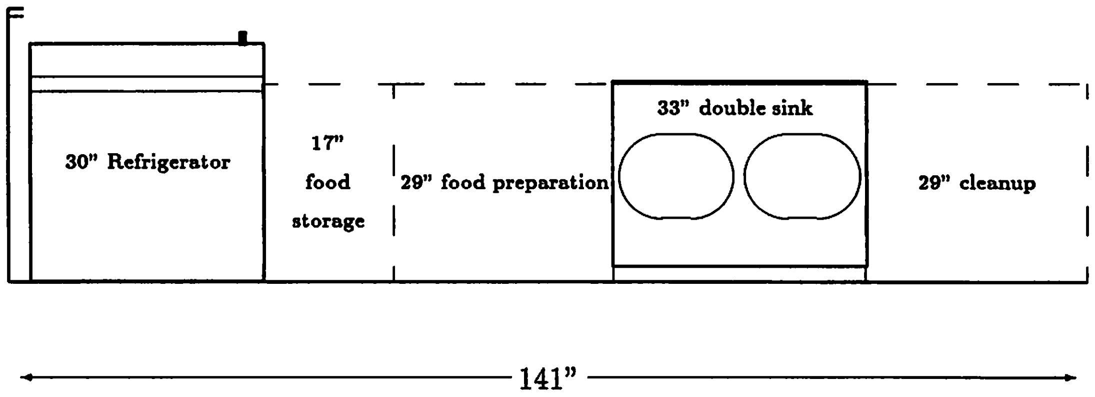

Figure 4.7: CONSTRAINTS Revised Solution

When a system receives a critique of a proposal it owns, it is receiving new information about the world. The information may fill a gap in its own knowledge or directly conflict with something it believes. Even if the information is directly conflicting, it must be given consideration. The system doesn't have to "change its mind" and believe what another system is saying, but if a compromise is going to be reached, it must try to find some middle ground. Because of the comparison of initial proposals that was done before deciding to attempt a compromise negotiation, it is known that the agents have some flexibility about the values they can assign along the conflicting dimensions. Some agents may have more flexibility than others and some agents may have more new information to work with than others. Based on these criteria, each agent must decide whether or not to generate a compromise proposal.

COUNTERS chooses not to submit a new proposal because it did not receive any new information from CONSTRAINTS. CONSTRAINTS, however, did receive new information in the OR constraint it received from COUNTERS. It modifies its proposal in light of that constraint and produces the compromise proposal shown in Figure 4.7.

In the revised solution, CONSTRAINTS relocates the counters adjacent to one another. However, there is still not enough room for all components to take their ideal sizes. COUNTERS' solution to this problem was to make the counters the ideal size and the other components smaller. CONSTRAINTS tries to balance the required cuts among all components. Because the new constraint resulted in 3" less required space for the counters, the cuts are smaller and the new proposal is given a 4 rating. This proposal is posted to GLBB.

The new proposal triggers an evaluation by COUNTERS which results in a 3 rating. Although COUNTERS rates its original proposal higher, the combined ratings of the

| Proposal | | Rating | |
|---|---|---|---|
| *Name* | *Created By* | CONSTRAINTS | COUNTERS |
| proposal-2 | CONSTRAINTS | 3 | 2 |
| proposal-3 | COUNTERS | 2 | 4 |
| proposal-4 | CONSTRAINTS | 4 | 3 |

Table 4.2: Proposal Ratings

new proposal are better as shown in Table 4.2.

## Summary

This section gives a simplified example of the CE approach to problem-solving. There are many issues that aren't addressed in the example because of the complexity and detail of the problem-solving behavior. For example, we do not address coordination issues or integrative negotiation. However, we do illustrate the feasibility of the CE approach. We have created a two-agent CE set and identified salient features of the set with respect to solution space coverage, communication, language, and autonomy. A simple representation of domain hypotheses and constraints was developed, including attributes to enable conflict resolution. A plausible problem is set up and conflicting solutions are proposed by the agents in the CE group. The agents are able to choose a negotiation strategy and identify and exchange sufficient information to revise their own proposals until a mutually acceptable solution is found. This solution satisfies the combined goals of the systems as well as possible since the problem is overconstrained. We will continue to develop the mechanisms described in this report to handle integrative negotiation situations as well.

# Bibliography

[1] Mark R. Adler, Alvah B. Davis, Robert Weihmayer, and Ralph W. Worrest. Conflict-resolution strategies for non-hierarchical distributed agents. In Michael N. Huhns, editor, *Distributed Artificial Intelligence, Volume 2*, Research Notes in Artificial Intelligence. Pitman, 1989.

[2] Can A. Baykan and Mark S. Fox. An investigation of opportunistic constraint satisfaction in space planning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1035–1038, Milan, Italy, August 1987.

[3] Can A. Baykan and Mark S. Fox. Constraint satisfaction techniques for spatial planning. 1989. In progress.

[4] Carol A. Broverman and W. Bruce Croft. Spandex: An approach toward exception handling in an interactive planning system. Technical Report 87-127, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1987.

[5] Paul R. Cohen and Milton R. Grinberg. A framework for heuristic reasoning about uncertainty. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 355–357, Karlsruhe, Federal Republic of Germany, August 1983.

[6] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. Technical Report 86-67, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1986.

[7] Daniel D. Corkill, Kevin Q. Gallagher, and Philip M. Johnson. Achieving flexibility, efficiency, and generality in blackboard architectures. In *Proceedings of the National Conference on Artificial Intelligence*, pages 18–23, Seattle, Washington, July 1987.

[8] Daniel D. Corkill, Kevin Q. Gallagher, and Kelly E. Murray. GBB: A generic blackboard development system. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1008–1014, Philadelphia, Pennsylvania, August 1986.

(Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 503–518, Addison-Wesley, 1988.).

[9] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.

[10] Edward de Bono. *Lateral Thinking for Management, A handbook of creativity.* American Management Association, 1971.

[11] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 875–883, Milan, Italy, August 1987.

[12] Edmund H. Durfee and Victor R. Lesser. Negotiation through partial global planning. In Michael N. Huhns, editor, *Distributed Artificial Intelligence, Volume 2*, Research Notes in Artificial Intelligence. Pitman, 1989.

[13] C.M. Eastman. Automated space planning. *Artificial Intelligence*, 4:41–64, 1973.

[14] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[15] M.S. Fox, B. Allen, and G. Strohm. Job-shop scheduling: an investigation in constraint-directed reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 155–158, Pittsburgh, Pennsylvania, August 1982.

[16] Michael Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 125–129, Washington, D.C., August 1983. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 200–204, Morgan Kaufmann, 1988.).

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

[18] Carl Hewitt. Organizational knowledge processing. January 1989. In progress.

[19] C.T. Kitzmiller and V. Jagannathan. Design in a distributed blackboard framework. In *Workshop on Intelligent CAI*, Cambridge, Massachusetts 02139, October 1987.

[20] J. Mostow. Toward better models of the design process. *AI Magazine*, 6(1):44–57, 1985.

[21] Dean G. Pruitt. *Negotiation Behavior.* Academic Press, 1981.

[22] D.R. Reddy, L.D. Erman, and R.B. Neely. A model and a system for machine recognition of speech. In *IEEE Transactions on Audio and Electroacoustics*, volume AU-21, pages 229–238, 1973.

[23] Earl D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 206–214, Stanford, California, August 1975.

[24] Arvind Sathi. Cooperation through constraint directed negotiation: Study of resource reallocation problems. Technical report, Graduate School of Industrial Administration, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, September 1988.

[25] Arvind Sathi, Thomas E. Morton, and Steven F. Roth. Callisto: An intelligent project management system. *AI Magazine*, 7(5):34–52, Winter 1986.

[26] Ekaterini P. Sycara. Resolving adversarial conflicts: An approach integrating case-based and analytic methods. Technical Report GIT-ICS-87/26, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, Georgia 30332, June 1987.

[27] Katia Sycara. Resolving goal conflicts via negotiation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 245–250, Minneapolis, Minnesota, August 1988.