Organizing cooperative search among heterogeneous expert agents

S. E. Lander and V. R. Lesser
Department of Computer Science
University of Massachusetts
Amherst MA 01003
USA

Abstract

We present *negotiated search*, a paradigm for cooperative search and conflict resolution among loosely-coupled expert agents. The paradigm is realized in TEAM, a framework that provides a flexible environment for agent integration. TEAM enables agents with heterogeneous characteristics and capabilities to work together cooperatively. Experimental results from a design application program implemented in TEAM are presented. These results indicate that system performance is correlated with the organization of the agent set based on the ability of agents to communicate, the interaction capabilities instantiated at each agent, and by the texture of agents' local solution spaces. The experiments show that heterogeneous agents can work together without tightly coordinated organization. However, they also demonstrate that some agent organizations have more potential for effective cooperation than others. We analyze agent characteristics that affect this potential and discuss the use of *negotiated-search strategies* that take advantage of the particular strengths of the agents in an agent set. Both agent characteristics and group dynamics have far-reaching implications for the development of multi-agent systems and for the design of agents that are intended to work within agent sets.

1 Introduction

Negotiated search is a paradigm for the cooperative development of mutually-acceptable solutions by a set of heterogeneous expert agents. In negotiated search, search and conflict resolution mechanisms are tightly interwoven, and conflict resolution is viewed as an integral part of problem-solving. Agents are not hostile and will not intentionally mislead or otherwise

This research was supported by DARPA under Contract #N00014-89-J-1877 and by a University Research Initiative Grant, Contract #N00014-86-K-0764. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

try to sabotage each others' reasoning. Agents are cooperative, meaning that an agent is willing to contribute both knowledge and solutions to other agents as appropriate and to accept solutions that are not locally optimal in order to find a mutually-acceptable solution.

An agent is designed to work outside of any statically defined agent set. Each agent represents an area of expertise that can be incorporated into dynamically formed sets of agents as required to work on problems that require that expertise. Given this model of independent agents, it is not possible to anticipate and engineer out all potential conflicts at agent-development time since it is not known what knowledge will be contained in the complete system (Hewitt 1986). Instead of dealing with conflict through knowledge engineering, each agent must address it explicitly as it occurs. The presence of conflict is not related to the willingness of the agents to cooperate. Conflict is inherent in the agent set due to inconsistent knowledge among agents, incomplete knowledge and/or incorrect assumptions, different problem-solving techniques, and different criteria for evaluating solutions. In the domain presented here, conflict is detected either through the explicit communication of local solution requirements or through the direct evaluation of a proposed solution.

There are two basic ways to characterize approaches to conflict resolution in negotiated search: *extended search* and *relaxation*. The first, *extended search*, is applied by an agent when it recognizes a conflict with another agent in an existing solution. The agent sidesteps the conflict by extending its local search until a solution is found that does not conflict. Extended search methods are used when an agent believes that a better solution can be developed if it continues to examine its local solution space for additional solutions. Some general methods that can be implemented as local operators to facilitate extended search in specific situations include: integrating external constraints to narrow the solution space; finding alternate goal expansions; or using case-based reasoning to find a ball-park solution and then refining that solution.

The second negotiated search method, *relaxation*, is applied when an agent must relax some requirement on a solution in order to expand its local solution space. Relaxation may lead directly to a solution. If a history of previously generated, but unacceptable, solutions is kept, one of these solutions may become immediately acceptable. If no history is kept, extended search can be applied in the expanded solution space with a better chance of finding a solution. Relaxation methods are utilized when it seems likely that the solution space is overconstrained or when the expense of further local search is unjustified. Some general methods that can be implemented as relaxation operators include: relaxing or relinquishing constraints, relaxing or relinquishing goals, manipulating constraints (e.g., unlinking, bridging (Pruitt 1981)), or manipulating evaluation criteria (Sycara 1985).

In negotiated search, two interwoven processes occur: first, *local search* by a single agent for an optimal solution to a subproblem under its local requirements for solutions; and second, *composition*[1] of local subproblem solutions into an overall solution in the composite space created from the intersection of the local solution spaces of the agents. Local search is guided by the domain expertise of the searching agent. Composition relies on the group problem-solving skills of each agent: communication, coordination, and assimilation of externally provided information. Effective integration of these two processes is captured in the TEAM framework described below.

[1]Sathi similarly uses the term composition as the name of a specific negotiation operator that combines local information (Sathi and Fox 1989).

Negotiated-search operators are specific methods for doing extended search or relaxation within an agent. The decision to apply a particular operator to a specific problem-solving situation is made locally by each agent. It is possible to get acceptable behavior without coordinated efforts by the agents. However, more coherent behavior is likely to result through the analysis of the skills of the agents and the desired dynamics of the group. A *negotiated-search strategy* guides the definition of operators and coordinates the scheduling of a sequence of operators across agents, eliciting a particular style of coherent inter-agent behavior within an agent set.

Section 2 introduces the steam condenser design domain and Section 3 describes the TEAM framework. In Section 4, we relate our work to that of other researchers. Section 5 gives a brief description of the negotiated search paradigm and its realization in the TEAM framework. A more detailed description is available in (Lander and Lesser 1991). In the experiments described in Section 6, we vary the sets of negotiated-search operators instantiated at each of the agents in a set and discuss the underlying characteristics of the agents and the dynamics of the agent set that lead to the observed behavior. In Section 7, we describe a negotiated-search strategy, *compromise*, and discuss the requirements for the use of a particular strategy with respect to agent characteristics and group dynamics that exist in an agent set.

2 The Parametric Design Domain

The initial domain for the TEAM framework is parametric design of steam condensers under a set of user-defined specifications. In parametric design, the general form of the artifact being designed is known, but the designer must find values for a set of variable parameters. Much of the application knowledge in the steam condenser domain was originally developed by Meunier for use in an *iterative respecification* system (Meunier 1988). A steam condenser comprises a set of components: a pump, heat exchanger, motor, platform, shaft and v-belt as shown in Figure 1. Each TEAM agent produces a *proposal*, a design for a single component

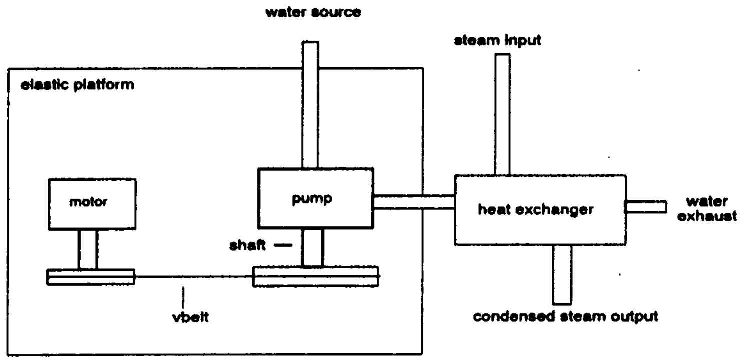

Figure 1: *A Steam Condenser*

of a condenser, or a *critique*, a local evaluation of a partial or complete design. Both proposals and critiques may include information about local solution requirements that will help to guide the local search of other agents. Each steam condenser component is designed by a separate agent, for example, PUMP AGENT produces pump proposals. The components are independent except for shared parameters that represent the interface points of the design.

3 The TEAM Framework

The TEAM framework supports loosely-coupled heterogeneous agents engaged in problem-solving through negotiated search. The architecture of the TEAM framework is shown in Figure 2. Agents are distinct entities that communicate through the shared memory. A set of

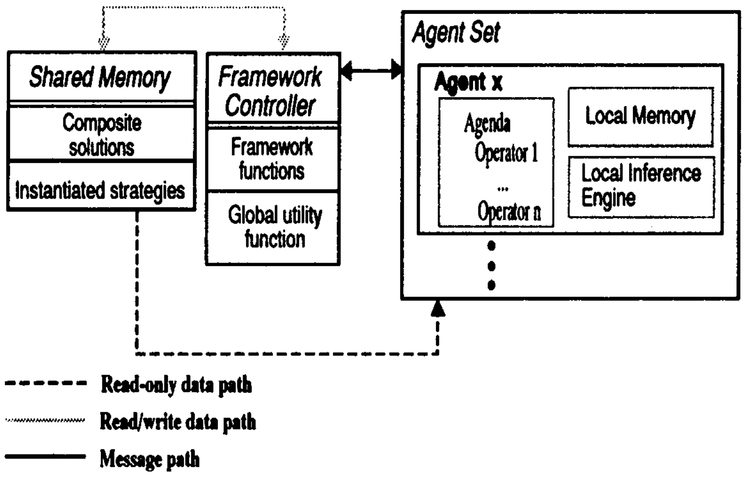

Figure 2: *Architecture of the TEAM Framework*

TEAM functions incorporate proposals into designs and do bookkeeping tasks on the designs, for example, totalling the cost of a design when a new component is added. These functions are domain-independent. For example, to create a new design from a proposal submitted by an agent, a TEAM function will build an empty design object and copy values from the proposal to the design. During processing, there are two distinct phases: 1) an agent cycle; and 2) a framework cycle. During the agent cycle, each agent is invoked sequentially. The invoked agent uses the information in shared memory to choose applicable operators and add them to its local agenda. It then invokes its highest-priority operator and returns its result. After all agents have executed, TEAM functions are invoked to update the shared memory in accordance with the returned results.

4 Related Research

Negotiated search has roots in blackboard problem-solving, constraint-directed search, and in negotiation, both human and computational. From blackboard problem-solving, we take ideas about opportunism, competing solutions, flexibility, and incremental extension of partial solutions.(Erman *et al.* 1980). Although the blackboard literature provides a great deal of insight into coordinated behavior (Lesser and Corkill 1983), TEAM is not itself a blackboard system. A discussion of the differences can be found in (Lander 1993).

From constraint-directed reasoning (Fox *et al.* 1982), we borrow (and simplify) a constraint representation to define the problem space and to generate and evaluate solutions within the space. Sathi and Sycara (Sathi and Fox 1989; Sycara *et al.* 1991) address constraint reasoning issues in their work on constraint-directed negotiation. The agents in those projects share an underlying integrated problem-solving methodology and differ only in the specifics of their constraints. The agents' interactions are centrally synchronized. In contrast, we are looking at problems where agents possess different knowledge, problem-solving skills, and communication capabilities, and in addition, they take responsibility for scheduling their own activities.

From negotiation literature comes an understanding of the underlying mechanisms of conflict resolution and the development and application of specific negotiated-search operators. Klein (Klein 1990) has focused on the development of a taxonomy of conflict types, while we are looking at determining appropriate operaters given particular agent and group characteristics. Werkman (Werkman 1990) has developed a multi-agent multi-criteria system that relies on an integrated knowledge representation and mediation framework to discriminate among alternative solutions. Because TEAM agents solve interactive subproblems rather than redundant ones as in Werkman's system, a centralized mediator doesn't have the appropriate domain expertise to make intelligent decisions. Sycara describes a negotiation system, PERSUADER, for a non-cooperative labor/management domain (Sycara 1985) that uses both extended search and relaxation methods through a mediator. In this case, one of the functions of the mediator is to act as a buffer between hostile agents, helping to maintain privacy of information and an objective viewpoint. Pruitt (Pruitt 1981) and Fisher (Fisher and Ury 1981) discuss human negotiation. In human negotiation situations, much of the effort expended in reaching settlement is directed toward minimizing the psychological discomfort of concessions. Despite this, the work offers valuable insights into specific methods for achieving mutual satisfaction.

5 Negotiated Search

In this section, we give a brief overview of the realization of the negotiated-search paradigm in TEAM. A more complete description of the system can be found in (Lander 1993). We begin with a description of how local problem-solving efforts of the agents are coordinated through the overall view of problem solving maintained in the shared memory. Next, the mechanisms used in local problem-solving at the agent level are presented.

5.1 Coordination of Local Search Efforts

Agents communicate and coordinate their efforts through a high-level view of problem-solving, maintained in a shared memory, and accessible to all agents. Each agent works on some subproblem(s) and produces *proposals* that represent subproblem solutions. Each proposal is assigned a rating by the agent that generated it. We use a rating scheme with values of *infeasible*, *poor*, *fair*, *good*, and *excellent*. The calculation of a local rating is domain-dependent and is defined independently for each agent. Each agent also assigns an acceptability value to a proposal: *acceptable* or *unacceptable*. Acceptability is calculated by an agent based on its current level of solution requirements. That is, a proposal rated as fair by an agent may be initially unacceptable, but as solution requirements are relaxed, it may become acceptable even though its rating hasn't changed.

A solution in shared memory is called a *design*. Each design is initiated by the generation of an agent proposal, called the *seed proposal* for that design. A seed proposal specifies a partial set of parameter values for a design. At any given point in problem-solving, there may be multiple active designs. Agents are free to generate seed proposals at any time, depending on their local scheduling algorithms. At least one agent in an agent set must generate a seed proposal in response to a user-specified problem definition in order to initiate problem-solving. However, it is generally the case that multiple agents generate seed proposals at system start-up time and at other times during problem-solving. Each seed proposal represents a different starting point (in the composite space) for a design; normally, it will take multiple attempts to find one that is acceptable.

Once a design has been initiated by the generation of a seed proposal, it is placed in shared memory and other agents can respond to it. An agent responds to a design by generating a proposal to extend that design or by generating a critique of the design. Conflicts are detected by the responding agent: an agent may attempt to extend an incomplete design and be unable produce an acceptable proposal given the parameter values that have already been specified; or an agent may attempt to criticize a design and find a local constraint violation. The responding agent returns the proposal or critique that was generated regardless of its acceptability along with any information that can be offered to explain detected conflicts.

The acceptability of a design is a function of two components: the set of acceptability ratings from each agent and, if desired, a global utility value. Agent acceptability ratings reflect individual agents' criteria for evaluating solutions. For example, one agent might consider safety a high-priority criterion while another would consider the life expectancy of a design to to be more important. Each of these criteria are important and should be considered in a final solution. However, the information needed to calculate a value for a particular criterion is often embedded in the local knowledge of an agent and cannot be represented outside of the expert environment. Global utility, on the other hand, provides a way to rank alternative designs on aspects of the overall solution that do not require domain expertise to evaluate (cost, for example). In TEAM the system developer can specify a function to apply to each complete solution that calculates a global utility value, as well as a threshold value for the utility. Thus, we take a two-level view to evaluating designs. The goal of negotiated search is to find solutions that optimize global utility value while maximizing agent acceptability.

If the design is complete (all components are represented) and acceptable, the system will add it to the set of completed designs and will stop if it has met a user-defined quota on the number of alternative designs required. If the design is not complete, it stays active while waiting for component proposals from other agents. Agents locally schedule their own activities and may respond at different times. A local scheduling algorithm might attempt to respond to the most complete acceptable design on each processing cycle. However, each agent is responsible for managing its own local agenda and may have other priorities: no global restriction is placed on local scheduling algorithms.

Unacceptable designs are saved along with information about why they were unacceptable. A design that is unacceptable because of constraint violation(s) can be considered a potential compromise. If the constraint(s) involved are relaxed at some future point, this compromise will become acceptable and the design will be reactivated. If a feasibility constraint has been violated in the design, the design is marked as infeasible and will not be considered as a possible compromise.

5.2 Local Search

Although there are many operators that can potentially be applied to conflict situations, they all fall under the general umbrella of either extended search or relaxation. To negotiate in a complex domain, an agent needs to have operators available from each class. The instantiation of an operator at a particular agent is relatively unrestricted: it must react to a particular input(s) with a required functionality and produce a specific output. However, the actual methodology used for achieving the desired functionality is not constrained. For our experiments, we have defined three representative operators. Every operator has a specific set of situational requirements that impact its effectiveness for a given agent. We will discuss using solution space and agent characteristics to choose appropriate operators for a particular agent further during the analysis of experimental results.

The three defined operators are *generate-proposal*, *respond-to-design*, and *relax-solution-requirement*. The first two are extended search operators, the third is a relaxation operator. Each of these operators has a defined functionality that is implemented locally by each agent in a style consistent with the agent's architecture, knowledge representation, and inference engine.

Search Operators: *Generate-proposal* is an extended search operator that an agent implements to generate a seed proposal that satisfies its locally-known set of constraints. The input required for *generate-proposal* is a user-defined problem specification. There are two possible outputs from the *generate-proposal* operator: either a proposal or a message that no proposal can be found. If an acceptable proposal cannot immediately be generated by an agent, it may unilaterally decide to relax some requirement and/or extend its search through a locally-defined mechanism. If an agent does not instantiate the *generate-proposal* operator, it will not create seed proposals for designs. However, it can still generate proposals in response to designs initiated by other agents as described below.

Respond-to-design is the second extended search operator defined. This operator takes as input an externally-initiated design (a composition of proposals from other agents). It serves to both evaluate the design and to extend it through the generation of a local proposal that matches (fits into) it. The local proposal is generated under the external specifications of the input design and is often not acceptable to its creator: in order to extend the input design, the agent must violate some local solution requirement. It outputs either a matching proposal, with information about local solution-requirement violations when necessary, or a message that the design is infeasible with respect to local constraints. The initiating agent for the design will usually react to an unacceptable evaluation by generating a new seed proposal. When an unacceptable proposal is returned with constraint violations, the violated constraints can be assimilated by other agents to initiate new designs that will avoid the same conflict(s).

Relaxation Operators: We define one relaxation operator, *relax-solution-requirement*. This operator can be invoked unilaterally when an agent can't find a solution under the current problem specification. It can also be invoked due to nonproductive iterative negotiation efforts: if an agent has generated multiple seed proposals and/or has responded to multiple designs without finding a suitable fit, there may not be a fit.

An agent can also relax a solution requirement in response to information received from another agent that explicitly conflicts with some local requirement. For example, if Agent A has the constraint $x \leq 100$ and Agent B has the constraint $x > 100$, one of the agents will have to relax its constraint. Constraints include information about flexibility (how important it is to relax or not relax) that an agent uses to decide whether to relax its own constraint or not. When a solution requirement is relaxed, previously developed unacceptable solutions are reevaluated to see if they are now acceptable in the expanded solution space induced by the relaxation.

6 Experiments

We present a set of experiments that investigate the effect of varying the operators available to a set of agents. In order to keep the discussion focused, the experiments reported here were kept simple. A more complete list of TEAM experiments to date appears in (Lander and Lesser 1991).

Each of the experiments involve three agents: one that produces pump proposals, PA, one that produces heat exchanger proposals, HEA, and one that produces motor proposals, MA. In each experiment, the agents have a different pattern of negotiated-search operator instantiation (see Table 1). These differences affect each agent's ability to generate seed proposals and/or extend designs initiated by other agents. We analyze the experimental results for each configuration and discuss the underlying characteristics that influence the results: namely, the effectiveness of a particular operator given the characteristics of the solution space and the effectiveness of the overall strategy embodied in the set of operators chosen for each agent.

Operator	Experiment Set		
	1	2	3
generate-proposal	PA HEA	PA	HEA
respond-to-design	PA HEA MA	HEA MA	PA MA
relax-solution-requirment	PA HEA MA	PA HEA MA	PA HEA MA

Table 1: *Agent/Operator Configurations for Experiment Sets*

Table 2 summarizes the results from the experiments described in Table 1. Average cost is the average of the global utility values calculated for each of the best three complete acceptable designs. The minimum cost is the global utility value calculated for the most highly-rated complete acceptable design. A processing cycle comprises the sequential invocation of each agent in the agent set and the TEAM functions required to process all returned values. The relaxation threshold is a user-specified value that controls how quickly an agent is able to relax a solution requirement. For example, if the relaxation threshold is 10, an agent must go through at least 10 search cycles before relaxing a requirement (unless an explicit constraint violation is found or no new solution can be generated under the current requirements). This parameter controls how much search should be done before assuming that no solutions can be found without relaxation.

Set #	Average Cost	Min Cost	Processing Cycles
1			
Relaxation threshold 10	1644	1577	11
Relaxation threshold 15	1600	1577	15
2			
Relaxation threshold 10	1600	1577	10
Relaxation threshold 15	1600	1577	15
3			
Relaxation threshold 10	1991	1767	30
Relaxation threshold 15	1907	1765	46

Table 2: *Experimental Results*

To simplify the discussion of the results, we first examine a simple, generic, two-agent example. The agents' solution spaces over the shared parameters x and y are shown in Figure 3. The agents in the system communicate only declarative constraints and not relationships

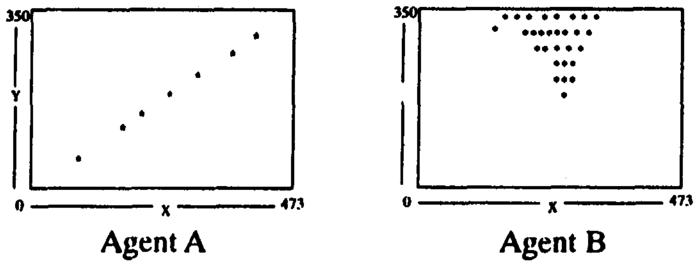

Figure 3: *Solution Spaces of Agents A and B over x and y*

that are procedurally described. For example, let Agent A have a functional relationship between x and y of the form: {`PROCEDURE Calculate-X-From-Y (Y) Return X := Y+1`}. Agent A cannot inform Agent B that x will always equal $y + 1$ because the information is not represented as a constraint. Of course, this is a very simplistic example of a functional relationship. In general, functional relationships are more complex, e.g, in the steam condenser domain, the relationship between parameters `water-flow-rate` and `head` is a function involving domain-specific calculations over locally-defined variables such as water input temperature, viscosity, and velocity.

Agent A can express boundary constraints, e.g., $0 \leq x \leq 473$ from Figure 3, and, if boundary constraints are available from Agent B, the solution space overlap (the composite space) can be generated. However, Agent A cannot further constrain the points within its boundaries to help other agents in their local searches since the required information is represented functionally. When it evaluates a design initiated by another agent that doesn't fulfill the required parameter relationship, it can only reply by indicating that the proposal is unacceptable without supplying any further information.

If A is the only agent in the system with this type of solution space, it should be the controlling agent in the system. That is, A should generate proposals and other agents should send back constraining information that would help to guide A's local search. In the context of the extended search operators defined earlier, A should instantiate *generate-proposal* and not *extend-design*. It is highly constrained by its functional relationships among parameters and can generate designs with the appropriate relationships, but cannot communicate that knowledge to assist other agents in their local searches. Other agents should instantiate only *extend-design* since they are unlikely to generate seed proposals with the correct parameter relationships and are better suited to providing guiding information to Agent A. This situation is exactly that which occurs in the experiments above. PA has a parameter relationship between water-flow-rate and head that severely constrains the set of acceptable solutions. We see in Table 2 that the best results are obtained in Experiment Set 2 where PA is the controlling agent (the only agent generating seed proposals).

In Experiment Set 1, both PA and HEA are initiating designs through seed proposals and extending each other's designs. The solutions found under relaxation threshold 10 (RT10) are not as good as those in Set 2 because PA is "distracted" by the need to respond (negatively) to HEA-initiated designs. It therefore generates only half the number of proposals in Set 1 as in Set 2 within 10 processing cycles. After 10 cycles, HEA relaxes a solution requirement that makes some of the already existing designs immediately acceptable and the system stops. Although some designs initiated by PA are acceptable, the quality of the designs is not as good as it would be if PA had generated more seed proposals. This is seen in the Set 1 and Set 2 results under relaxation threshold 15 (RT15). These results are identical because PA generated enough proposals in each case to find identical solutions. Although the time of creation for individual designs is not represented in the table, the Set 2 solutions were generated sooner than those in Set 1, but did not become acceptable until relaxation occurred at cycle 15.

In Experiment Set 3, HEA is the controlling agent with PA responding. Performance is much worse, both in terms of quality and processing time. This was expected since HEA has no way of intelligently guiding its proposal generation process and mutually-acceptable solutions are found in a hit-or-miss fashion. The quality of solutions improves under RT15 because HEA searches longer for high-quality solutions.

Returning to the generic two-agent example discussed above, consider the case where both agents have functional relationships that cannot be communicated. This forms a very problematic agent set: the best-case scenario is that by random generation of seed proposals and extension of designs, eventually the two agents find an overlapping solution. In this case both agents should instantiate *generate-proposal* and *extend-design* since neither has an advantage in controlling the interactions. If the composite space is large and sparse, any mutually consistent solution found may be considered acceptable (the relaxation threshold should be very low). On the other hand, if the composite space is small enough and dense enough that the agents are likely to find a good fit eventually, the relaxation threshold should be higher. In earlier work (Lander *et al.* 1991), we named this strategy *generate-random-alternatives* and described several other strategies.

7 Negotiatied-Search Strategies

This section examines how a particular negotiated-search strategy, *compromise*, relates to solution space characteristics of a set of agents. We narrowly define the compromise strategy to be a series of negotiatied-search operators in which two or more agents adjust the value of a shared parameter by sliding the potential value along some ordered scale. As the parameter value is increased, the utility of the solution is decreased for one agent and increased for another. An example of compromise is buying/selling a house, where the parameter in question is the price of the house and the scale is the numeric scale. Note that compromise includes a synchronized sequence of planned actions (compromise *operators*) by at least two agents.

What characteristics make compromise a viable option for a particular agent? The first requirement is that a monotonic relationship exists between utility value and potential values of a solution parameter: given a utility function, $U(p_1...p_n)$, if the value of p_1 increases, the value of $U(p_1...p_n)$ either strictly increases or decreases. If an agent has this characteristic, locally defining a compromise *operator* is potentially useful. If two agents share this characteristic over some p_x and if their utility functions are inversely related over p_x, the compromise *strategy* can be applied. Their local utilities, U_A and U_B, can be adjusted in a controlled manner by sliding the value of p_x along the scale until an acceptable intermediate value is reached.

Compromise can be thought of as a shortcut to iterative applications of extended search and relaxation operators. By taking advantage of the utility/parameter-value relationship, it is possible to focus the search on a small section of the space and to skip multiple relaxation/search cycles by having agents agree to relax requirements equally on that parameter. However, the applicability of the compromise strategy depends on the existence of both the intra-agent utility/parameter value relationship and the inter-agent inverse-utility relationship.

8 Conclusions

In this paper, we introduce the concept of negotiated search, a multi-agent distributed search paradigm that integrates search and relaxation to enable cooperative solution development. The TEAM framework has been developed to support negotiated search and is demonstrated in an implemented application for the parametric design of steam condensers. The application currently includes seven agents although the experiments in this paper were run with three agents for simplicity.

In the experiments, we examine the impact of agent and group characteristics on the selection of operators in negotiated search and specifically focus on two points: 1) how agent characteristics can be used to select operators that produce the most effective overall performance; and 2) how group characteristics can be used to develop agent sets that take advantage of inter-agent relationships to improve performance.

We note the influence of an agent's ability to communicate constraining information and of the texture of its local solution space on the effectiveness of the agent's role in problem solving. Solution space texture includes characteristics such as the size and density of the space and its intersection with the composite space. In general, we observe that the best system performance occurs when a tightly constrained agent with little ability to provide guiding information to other agents has sole control of the initial generation of design proposals. However, performance is also influenced by solution space texture and by the degree to which agents have different characteristics.

In looking at group characteristics, we note that a negotiated-search strategy (a coordinated sequence of relatively simple operators across agents) can take the place of uncoordinated iterative cycles of search and relaxation if a particular set of conditions exist. We explore the use of negotiated-search strategies further in (Lander and Lesser 1992).

TEAM has proven to be an effective tool for experimenting with negotiated search. It provides a flexible environment for testing hypotheses about agent/operator relationships and the group dynamics of agent sets. These results are important because they provide a concrete foundation for guiding the design of agents for multi-agent sets, for deciding whether or not a particular agent is an appropriate candidate for inclusion in a particular set, and for determining the role an agent should play within a set.

References

[Erman *et al.*, 1980] 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.

[Fisher and Ury, 1981] R. Fisher and W. Ury. *Getting to Yes: Negotiating Agreement without Giving In*. Houghton Mifflin, 1981.

[Fox *et al.*, 1982] 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.

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

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

[Lander and Lesser, 1991] Susan E. Lander and Victor R. Lesser. Negotiated search: A framework for cooperative design. Technical Report 91-79, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, November 1991.

[Lander and Lesser, 1992] Susan E. Lander and Victor R. Lesser. Customizing distributed search among agents with heterogeneous knowledge. In *Proceedings of the First International Conference on Information and Knowledge Management*, pages 335–344, Baltimore, Maryland, November 1992.

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

[Lander, 1993] Susan E. Lander. *Distributed Search in Heterogeneous and Reusable Multi-Agent Systems*. PhD thesis, University of Massachusetts, Amherst, Massachusetts, 1993. In preparation.

[Lesser and Corkill, 1983] Victor Lesser and Daniel Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, 1983. (also in *Blackboard Systems*, R. Engelmore and T. Morgan (eds.), pp 353-386, Addison-Wesley, 1988).

[Meunier, 1988] Kenneth L. Meunier. Iterative respecification: A computational model for automating parametric mechanical system design. Master's thesis, University of Massachusetts, Amherst, Massachusetts 01003, February 1988.

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

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

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

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

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