Multistage Negotiation for Distributed Constraint Satisfaction

Susan E. Conry, *Member, IEEE*, Kazuhiro Kuwabara, Victor R. Lesser, *Member, IEEE*, and Robert A. Meyer, *Member, IEEE*

Abstract—**A cooperation paradigm and coordination protocol for a distributed planning system consisting of a network of semi-autonomous agents with limited internode communication and no centralized control is presented. A multistage negotiation paradigm for solving distributed constraint satisfaction problems in this kind of system has been developed. A distributed constraint satisfaction problem arises when multiple goals must be satisfied, with satisfaction of each goal requiring a coordinated set of actions distributed over the network, subject to local constraints at each node. The strategies presented in this paper enable an agent in a distributed planning system to become aware of the extent to which its own local decisions may have adverse nonlocal impact in planning. An example problem is presented in the context of transmission path restoral for dedicated circuits in a communications network. Multistage negotiation provides an agent with sufficient information about the impact of local decisions on nonlocal state so that the agent may make local decisions that are correct from a global perspective, without attempting to provide a complete global state to all agents. Through multistage negotiation, an agent is able to recognize when a set of global goals cannot be satisfied—a condition making the problem overconstrained—and also is able to solve a related problem by finding a way of satisfying a reduced set of goals.**

I. Introduction

WE CONSIDER a distributed problem-solving system to be a computer network in which many problems or tasks are presented to the network for solution by one or more processing nodes within the network. In the systems we consider, problems may be presented to the network by one or more nodes and they generally cannot be solved by a single processing node acting alone. The networks of interest to us are the "coarse grained" variety in which each processing node is a semiautonomous problem solver having a finite set of local resources and a limited communication capacity with other nodes. A distributed problem-solving system is characterized by the absence of any centralized decision maker or control authority. The network attempts to complete all tasks by some form of cooperative action among the nodes.

In this paper, we present multistage negotiation as a cooperation paradigm and a coordination protocol that results in effective network problem solving. The strategies we describe illustrate the issues involved in performing a complex distributed search in which incompatibilities among solutions to interacting subproblems may not be directly observable at an agent, but can only be recognized as a more global view of the solution to the entire problem is constructed.

Within the context of distributed problem solving systems in general, an important class of problems known as distributed resource allocation problems has been extensively studied. Examples include job scheduling in distributed operating systems [18], distributed file allocation [10], and routing in packet switched networks [9]. In each of these applications, the problem solving goal is to find an allocation of resources among the network nodes that optimize some system performance measure.

Here we consider a related class of problems in which satisfaction of each goal presented to the network requires a coordinated set of actions distributed over a subset of the nodes for completion. Each node has limited resources available for satisfaction of global goals, and once a resource has been committed in satisfaction of one goal, it cannot be used for another. Planning relative to satisfaction of the set of global goals is nontrivial. The combination of local resource constraints and required coordination of actions among nodes that results gives rise to a complex set of global, interdependent constraints. We refer to these problems as *distributed constraint satisfaction problems*. They are related to classical constraint satisfaction problems [14] and other types of distributed constraint satisfaction problems [19], [22] in that allocation of resources within an agent for satisfaction of a particular goal is analogous to assigning a value to a variable in these more classical problems.

Solving this kind of distributed constraint satisfaction problem requires two primary steps. First, as each goal is introduced to the network, a set of alternative goal decompositions must be generated in which each alternative represents one feasible means for satisfying a specific goal. (This step corresponds to identification of the sets of values that variables can take in conventional constraint satisfaction problems.) The second step is a distributed search among these alternatives to identify a set of choices that enables the satisfaction of

Manuscript received December 3, 1990; revised May 1, 1991. This work was supported in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, NY, in part by the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008 (which supports the Northeast Artificial Intelligence Consortium (NAIC)), and in part by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD) (monitored by the Office of Naval Research under Contract NR049-041).

S. E. Conry and R. A. Meyer are with the Electrical and Computer Engineering Department, Clarkson University, Potsdam, NY 13699.

K. Kuwabara is with NTT Communications and Information Processing Laboratories, Nippon Telegraph and Telephone Corp., 1-2356 Take, Yokosuka, 283–03, Japan.

V. R. Lesser is with the Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003.

IEEE Log Number 9102933.

as many global goals as possible. This distributed search is a complex one because it is asynchronous and agents must simultaneously attempt to identify an overall solution, avoid falling into endless cycles of activity, and recognize when the problem is overconstrained. Though none of these problems would be difficult to solve (given a global perspective), lack of a global view makes them nontrivial. Thus the distributed search step results in an iterative exchange in which agents negotiate in an attempt to acquire enough knowledge about the global state to identify conflicts that exist and resolve them. In this paper we present multistage negotiation and algorithms for accomplishing the requisite negotiation as a technique for performing the distributed search necessary in this second step.

The primary contributions of our negotiation protocol are twofold. First, multistage negotiation makes it possible to detect and resolve conflicts among locally "good" partial solutions in a distributed environment without requiring interchange of detailed local state among nodes. The solution to the set of global goals is never known to any agent in the system. Second, multistage negotiation permits the network to detect overconstrained problem sets and act to remedy the situation by finding a solution to a reduced problem set that is related but feasible, given the current set of constraints.

Multistage negotiation is specifically *not* intended as a mechanism for problem decomposition in the system, but rather as a means for selecting one possible decomposition from among several alternatives. Our protocol may be viewed as a generalization of the contract net protocol [6], [16], [17]. The contract net was devised as a mechanism for accomplishing task distribution among agents in a distributed problem solving system. In a contract net, task distribution takes place through a negotiation process involving contractor task announcement followed by bids from competing subcontractors and finally announcement of awards. Multistage negotiation generalizes this protocol by recognizing the need to iteratively exchange assessments made by a node about the impact of its own choice concerning what local tasks to perform in contributing to a solution of *global problems*.

Multistage negotiation produces a cooperation strategy similar in character to the functionally accurate/cooperative paradigm [12] in which nodes iteratively exchange tentative and high level partial results of their local subtasks. This strategy results in solutions that are incrementally constructed to converge on a set of complete local solutions that are globally consistent.

In the sections that follow, we first provide an intuitive introduction to the problems that are encountered in dynamically solving distributed constraint satisfaction problems. We then describe an application domain in which these types of problems must be solved in a diffuse manner. This application domain forms the framework within which we illustrate the details of our protocol. In Section IV, we formalize the problem, giving precise formulations of the kinds of information concerning local and nonlocal conflict that must be propagated. In this section, we also indicate how this knowledge concerning impact is propagated among problem solving nodes. After that, we show how internode coordination is accomplished so that the protocol results in convergence on a solution. Finally, we discuss issues related to implementation and evaluation of these strategies, with the aid of a trace of typical activity during problem solving.

TABLE I
A SIMPLE SCENARIO—GLOBAL VIEW

Agent	Goal	Alternative	Resources Required
A	g1	a1	r1, r2, r3, r4
		a2	r2, r5, r6
B	g2	b1	r6, r8
		b2	r4, r7
C	g3	c1	r1, r7
		c2	r5, r8

TABLE II
RESOURCE CONTROL FOR THE SIMPLE SCENARIO

Agent	Resources
A	r1 r2 r3
B	r4 r5 r6
C	r7 r8

II. SOLVING DISTRIBUTED CONSTRAINT SATISFACTION PROBLEMS

In this section, we give an overview of the problem solving process, with emphasis on providing an intuitive feel for the nature of the problems that are encountered in dynamically solving distributed constraint satisfaction problems. We presume that the first phase of problem solving has already been accomplished (as described in [15]), so that several alternatives for goal satisfaction have been identified, each of which is feasible, when considered in isolation.

As an example, consider a situation in which there are three system goals: $g1$, $g2$, and $g3$. In this scenario, Agent A has primary responsibility for satisfying $g1$, Agent B for satisfying $g2$ and Agent C for satisfying $g3$. Globally, there are two alternative ways of satisfying each of the goals. These are summarized in Table I.

It is easy to see, given this global perspective of the problem, that alternatives a1, b1, and c2 are in conflict because multiple copies of resource r8 would be required to implement this family of alternatives. Alternatives a2, b2, and c1 are in conflict for a similar reason. (We do not regard alternatives a1 and a2 as being in conflict because they are both alternatives for the same goal, hence would never be in contention for resources.) Indeed, given the global perspective, it is immediately evident that it is not possible to satisfy all three global goals: the problem is overconstrained.

Consider the problem from the agents' local perspectives, however. Each agent only has knowledge concerning its own local resources and those resources whose control it may share with another agent. Suppose that resource control is distributed according to Table II.

From this perspective, it is evident that Agent A has knowledge regarding only those portions of alternatives a1, a2, and c1 that utilize resources it controls. Similarly, Agent B knows only about those portions of alternatives a1, a2, b1,

b2, and c2 that involve its resources and Agent C is aware of fragments of alternatives b1, b2, c1, and c2. Each agent can find portions of solutions that are locally feasible for each of the goals about which it has knowledge. There is no evidence in any agent's local knowledge that would allow it to infer that the overall problem of satisfying all three global goals is infeasible.

It is important to observe (based on Table II) that selection of alternative a1, for example, requires the coordinated action of agents A and B and the selection of alternative b2 involves joint activity on the part of agents B and C. This need for coordinated activity makes it necessary for agents to be able to assess what the impact of their own local choices is, nonlocally.

In our work, the impact of any local decision is ultimately due to the fact that a particular resource cannot be committed more than once. We capture the essence of this lowest level contention among resources in a locally determined *conflict set*. The conflict set (in agent i) for a local fragment f of alternative $a1$ consists of other alternative fragments in agent i that are in contention with fragment f over some resource controlled by agent i. In the aforementioned scenario, the conflict set in Agent B for the local fragment of alternative a1 would contain the local fragment of alternative b2.

If all of the locally known fragments for potential satisfaction of some goal g_i are in contention with some local fragment f for another goal g_j, then fragment f locally *excludes* satisfaction of g_i. Thus the information contained in local conflict sets can be aggregated to determine (local) *exclusion sets* for each fragment.

As is evident from the example above, knowledge about which fragments are in local contention with other fragments is not adequate for arriving at a clear picture of the nonlocal conflict that is present. It is necessary, for example, for Agent C to understand that its selection of local fragments associated with alternatives b2 and c2, though apparently feasible, leads to conflict when viewed from Agent B's perspective (because Agent B would have no way of satisfying g1 if its choices that are compatible with alternatives b2 and c2 were selected). It is clear that a form of choice exclusion may be induced by nonlocal constraints. Thus there is a need for propagating information concerning *induced exclusion* of alternatives.

Finally, detection of overconstrained situations is necessary. In the scenario we have discussed, agents must recognize that it is possible to satisfy any two of the three goals, but not all three. Thus it is necessary to construct *induced goal exclusion sets* that indicate what subsets of goals are intrinsically in conflict with one another.

The information about nonlocal conflict that is required to solve distributed constraint satisfaction problems of the kind described here is aggregated through multistage negotiation. In negotiation, agents first make tentative commitments to local alternatives that would serve to partially satisfy the goals for which they are responsible. In making a tentative commitment, an agent removes the affected resources from the pool of available resources.

After making a tentative commitment to some alternative or set of alternatives, an agent requests that other agents confirm this commitment by making their own local commitments in ways that are compatible. If all of the agents involved in a global alternative are able to do this, the agents have arrived at an acceptable solution. If some agent finds that it cannot do so, it replies that it cannot, and other agents must revise their tentative commitments. This leads to an iterative exchange that results in the construction of induced exclusion sets and goal exclusion sets in the agents.

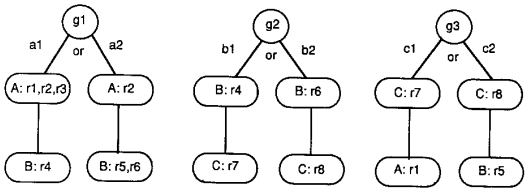


Fig. 1. Distributed depth first search.

The process of attempting satisfaction of multiple global goals occurs simultaneously in the network. Each goal is processed by carrying out what is essentially a depth first search of its space of plan alternatives, as depicted in Fig. 1. Problems arise when one agent's choices act to block another's exploration of its space of alternatives. When a tentative commitment cannot be confirmed, an agent has no real choice but to retract it, thereby making the resources that were previously allocated (albeit tentatively) available. It is evident that the problem solving context changes over time. It is sometimes necessary to retry alternatives that had previously failed when tentative commitments have been retracted. For this reason, the protocol must provide a mechanism for determining when it is appropriate to retry a previously explored portion of the search space and a mechanism for avoiding endless loops.

III. Application Domain

We illustrate multistage negotiation in the context of an application domain involving transmission path restoral in a complex communications system. Such a system consists of a network of sites, each containing a variety of communications equipment, interconnected by links. These sites are partitioned into several geographic subregions with a single site in each subregion designated as a control facility. Each control facility has responsibility for communication system monitoring and control within its own subregion and corresponds to a single agent in the distributed problem solving network. In order to distinguish between the communication network and the problem solving network, we reserve the term "site" to mean a physical location in the communication system. The term "node" will be used to refer to those sites at which processing and control reside.

The communication network considered here represents a long-haul, transmission "back-bone" of a larger, more complex communications system. From this transmission oriented perspective, each user is provided with a dedicated set of resources (equipment and link bandwidth) that establishes a point-to-point connection, or *circuit*, for a significant period of time. Any equipment failure or outage will cause an interruption of service to one or more users. Each service in-

TABLE III
GLOBAL PLANS IN EXAMPLE 1

goal	global plan	subgoals	resources required
$g1$	$p_{1,1}$	$1a-1b-1c-1d-1e-1f$	$r11, rb2, rb1, rb4, r21, r31, r41, re1, r51$
	$p_{1,2}$	$2a-2b-2c-2d-2e-2f$	$r12, r22, rc1, rc2, rc3, r32, rd1, r42, r52$
$g2$	$p_{2,1}$	$3a-3b-3c-3d-3e-3f$	$r13, r23, r33, rd1, r43, re1, r53$
	$p_{2,2}$	$4a-4b-4c-4d-4e-4f$	$r14, rb1, r24, rc1, r34, r44, r54$
	$p_{2,3}$	$4a-5b-5c-5d-4e-4f$	$r14, rb1, r25, rc3, r35, r44, r54$

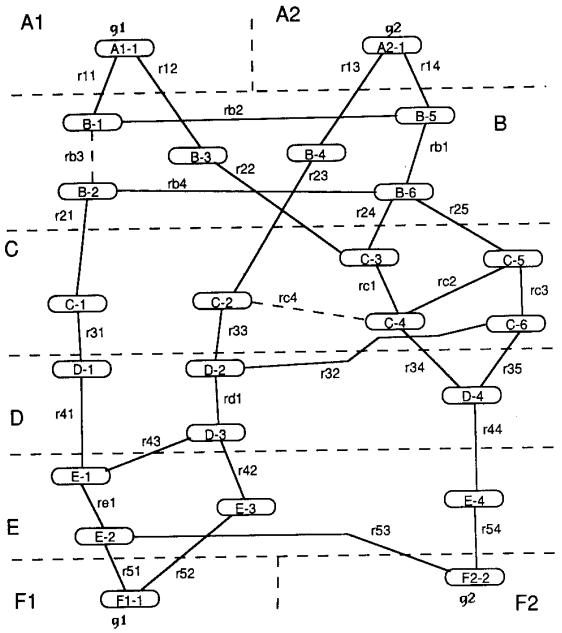


Fig. 2. Network example 1.

terruption gives rise to a network task whose solution requires transmission path restoral. Since any single interruption may be observed at one or more nodes in the network, the same network problem may arise in multiple nodes independently.

The distributed constraint satisfaction problems addressed in this paper and our approach to solving them can best be understood with the aid of an example. A simplified diagram of a small network is shown in Fig. 2.

There are eight subregions shown in Example 1, labeled A1, A2, B, C, D, E, F1, and F2. There is assumed to be one intelligent agent responsible for control within each subregion. Each site is designated by a letter-number pair, where the letter indicates the subregion in which the site is located. The communication network links are designated by r-number.

For the purposes of describing the restoral problem, we assume that there is an equipment malfunction at station B-1 that fails all communication using link *rb3* and another at site C-2 that fails all communication using *rc4*. For the sake of simplicity, we further assume that each link can handle at most one circuit.

As a result of the presumed failure, two circuits are disrupted, namely ckt-1 and ckt-2. In this scenario, ckt-1 established a path from A1–1 to F1–1 while ckt-2 involved a route from A2–1 to F2–2. The restoral activity is initiated when an agent observes disruption of a circuit terminating within its subregion and recognizes that restoral is required. In this example, the restoral goals are autonomously created in subregion A1 (for ckt-1), and subregion A2 (for ckt-2). Each agent initially has only the following knowledge about *each circuit terminating in its subregion:*

- a circuit identifier that is unique within the network,
- a priority or degree of urgency for restoral,
- detailed routing of this circuit within this node's area of responsibility, and
- the end stations of the circuit and the nodes responsible for them.

In addition, each agent has detailed knowledge concerning the status of resources resident in its subregion.

The first phase of the restoral process involves generation of alternative plans and is discussed elsewhere [15]. When viewed from a global perspective, plan generation (applied to our example) produces multiple alternative restoral plans for each circuit. Each plan is represented in Table III as a list of local subgoals, or plan fragments.

To clarify the example, we have adopted a naming convention for restoral tasks and alternative plans that incorporates the circuit name and plan number; thus the two alternative plans for restoring circuit ckt-1 are designated $p_{1,1}$ and $p_{1,2}$. It is essential to remember that these are global plans that have been generated in a distributed manner, and no single node *necessarily* knows of all plans or any one complete plan.

As a result of plan generation, each agent produces local alternative *plan fragments* that may be used to complete global restoral tasks. Each global plan then consists of a set of locally known plan fragments distributed over the network and can be viewed as a problem decomposition over the network. Relative to our example, each global plan listed in Table III is composed of several fragments (each satisfying a local subgoal) distributed over a subset of the agents. This is illustrated in Table IV, which summarizes what each node knows about goals, subgoals, alternative plan fragments, and local resources.[1] Plan fragments are numbered and each is identified by a letter indicating the responsible agent. Note that agents are not *explicitly* aware of global alternative plans, but are only aware of local alternatives. For example, even though Agent A2 has resources needed by $p_{2,1}$, $p_{2,2}$ and $p_{2,3}$, the local plan fragment is the same in two of these cases (namely $p_{2,2}$ and $p_{2,3}$). Thus Agent A2 "sees" only two alternative plans

[1] Resources $rb3$ and $rc4$ are not shown in Table IV because they are presumed to have failed, and are not available for use.

TABLE IV
LOCAL KNOWLEDGE IN EXAMPLE 1

Agent $A1$

goal	subgoal	pf	$r11$	$r12$
resource count			1	1
$g1$	$1a$	$1a$	1	
	$2a$	$2a$		1

Agent $A2$

goal	subgoal	pf	$r13$	$r14$
resource count			1	1
$g2$	$3a$	$3a$	1	
	$4a$	$4a$		1

Agent B

goal	subgoal	pf	$r11$	$r12$	$r13$	$r14$	$rb1$	$r21$	$r22$	$r23$	$r24$	$r25$	$rb2$	$rb4$
resource count			1	1	1	1	1	1	1	1	1	1	1	1
$g1$	$1b$	$1b$	1				1	1					1	1
	$2b$	$2b$		1					1					
$g2$	$3b$	$3b$			1					1				
	$4b$	$4b$				1	1				1			
	$5b$	$5b$				1	1					1		

Agent C

goal	subgoal	pf	$r21$	$r22$	$r23$	$r24$	$r25$	$rc1$	$rc2$	$rc3$	$r31$	$r32$	$r33$	$r34$	$r35$
resource count			1	1	1	1	1	1	1	1	1	1	1	1	
$g1$	$1c$	$1c$	1								1				
	$2c$	$2c$		1				1	1	1		1			
$g2$	$3c$	$3c$			1								1		
	$4c$	$4c$				1		1						1	
	$5c$	$5c$					1			1					1

Agent D

goal	subgoal	pf	$r31$	$r32$	$r33$	$r34$	$r35$	$rd1$	$r41$	$r42$	$r43$	$r44$
resource count			1	1	1	1	1	1	1	1	1	1
$g1$	$1d$	$1d$	1						1			
	$2d$	$2d$		1				1		1		
$g2$	$3d$	$3d$			1			1			1	
	$4d$	$4d$				1						1
	$5d$	$5d$					1					1

Agent E

goal	subgoal	pf	$r41$	$r42$	$r43$	$r44$	$re1$	$r51$	$r52$	$r53$	$r54$
resource count			1	1	1	1	1	1	1	1	1
$g1$	$1e$	$1e$	1			1	1				
	$2e$	$2e$		1					1		
$g2$	$3e$	$3e$			1		1			1	
	$4e$	$4e$				1					1

Agent $F1$

goal	subgoal	pf	$r51$	$r52$
resource count			1	1
$g1$	$1f$	$1f$	1	
	$2f$	$2f$		1

Agent $F2$

goal	subgoal	pf	$r53$	$r54$
resource count			1	1
$g2$	$3f$	$3f$	1	
	$4f$	$4f$		1

for goal $g2$, even though there are three global plans for $g2$ in which agent A2 participates.

This example is considerably oversimplified in order to focus attention on the significant characteristics of the problem and to illustrate the cooperation strategy that results from multistage negotiation. The communication network has been simplified so that link capacity is the only resource, and there are no constraints arising from local equipment configurations. The number of circuits and link capacities are also much smaller than is typical. Since only two system goals exist, the interactions are simple and can be recognized quickly. In a larger problem, indirect interactions often involve multiple dependencies and may require several steps of negotiation to detect and resolve.

The multistage negotiation protocol presented in this paper involves a three phase algorithm. First, an *asynchronous*

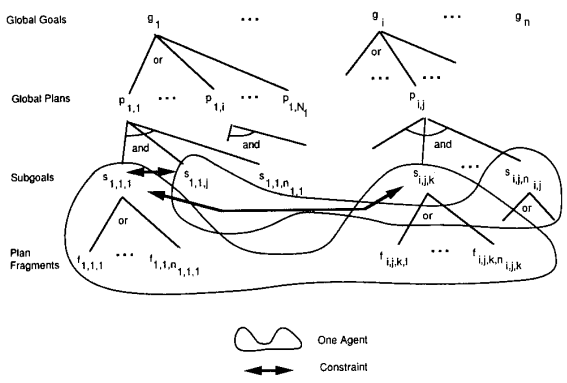


Fig. 3. Global search space in the multistage negotiation.

search phase is initiated to identify sets of joint agent actions that satisfy as many system goals as possible. During this phase, agents exchange information concerning the nonlocal impact of their local tentative decisions. This exchange of information gives agents the ability to recognize conflicts early in problem solving and serves to guide the search in more fruitful directions. Second, when it becomes clear that there is not enough information available, even after all local choices have been examined, a *coordinated search* phase is entered. During this coordinated search, a complete but abstracted view of nonlocal conflict in the problem solving network is constructed. Third, it is possible that the problem is overconstrained in the sense that not all of the global goals can be satisfied, given the current set of local constraints. When an overconstrained problem is recognized, an *overconstrained resolution* phase is required to select a suitable set of global plans for execution.

IV. Formalizing the Problem

As has been indicated (and as is shown in Fig. 3), we can view the global search space as consisting of a set of high-level global goals. Each global goal, g_i, has associated with it a set of global plans, $\{p_{i,j}\}$, any one of which can satisfy the goal. A global plan, $p_{i,j}$, is further decomposed into subgoals, each of which corresponds to establishment of point-to-point transmission paths between two sites. Let $s_{i,j,k}$ denote the kth subgoal for a global plan $p_{i,j}$. Since subgoals are related to establishment of portions of a circuit in a region, each subgoal is associated with a specific agent (namely the agent responsible for control activity in the relevant geographic region). It should be noted that a subgoal may belong to multiple global plans because a single subpath may be a component of several distinct sets of global restoral plans. In addition, an agent may know of multiple ways of satisfying a subgoal. These multiple sets of control actions for subgoal satisfaction are represented as plan fragments, $f_{i,j,k,l}$.

When an agent begins its restoral activity, it has knowledge of goals that have been locally instantiated. A space of alternatives to perform some part of these tasks has been formulated during plan generation without regard for any interaction problems. After this phase, each agent is aware of two kinds of goals: *primary goals* (or p-goals) and *secondary goals* (or s-goals). In this application domain, p-goals are those instantiated locally by an agent in response to an observed outage of a circuit for which the agent has primary responsibility (usually because the circuit terminates in the agent's subregion). An agent's p-goals are important in general because negotiation is initiated relative to a particular system goal g by the agent that has g as a p-goal. An agent's s-goals are those that have been instantiated as the result of a request from some other agent. An agent regards each of its s-goals as a possible alternative to be utilized in solution to a problem relative to some other agent's p-goal.

Resources (link capacity) in the system either reside in an agent, or are shared by two agents. When a resource is shared by two agents, utilization of the resource must be coordinated between both agents. Suppose that a resource $r1$ is shared by two agents, agent A and agent B. If $r1$ is allocated in agent A in order to satisfy subgoal $s_{k,i,j}$, agent B must also utilize $r1$ in satisfying its subgoal $s_{k,i,l}$ that is part of the same global plan. This means that agent B must select a subgoal that requires $r1$, so negotiation between agents is required.

The concepts of conflict, exclusion, goal exclusion, and induced exclusion were intuitively discussed in Section II. In order to devise algorithms that properly ensure the requisite exchange of knowledge among agents during negotiation, it is necessary to formalize these concepts associated with the propagation of knowledge concerning local and nonlocal impact of local decisions. Our formalisms are given in terms of first order logic and are used as the basis for the negotiation algorithms we have developed.

A. Conflict Set

The *conflict set* for a subgoal s represents a set of subgoals that cannot be selected for satisfaction concurrently with subgoal s. Intuitively, it represents the minimal local (negative) impact of selecting subgoal s for satisfaction. It also embodies the effects that local resource constraints have on the set of alternative actions available to an agent. Once computed, the conflict set for a subgoal does not change as problem solving proceeds, because it is based only on local constraints, *prior* to commitment of any resources during negotiation. Computation of the conflict set for a subgoal is based on the observation that any set of plan fragments for distinct goals is incompatible (or in conflict) if it requires more resources than are locally available.

In Example 1 (see Fig. 2 and Table IV), agent B has five subgoals: $\{1b, 2b, 3b, 4b, 5b\}$. Each subgoal has one plan fragment satisfying it. Since plan fragments $1b$ and $4b$ use the same resource, $rb1$, and there is only one copy of $rb1$, $1b$ and $4b$ cannot be selected at the same time. Similarly, $1b$ and $5b$ cannot be selected simultaneously. Thus, $\{1b, 4b\}$ and $\{1b, 5b\}$ are not compatible.

In our logical formulation, we assign the value *true* to a subgoal s_i when it is selected and *false* when it is not selected. Thus, when the formula $s1 \wedge s_2$ holds, both s_1 and s_2 have been selected.

Let $D_k = \{s_1, s_2, \ldots, s_{n_{D_k}}\}$ be a set of subgoals for distinct goals. Whenever D_k is not compatible, the following formula holds:

$$\neg(\bigwedge_{s_i \in D_k} s_i).$$

Suppose now that $\{D_1, D_2, \ldots, D_n\}$ is a collection of sets, each of which is not compatible. Whenever subgoal $s_i \in D_k$ and D_k is a set of subgoals for distinct goals that is *not* compatible, s_i cannot be satisfied along with all the other subgoals in D_k. Thus we let

$$N_{s_i} = \{D_k | \; s_i \in D_k \wedge D_k \text{ is not compatible}\}.$$

Logically, we observe that N_{s_i} can be represented as

$$\bigwedge_{D_k \in N_{s_i}} \neg(s_i \wedge (\bigwedge_{s_j \in D_k} s_j)).$$

This is equivalent to

$$\neg s_i \vee (\bigwedge_{D_k \in N_{s_i}} \bigvee_{s_j \in D_k, j \neq i} (\neg s_j)).$$

The second term in this disjunction corresponds to the conflict set CS_{s_i} defined in [4]. We therefore define the conflict set for a subgoal s_i as

$$C_{s_i} = (\bigwedge_{D_k \in N_{s_i}} \bigvee_{s_j \in D_k, j \neq i} (\neg s_j)).$$

Recall that Example 1, in agent B, the sets $\{1b, 4b\}$ and $\{1b, 5b\}$ are not compatible. Therefore,

$$N_{1b} = \{\{1b, 4b\}\{1b, 5b\}\}.$$

This is logically equivalent to the following:

$$N_{1b} = \neg 1b \vee (\neg 4b \wedge \neg 5b).$$

Informally, this says that "one must either give up $1b$ or both $4b$ and $5b$." The conflict set C_{1b} is given as $\{\{4b, 5b\}\}$, logically represented as, $C_{1b} = \neg 4b \wedge \neg 5b$.

B. Exclusion Set Construction

The conflict set for a subgoal s_i represents sets of subgoals that cannot be selected for satisfaction concurrently with s_i. The exclusion set for a subgoal indicates sets of other subgoals and alternative plan fragments satisfying those subgoals that cannot be utilized in the event that this particular subgoal is selected for satisfaction.

The exclusion set for a subgoal s_i is defined in terms of a collection of sets of global *goal descriptions*. For the purposes of this discussion, a goal description is a pair consisting of a goal and a *choice list*. Each choice-list is associated with a subgoal and intuitively corresponds to a set of alternative plan fragments for satisfaction of that subgoal that have been explored in negotiation. Since global plans are not known to agents beforehand, choice lists must be incrementally constructed and updated during problem solving.

Choice Lists: The negotiation phase of problem solving begins in some agent having at least one p-goal. This agent examines its own p-goals, makes a tentative commitment, and sends a message to other appropriate agent(s) to confirm this tentative commitment. This message serves to initiate negotiation regarding the relevant global goal. It consists of a subgoal in the tentative commitment, resources to be shared, and the number of copies of each shared resource required.

The initial choice-list for a subgoal of g in the agent that has g as a p-goal is of the form

(agent index total-number).

Here, *agent* refers to the agent that has the p-goal for g, *index* represents a local identifier for the tentative choice, and *total-number* refers to the total number of alternatives known locally. This initial choice-list is determined prior to negotiation. As negotiation proceeds, the agent transmits a message requesting confirmation of a tentative commitment. This message contains the subgoal for which confirmation is requested and its choice-list, reflecting the current tentative commitment.

Upon receipt of a request for confirmation, an agent examines its local options regarding alternative subgoals that are compatible with the subgoal in the confirmation request. It then selects one of its local subgoals and builds a choice-list for that subgoal by forming the conjunction of the choice-list that was transmitted and its local choice for satisfaction of that subgoal. The agent's local choice is determined through examination of local data structures that are similar in form to routing tables, as shown in Table V. Negotiation continues as this agent in turn transmits requests for confirmation.

Formally, the choice list is represented as a disjunction of terms, each of which is of the form $\bigwedge$*local-id*. Thus

$$\textit{choice-list} = \bigvee \bigwedge \textit{ local-id.}$$

In this expression, each set of conjuncts represents one set of alternatives for satisfaction of the subgoal that has been explored during negotiation (so far). Each *local-id* in a choice-list represents a choice made at an agent, so

$$\textit{local-id} = (\textit{agent index total-number}) | (\textit{entry-id index total-number}).$$

The choice list transmitted in a message is used in identifying the global plan to which a particular subgoal belongs. As has been mentioned before, it is possible for a given subgoal to be associated with different global plans, and the choice list provides a compact abstracted representation for global plans that is similar in nature to the tags that are used during plan generation [15]. Choice-lists, however, play a crucial role in control for multistage negotiation. Incremental construction of the choice-list can be illustrated using a simple scenario drawn from Example 1.

In Example 1, $4e$ is selected when either $4d$ or $5d$ is selected by agent D. In this example, the choice list of $4d$ is ($A2$ 2 2) $\wedge$ (B-4 1 2) and the choice list of $5d$ is ($A2$ 2 2) $\wedge$ (B-4 2 2). Suppose that agent E receives a request for confirmation of $4d$ first, and agent E updates the choice

TABLE V
INTERNAL TABLE OF AGENT agent $BinExample1$

entry-id	goal	resource	resource count	subgoal list
B-1	$g1$	$r11$	1	$1b$
B-2	$g1$	$r12$	1	$2b$
B-3	$g2$	$r13$	1	$3b$
B-4	$g2$	$r14$	1	$4b, 5b$
B-5	$g1$	$r21$	1	$1b$
B-6	$g1$	$r22$	1	$2b$
B-7	$g2$	$r23$	1	$3b$
B-8	$g2$	$r24$	1	$4b$
B-9	$g2$	$r25$	1	$5b$

list of $4e$ to that of $4d$. When agent E receives a request for confirmation of $5d$ later, it updates the choice list of $4e$ to the disjunction of the current choice list of $4e$ and the choice list of $5d$. Thus, the choice list of $4e$, $cl(4e)$, would become:

$$\begin{aligned} cl(4e) &= ((A2\ 2\ 2) \wedge \text{(B-4 1 2)}) \vee ((A2\ 2\ 2) \wedge \text{(B-4 2 2)}) \\ &= (A2\ 2\ 2) \wedge (\text{(B-4 1 2)} \vee \text{(B-4 2 2)}). \end{aligned}$$

Because this reflects all of the options for entry B-4 in Table V, the choice list for $4e$ can be simplified to

$$cl(4e) = (A2\ 2\ 2) \wedge \text{(B-4)}.$$

Exclusion Sets: At this point, we have formalized the concept of a conflict set and indicated how choice-lists are constructed. Suppose that the conflict set for a subgoal s_i is $C_{s_i} = \bigvee\bigwedge\neg s_j$ and that $cl(s_j)$ denotes the choice-list associated with s_j. Then we define the *exclusion set*, E_{s_i}, for subgoal s_i as

$$E_{s_i} = \bigvee\bigwedge\neg\!<goal(s_j), cl(s_j)>.$$

To illustrate these constructs, we calculate the exclusion set of $1b$, E_{1b}, in Example 1. The conflict set of $1b$, C_{1b}, is $\neg 4b \wedge \neg 5b$. We again presume that the choice list of $4b$ is $(A2\ 2\ 2) \wedge$ (B-4 1 2), the choice list of $5b$ is $(A2\ 2\ 2) \wedge$ (B-4 2 2), and their goal is $g2$, E_{1b} is given as

$$\begin{aligned} E_{1b} &= \neg\!<g2, (A2\ 2\ 2) \wedge \text{(B-4 1 2)}> \\ &\quad \wedge \neg\!<g2, (A2\ 2\ 2) \wedge \text{(B-4 2 2)}> \\ &= \neg\!<g2((\text{A2 2 2}) \wedge \text{(B-4 1 2)}) \\ &\quad \vee ((\text{A2 2 2}) \wedge \text{(B-4 2 2)}))> \\ &= \neg\!<g2, (A2\ 2\ 2) \wedge \text{(B-4)}>. \end{aligned}$$

This means that a decision on Agent B's part to select $1b$ for $g1$ means that Agent B may not satisfy $g2$ using its local entry B-4 and Agent A2 may not use its alternative $3a$ for $g2$. (The interested reader can find more details concerning choice-list and exclusion set construction and simplification in [5].)

C. Induced Exclusion

The initial local exclusion set for a subgoal s_i, E_{s_i}, represents the local impact of selecting s_i. This local exclusion set is calculated using only local resource constraints as reflected in choice-lists initially determined for each subgoal. Information contained in local exclusion sets is exchanged as agents communicate so that they can learn about nonlocal conflicts that are related to their local alternatives. The resulting knowledge is accumulated in an *external exclusion set*. The overall exclusion set that combines the local exclusion set and the external exclusion set is called an *induced exclusion set*.

The induced exclusion set for a subgoal s_i, I_{s_i}, represents the local *and* nonlocal impact of selecting s_i. When negotiation begins, it is the same as the local exclusion set, and it monotonically grows as the agent receives information about conflicts from other agents (in its external exclusion set).

Suppose that agent A receives a request from agent X to confirm agent X's local choice s_X and agent A knows of n alternatives $s_{A,1}, \ldots, s_{A,n}$ that it could select with s_X. If all these alternatives turn out to cause conflicts, agent A should transmit information concerning these conflicts to agent X. Thus, it is necessary to combine the induced exclusion sets of $s_{A,1}, \ldots, s_{A,n}$ to reflect aggregated nonlocal impact.

In agent A, we know that

$$\neg s_{A,i} \vee I_{s_{A,i}}\ (1 \leq i \leq n). \tag{1}$$

Because s_X is only part of a global plan, whenever it is selected in the final solution, one of $s_{A,i}$ must also be selected. Conversely, when one of $s_{A,i}$ is selected in the final solution, so must s_X. Consequently:

$$\neg subgoal \Leftrightarrow \bigvee_i s_{A,i} \tag{2}$$

From (2),

$$\neg s_X \vee (\bigvee_i s_{A,i}). \tag{3}$$

From (1) and (3), the following can be derived:

$$\neg s_X \vee (\bigvee_i I_{s_{A,i}}). \tag{4}$$

Therefore, agent A should send the following information to agent X:

$$\bigvee_i I_{s_{A,i}}.$$

Agent agent X will then update the induced exclusion set of s_X to be the conjunction of this expression and its current induced exclusion set for s_X.

To illustrate, we again consider Example 1. When agent B receives a request for confirmation of $4a$ from agent A, it finds two local candidates, $4b$ and $5b$. Therefore, agent B needs to combine the induced exclusion sets of $4b$ and $5b$ to aggregate the exclusion set information concerning $4a$. Suppose that agent B has received exclusion set information from agent C concerning $4b$ and $5b$. In this case, we assume that

$$I_{4b} = \neg\!<g1, (A1)>$$

and

$$I_{5b} = \neg\!<g1, (A1)>.$$

The induced exclusion set concerning $4a$ that is transmitted from agent B to agent A is

$$I_{4b} \vee I_{5b} = \neg<g1, (A1)>.$$

On receipt of this knowledge, agent A accumulates it in the external exclusion set of $4a$.

Unfortunately, knowledge of how a local choice impacts *one* other agent is not enough in general. For agent A to select a particular subgoal s_A, agent A must obtain confirmation from all of the other agents with whom coordinated action is required. When the replies from other involved agents indicate that there is a conflict, it is necessary for agent A to combine the relevant induced exclusion sets and add the resulting knowledge to the induced exclusion set for s_A.

For the purposes of exposition, suppose that s_{X_i} is the subgoal in agent X_i, which must be selected in conjunction with s_A. The induced exclusion set for s_A, I_{s_A}, is defined recursively as

$$I_{s_A} = (\bigwedge_i I_{s_{X_i}}) \wedge E_{s_A}.$$

To illustrate this, we consider the induced exclusion set of $1a$, I_{1a}, in Example 1. $1a$ is a part of a global plan $p_{1,1}$, which uses subgoals $1a$, $1b$, $1c$, $1d$, $1e$ and $1f$. Thus, I_{1a} eventually becomes the conjunction of exclusion sets of subgoals $1a$, $1b$, $1c$, $1d$, $1e$ and $1f$. Suppose that E_{1b} is $\neg<g2, (\text{A2 2 2}) \wedge \text{(B-4)}>$ and E_{1e} is $\neg<g2, (\text{A2 1 2})>$, and all other exclusion sets, E_{1a}, E_{1c}, E_{1d} and E_{1e} are empty. Therefore,

$$\begin{aligned} I_{1a} &= E_{1b} \wedge E_{1e} \\ &= \neg<g2, (A2\ 2\ 2) \wedge \text{(B-4)}> \wedge \neg<g2, (A2\ 1\ 2)>. \end{aligned}$$

The choice list in this expression can be simplified, using simplification rules that are described in [5] so that

$$\begin{aligned} I_{1a} &= \neg<g2, ((A2\ 2\ 2) \wedge \text{(B-4)}\ ) \vee (A2\ 1\ 2)> \\ &= \neg<g2, (A2) \wedge ((\text{B-4}) \vee (A2\ 1\ 2))> \\ &= \neg<g2, ((A2) \wedge (\text{B-4})) \vee ((A2) \wedge (A2\ 1\ 2))> \\ &= \neg<g2, (A2) \wedge \text{(B-4)}>. \end{aligned}$$

Notice that here the choice list in the goal description reveals that there is conflict with all known alternatives from agent $A2$, so agent $A1$ can recognize that $1a$ conflicts with $g2$.

D. Goal Exclusion

As has been observed in the preceding discussion, exclusion sets and induced exclusion sets are associated with subgoals and reflect sets of other subgoals (and alternatives for satisfying them) that cannot be used if the associated subgoal is selected for satisfaction. We have observed that when all known alternatives for possible satisfaction of a p-goal are represented in a goal description, it is possible for agents to conclude that their local alternatives are in conflict with satisfaction of global goals. Extending our formalisms slightly, we see that it is also possible for agents to recognize when subsets of global goals are in conflict with one another.

Suppose that p-goal g has n alternatives $s_1, \ldots, s_n$ in an agent. As we have observed before, this means that

$$\neg g \vee (\bigvee_i I_{s_i}).$$

In English, this means that we either abandon g or we incur the impact reflected in the exclusion set of one of its local alternatives. This motivates the definition of a *local goal exclusion set* for g, GX_g as

$$GX_g = \bigvee_i I_{s_i}$$

where GX_g represents the impact of satisfying g. It contains goal descriptions that conflict with g. This set is called *local* because it does not take into account any indirect interactions among global goals.

Consider, for example, the local goal exclusion set for $g2$ in Example 1. In agent $A2$, there are two alternatives, $3a$ and $4a$, for $g2$. The induced exclusion set of $3a$, I_{3a}, is eventually given as $\neg<g1, (\text{A1})>$, and the induced exclusion set of $4a$, I_{4a}, is $\neg<g1, (\text{A1})>$. Goal $g2$ can only be satisfied by selecting either $3a$ or $4a$. Thus,

$$\begin{aligned} GX_{g2} &= I_{3a} \vee I_{4a} \\ &= \neg<g1, (A1)> \vee \neg<g1, (A1)> \\ &= \neg<g1, (A1)>. \end{aligned}$$

It is evident from this goal description that $<g1, (\text{A1})>$ indicates a conflict of $g2$ with all potential ways of satisfying $g1$. Thus, agent $A2$ can recognize that there is a conflict between $g1$ and $g2$.

Single goal conflicts of the kind illustrated in this example can be correctly detected using the mechanisms that have been discussed so far in this paper. However, our mechanisms need further extension in order to function properly when indirect interactions occur. As the size of the problem solving networks increases and subgoals become less tightly coupled, indirect interactions occur.

Suppose, for example, that we have a scenario in which there are three goals, gx, gy, and gz. In this scenario, agent agent X has gx as a p-goal, agent Y has gy as a p-goal, and agent Z has gz as a p-goal. There are five global plans: $p_{x,1}$ (for gx), $p_{y,1}$ and $p_{y,2}$ (for gy), and $p_{z,1}$ and $p_{z,2}$ (for gz). Globally, we presume that the following formulas hold:

$$\begin{aligned} &\neg(p_{x,1} \wedge p_{y,1}) \\ &\neg(p_{x,1} \wedge p_{z,2}) \\ &\neg(p_{y,2} \wedge p_{z,1}). \end{aligned}$$

Thus, no two goals are in conflict, but all three goals cannot be satisfied at the same time.

Now suppose that the goal exclusion sets ultimately determined are: In agent agent X:

$$GX_{gx} = \neg<gy, (Y\ 1\ 2)> \wedge \neg<gz, (Z\ 2\ 2)>.$$

In agent agent Y:

$$GX_{gy} = \neg<gx, (X)> \vee \neg<gz, (Z\ 1\ 2)>.$$

In agent agent Z:

$$GX_{gz} = \neg<gy,(Y\ 2\ 2)> \vee \neg<gx,(X)>.$$

No agent can recognize the conflict among goals, though partial goal conflict can clearly be detected. This phenomenon occurs because agents do not know indirect global goal interactions. In order to overcome this problem, it is necessary for agents to exchange local goal exclusion sets. Each agent accumulates transmitted local goal exclusion sets as an external goal exclusion set, in a manner similar to that employed in construction of induced exclusion sets.

Recall that the local goal exclusion set GX_g for g is given as

$$GX_g = \bigvee_i I_{s_i}$$

where each s_i is a subgoal for g in the agent that has g as a p-goal. We guarantee that agents learn about indirect conflicts by insisting that an agent transmit its local goal exclusion set to each agent that has a p-goal that appears in its local goal exclusion set.

We define an *external goal exclusion set*, EX_A for agent A as

$$EX_A = \bigwedge_{g \in IGoals(\text{agent } A)} (\neg g \vee GX_g)$$

where $IGoals$(agent A) represents the set of goals that directly or indirectly interact with p-goals in agent A. We also define an *induced goal exclusion set*, IG_A for agent A as

$$IG_A = \bigwedge_{g \in PGoals(\text{agent } A} (\neg g \vee GX_g)) \wedge EX_A$$

where $PGoals$(agent A) represents the set of p-goals in agent A. It is important to observe that the induced goal exclusion set is defined for an agent, not for each p-goal because recognition of these indirect interactions can only occur at the agent's level.

Returning once again to the scenario we have been considering, agent X transmits its local goal exclusion set GX_{gx} to agent Y and agent Z, because GX_{gx} contains references to gy and gz, which are the p-goals of agent Y and agent Z respectively. Similarly, agent Y sends its local goal exclusion set GX_{gy} to agent X and agent Z, and agent Z transmits its local goal exclusion set GX_{gz} to agent X and agent Y.

After receiving the local goal exclusion sets, agent X combines them into its induced goal exclusion set. Thus,

$$\begin{aligned} IG_X =& (\neg gx \vee (\neg<gy,(Y\ 1\ 2)> \wedge \neg<gz,(Z\ 2\ 2)>)) \\ & \wedge\ (\neg gy \vee \neg<gx,(X)> \vee \neg<gz,(Z\ 1\ 2)>) \\ & \wedge\ (\neg gz \vee \neg<gy,(Y\ 2\ 2)> \vee \neg<gx,(X)>). \end{aligned}$$

From this formula, it can be determined that

$$\neg gx \vee \neg gy \vee \neg gz.$$

We call this kind of list a *nogood-goal list* that can be regarded as an aggregation of the induced goal exclusion sets. Clearly, agent X can recognize that either gx or gy or gz should be abandoned, and agent Y and agent Z also have the same nogood-goal list.

In this protocol, each agent is required to send its local goal exclusion set to agents whose p-goal is included in its induced goal exclusion set. It need not broadcast the goal exclusion set to all agents. Suppose that there were another agent, agent I, which has gw as a p-goal and, in this example, gw has no interactions with gx, gy or gz. In this case, the local goal exclusion sets of gx, gy, and gz would not be transmitted to agent I. Thus, agent I would avoid exploring the combinations of gw and other goals.

Resolving an Overconstrained Situation: Once global goal conflicts are detected, it must be determined which goals should be abandoned. Clearly, if there are conflicts among sets of global goals, the overall problem is overconstrained and no solution exists for the entire set of global goals. Some set of goals must be abandoned so that a satisficing solution to the system's problem can be achieved. In many situations, the resolution is clear, and is based on satisfaction of a maximal number of global goals and other domain specific cost criteria.

In the scenario above, all agents, agent X, agent Y, and agent Z, recognize the following formula:

$$\neg gx \vee \neg gy \vee \neg gz.$$

This means that all the goals, gx, gy, and gz, cannot be satisfied simultaneously, but if one of three goals is abandoned, the remaining two goals can be satisfied. In this case, one heuristic that could be used for determining which goal to abandon is the total resource count required for goal satisfaction. If satisfying gx would take more resources than satisfying either of the other two goals, it should be dropped.

Based upon its nogood-goal list, each agent can reach the same conclusion concerning which goal should be given up, assuming that each agent has the same criteria for determining which goal is to be dropped. In general, criteria for determining which goal or goals are to be abandoned are domain specific. There may be a criticality measure associated with goals or a cost functional may provide the basis for making a choice. Thorough treatment of this issue is beyond the scope of this paper.

V. Coordination Among Agents

In the preceding section, we have presented a formalism and outlined an asynchronous and distributed protocol for distributed problem solving that is associated with that formalism. Unfortunately, the problem of guaranteeing that a search terminates with either the recognition of an overconstrained situation or the satisfaction of all outstanding goals is nontrivial. When agent communication only involves exchanging exclusion sets, it is possible for agents to repeatedly try the same combinations in attempting to satisfy their local goals. Since the tentative commitments of other agents may change over time, choices that were previously unavailable may become valid due to changes in availability of nonlocal resources. Recognizing that changes in the nonlocal resource context make it appropriate to once again explore a previously abandoned choice requires understanding the search state of

other agents processing goals that pose potential resource conflicts. It also requires understanding the state of other agents that are involved in treating other aspects of the goal that the agent is working to satisfy. In order to avoid repetitious behavior, it is necessary to further coordinate the search among agents.

The required coordination mechanism is based on an agent's ability to recognize that its subgoals' induced exclusion sets are stable and there can be no further change in them. When an agent recognizes this, it knows that it should stop exchanging exclusion set information and force other agents to try other alternatives to provide a new context in which to attempt satisfaction of the desired goal. When induced exclusion sets are stable, an agent also has complete information regarding relationships among global plans. Thus, agents can determine whether there are conflicts among goals and, if so, they can negotiate to determine which goals should be satisfied. They can also determine which global plans should be used to satisfy each goal (given appropriate domain dependent choice heuristics).

The use of this information on the state of exclusion sets (stable and complete or incomplete) and the addition of message protocols that force agents to explore other alternatives and to compute a complete exclusion set lead to a multiphase coordination protocol. In the paragraphs that follow, we indicate how agents can recognize when various sets are complete and describe the resulting overall control structure that arises from the formalisms we have presented.

A. Identifying a Complete Exclusion Set

The induced exclusion set of a subgoal is calculated from the local exclusion set, which in turn is locally obtained from the choice lists of conflicting subgoals and the external exclusion sets of other subgoals that belong to the same global plan. An exclusion set or choice list is said to be *complete* if it has complete information and will not change.

The local exclusion set of subgoal s in agent A is complete if, and only if, the choice lists of *all* the conflicting subgoals of s in agent A are complete. Thus, if an agent can recognize whether the choice list of a subgoal is complete or not, it can recognize whether the local exclusion set is complete or not. Thus, in Example 1, the local exclusion set of $1e$ is complete when the choice list of $3e$ is complete, because $3e$ is the only subgoal conflicting with $1e$. When a choice-list is complete, the global plan to which the subgoal associated with this choice list belongs can be identified.

Extending this line of reasoning, the external exclusion set of subgoal s in agent A is complete relative to agent X if all the induced exclusion sets received from agents other than agent X are complete. Thus, in Example 1, the external exclusion set of $1e$ is complete relative to agent D if the induced exclusion set received from agent $F1$ is complete. The external exclusion set of $1f$ relative to agent E is immediately known and is complete, because $1f$ does not introduce any conflict in agent $F1$, and $1f$ is used only with $1e$ in agent E.

Determining When the Choice List Is Complete: The choice list is originally transmitted by an agent with a p-goal. As long as the subgoal is used by only one global plan, it is trivial to recognize whether the choice list is complete or not; the first choice list an agent receives is complete. When a subgoal is used by multiple global plans, the choice list of the subgoal will change as each global plan is tried. Thus, it is necessary to provide additional information indicating whether the choice list is complete or not. When a choice list is complete, the agent can recognize which global plan is associated with the relevant subgoal. Since the global plan is not known beforehand, this observation is crucial in determining when the global plan is recognized.

In Example 1, subgoal $4e$ is used by global plans $p_{2,2}$ and $p_{2,3}$. If the global plan $p_{2,2}$ is explored first, the choice list ($A2$ 2 2) $\wedge$ (B-4 1 2) is transmitted from agent D. Since agent D knows that its subgoals $4d$ and $5d$ use the same resource $r44$, it knows that the choice list to be transmitted to agent E at this time is not a complete one for $4e$. Thus, agent D should send additional information to agent E, which says that ($A2$ 2 2) $\wedge$ (B-4 1 2) is the first of two choice lists $4e$ will receive. Using this information agent E knows that the choice list of $4e$ is not complete yet. After agent E receives another choice list, ($A2$ 2 2) $\wedge$ (B-4 2 2), it can recognize that the choice list of $4e$ is complete.

When the choice list has been propagated to the other end of a path, the agent at the end of the path must return a *choice list acknowledgment* along the path. In Example 1, this means that when the choice list of $4f$ becomes complete, agent $F2$ should send an acknowledgment to agent E. agent E then passes this acknowledgment to agent D and so on. When an agent receives this kind of acknowledgment, it can recognize that it has exhausted its alternatives. Choice list acknowledgment is the key to avoiding endless loops. (It should be noted that the techniques used to generate alternative plans during plan generation [15] ensure that there are no cycles in the path. Thus termination of this process is not difficult to guarantee.)

Complete Induced Goal Exclusion Sets: If a situation that is overconstrained is recognized, it is necessary to recognize any existing interactions among global goals and decide which goals should be abandoned. In order to correctly recognize the global goal interactions, it is necessary to determine if the induced goal exclusion set is complete or not.

The induced goal exclusion set is the combination of the local goal exclusion set and the external goal exclusion set. A local goal exclusion set is complete when the associated exclusion sets of subgoals for this global goal are complete. When a local goal exclusion set is transmitted to relevant agents, the information concerning its completeness must be attached so that an agent that receives a local goal exclusion set can determine whether the goal exclusion set it receives is complete. However, it is not trivial to recognize that the induced goal exclusion set is complete, because the agent does not know beforehand which goals interact with its own global goals.

This problem can be addressed by requiring that a p-goal agent maintain an *interacting goal list* for each of its own p-goals. This list associates each goal that interacts with the p-goal with a flag indicating whether the local goal exclusion set concerning this goal is complete or not. When an agent's

```
var
    IG = induced goal exclusion set;
    IGL = association list of a goal which directly or indirectly;
          interferes with p-goals of the agent and a flag
          which indicate the the complete goal exclusion set
          for the goal has been received or not;
when a local goal exclusion set GX_{g_i} for g_i becomes complete do
    IG:= (¬g_i ∨ GX_{g_i}) ∧ IG;
    for each g_j ∈ GX_{g_i} do
        if not assoc(g_j, IGL) Then
            push((g_j, nil), IGL);
            agent :=The agent which has g_j as a p-goal ;
            Send a request-goal-exclusion-set message for g_j to agent;
        end if;
    end do;
end do;
```

Fig. 4. Algorithm for detecting the complete induced goal exclusion set.

own local goal exclusion set becomes complete, the agent first identifies global goals conflicting with its own p-goal. This can be accomplished by examining the local goal exclusion set of the p-goal. It then sends the local goal exclusion set to the agents in the interacting goal list.

When a local goal exclusion set is received, an agent incorporates it in its own induced goal exclusion set. It then checks the interacting goals included in the local goal exclusion set received and adds them to its own interacting goal list. When a goal is added to the interacting goal list, the agent's complete local goal exclusion set is transmitted to the agent which has the newly added goal as a p-goal.

Since the interacting goal list contains all the goals that interact with the agents' p-goal directly or indirectly, the agent can recognize whether its induced goal exclusion set is complete or not by examining the interacting goal list. If all the complete local goal exclusions for goals in its interacting goal list have been received, then the agent knows that its induced goal exclusion set, which is the combination of the local goal exclusion set(s) received and its own local goal exclusion set, is also complete.

Fig. 4 summarizes the algorithm for detecting complete induced goal exclusion set.

VI. Implementation and Evaluation

One strategy for implementing a negotiation protocol based on the formalisms presented in this paper would involve first computing all choice lists. If all the choice lists are determined first, then it is possible to obtain the exclusion sets in complete form. This strategy would have agents calculate all the exclusion sets and then negotiate to determine which goals will be satisfied using which global plan. In this way, solutions can be found without backtracking. Each agent updates all the choice lists independently of other agents, and thus will never try the same path again.

This sort of scheme, however, is equivalent to a distributed breadth first search to explore all the combinations of alternatives and then establish the solution. If the problem is overconstrained, this approach does not impose any inefficiency, since the system must try all the combinations in order to recognize the complete set of goals that is involved in an overconstrained situation. If the problem is not overconstrained, however, it may be possible to find a solution without trying all the combinations.

In order to avoid unnecessary search, we have adopted an alternative strategy. In our scheme, agents with p-goals first try to establish a solution by selecting one of their alternatives for each p-goal. Then, when a conflict is detected, they exchange exclusion sets with other agents having p-goals and determine the goals to be satisfied. Finally, they establish a solution for each goal.

A. Multiphase Problem Solving Process

Our overall problem solving strategy involves three phases: an asynchronous search phase, a coordinated search phase, and an overconstrained resolution phase. In the paragraphs that follow, we describe these phases of problem solving, making reference to several message types and actions taken in response to them. The interested reader can find a complete set of messages and a description of the relevant message handlers in [5]

Asynchronous Search Phase In this phase, an agent tries to find a solution for its own p-goals. If a feasible solution for all goals is found, then the negotiation terminates. If not, an agent continues searching for solutions.

In doing this, an agent tries to select one of its subgoals providing both of the following conditions hold:

- The induced exclusion set of this subgoal is not complete. (When the induced exclusion set of the subgoal is complete, the agents have complete information regarding the interactions with this subgoal. Thus, there is no need to try to gather more information regarding the subgoal.)
- Choice list acknowledgment has not been received. (When the choice list acknowledgment has been received, agents that have subgoals of the same global plan recognize the global plan. Thus, even if the agent tried this subgoal,

there would be no information added to the subgoals of the same global plan in other agents.)

When an agent attempts to explore a subgoal for a second time, it uses a `retry` message. With this type of message, requested subgoals are selected and required resources are allocated even if previously selected conflicting subgoals need to be decommitted in order to release required resources. This `retry` message can be viewed as invoking backtracking.

Once an acknowledgment of a choice list is returned, an agent will not explore that possibility further. Eventually, then, the choice lists of all the subgoals become complete.

If an agent with p-goals recognizes that all of its subgoals for a particular p-goal have received choice list acknowledgment and it has not reached a solution, the agent moves into the next phase regarding this p-goal.

Coordinated Search Phase When an agent has updated the choice lists of all the subgoals of its global plans and still the goal exclusion set is not complete, it attempts to make the goal exclusion set complete. This involves a `request-exclusion-set` message. This message requests that an agent obtain a complete goal exclusion set for the relevant subgoal. Calculating the exclusion set may invoke a `request-choice-list` message in order to obtain the choice list of conflicting subgoals.

When the induced exclusion sets of subgoals regarding a p-goal in an agent become complete, the agent transmits its goal exclusion set to other relevant agents whose p-goal is included in the goal exclusion set. When an agent receives a complete goal exclusion set, it switches to the coordinated search phase even if it is in the asynchronous search phase. Since the goal exclusion set is not transmitted to all agents with p-goals, but only to agents with relevant p-goals, it is possible to limit the number of agents that move to the overconstrained resolution phase. Thus, negotiation is conducted among only those agents that have conflicting p-goals.

If all the goal exclusion sets become complete and no solution has been found for a p-goal, an agent moves to the overconstrained resolution phase.

Overconstrained Resolution Phase In this phase, agents determine goals to be satisfied (if the problem is overconstrained), based upon nogood-goal lists. When the goals to be satisfied are agreed upon, agents negotiate to determine which global plan to use to satisfy goals using the induced exclusion set information.

It is significant that this protocol does not require that satisfaction of noninterfering goals be coordinated. Thus, it is possible for some agents to be in the asynchronous search phase while others are in the coordinated search phase and still other agents are in the overconstrained resolution phase.

B. Example

The formalism and protocols that have been developed in this paper are complex. This complexity arises because the sets of interactions that occur are not simple and the distributed search is carried out in an asynchronous and incremental fashion. In this section, we illustrate the activity that occurs as multistage negotiation proceeds. For this purpose, we again consider Example 1. In this example, there are two goals, $g1$ and $g2$. Goal $g1$ has two global plans, $p_{1,1}$ and $p_{1,2}$, and goal $g2$ has three global plans, $p_{2,1}$, $p_{2,2}$, and $p_{2,3}$. Thus, there are six possible combinations:

$$p_{1,1}-p_{2,1} \quad p_{1,1}-p_{2,2} \quad p_{1,1}-p_{2,3}$$
$$p_{1,2}-p_{2,1} \quad p_{1,2}-p_{2,2} \quad p_{1,2}-p_{2,3}.$$

An abstracted trace of the multistage negotiation process is given in the following paragraphs. This trace is intended to provide an overall view of the incremental problem solving process and a sense of how each phase proceeds. For simplicity, the trace involves an example in which each subgoal has only one plan fragment. Thus, the local interactions among subgoals are same as the local interactions among plan fragments.

Suppose that $p_{1,1}\}$ is tried first, and the resources required for this global plan have been allocated. The following activity ensues:

1) Asynchronous Search Phase:

a) Trying $p_{2,1}$, $p_{2,2}$ and $p_{2,3}$.

Agent agent $A2$ tries to satisfy goal $g2$ and finds that all the global plans for $g2$ are in conflict with $p_{1,1}$. Using a goal description, this can be written as

$$\neg g2 \vee \neg <g1, (A1\ 1\ 2)>$$

since $p_{1,1}$ can be represented as $<g1, (\mathrm{A1}\ 1\ 2)>$. At this point, $1a$, which is a part of $p_{1,1}$, has received a choice list acknowledgment.

Since $p_{2,1}$ is in conflict with tentatively committed $p_{1,1}$ in agent E, the choice list of $3f$, which is at the other end of $p_{2,1}$, has not been updated. Thus, agent $A2$ has not received a choice list acknowledgment regarding $p_{2,1}$. Similarly, $p_{2,2}$ and $p_{2,3}$ are in conflict with $p_{1,1}$ in agent B, agent $A2$ has not received a choice list acknowledgment regarding $p_{2,2}$ and $p_{2,3}$. Agent agent $A2$ also knows that the induced exclusion sets of its subgoals are not complete yet.

b) Retrying $p_{2,1}$.

Agent agent $A2$ tries $3a$ (which is a part of $p_{2,1}$) again, because its induced exclusion set is not complete, and it has not received the choice list acknowledgment. This time agent $A2$ uses a `retry` message.

When the `retry` message reaches agent E, agent E decommits the conflicting $1e$, which is a part of $p_{1,1}$, and allocates the resources for $3e$, which is a part of $p_{2,1}$. The decommitment of $1e$ is propagated to agent $A1$, and agent $A1$ knows that $p_{1,1}$ is not appropriate.

c) Trying $p_{1,2}$.

Agent agent $A1$ tries to select another solution, $p_{1,2}$, for $g1$. Subgoal $1a$ will not be tried again, because it has received the choice list acknowledgment.

In agent D, $p_{1,2}$ conflicts with $p_{2,1}$. Agent agent D rejects the resource allocation of $p_{1,2}$, because $p_{2,1}$ is selected at this moment. Agent agent $A1$ then recognizes that $p_{1,2}$ conflicts with $p_{2,1}$.

Since $p_{1,2}$ is in conflict with tentatively committed $p_{2,1}$ in agent D, agent $A1$ has not yet received the choice list acknowledgment.

d) Retrying $p_{1,2}$.

Since the induced exclusion set of $2a$, which is a part of $p_{1,2}$, is not complete and it has not received the choice list acknowledgment, agent $A1$ tries $p_{1,2}$ again using a `retry` message. This time agent D selects $2d$, which is a part of $p_{1,2}$, and it decommits $3d$, which is a part of $p_{2,1}$.

Since the `retry` message concerning $p_{1,2}$ reaches agent $F1$, agent $A1$ eventually receives the choice list acknowledgment regarding $p_{1,2}$. Now agent $A1$ has received the choice list acknowledgment for all the global plans of $g1$.

e) Retrying $p_{2,2}$.

Since $p_{2,1}$ is decommitted, agent $A2$ retries $p_{2,2}$. Agent C decommits $2c$, which is a part of $p_{1,2}$, and selects $4c$, which is a part of $p_{2,2}$.

At this point, $p_{2,2}$ is selected, and no global plan is selected for $g1$. Since agent agent $A1$ has received the choice list acknowledgments for subgoals $1a$ and $2a$, it stops retrying and moves into the next phase concerning $g1$.

2) Coordinated Search Phase:

Agent agent $A1$ tries to make the goal exclusion set of $g1$ complete. The induced exclusion set of $1a$ is complete, but that of $2a$ is not complete at this point. Thus, Agent agent $A1$ sends a `request-exclusion-set` message concerning $2a$ to agent B. agent B passes this message to agent C.

Having received the `request-exclusion-set`, agent C calculates the local exclusion set of $2c$, which is a part of $p_{1,2}$. This is done by obtaining the choice list of conflicting subgoals, $4c$ and $5c$.

The choice list of $4c$, which is a part of $p_{2,2}$, is complete, but the choice list of $5c$, which is a part of $p_{2,3}$, is not complete. Agent agent C sends a `request-choice-list` message to agents agent B and agent D in order to obtain the complete choice list of $5c$. In reply to this message, agent B sends back a choice list for $5c$, and the local exclusion set of $3c$ becomes complete.

Since agent C has received the complete external exclusion set from agent D concerning $p_{1,2}$, the induced exclusion set of $2c$ becomes complete. Then, the newly obtained exclusion set is propagated to agent $A1$ and the induced exclusion set of $2a$ becomes complete. Now the goal exclusion set of $g1$ is complete, and agent $A1$ recognizes that $g1$ and $g2$ are in conflict.

Agent agent $A1$ sends its goal exclusion set to agent $A2$ and both agents move into the overconstrained resolution phase.

3) Overconstrained Resolution Phase

Agent agent $A1$ negotiates the goals to be satisfied with agent $A2$. At this point, agent $A2$ has established a solution for $g2$. If the agents agree to drop $g1$, then multistage negotiation terminates. If they decide to drop $g2$, agent $A2$ decommits $p_{2,2}$, and then agent $A1$ tries to establish a solution for $g1$ selecting either $p_{1,1}$ or $p_{1,2}$.

In this way, either $g1$ or $g2$ can be solved.

C. Evaluation

The protocols that are described in this paper have been implemented in Common Lisp, and experimental results have been collected for a variety of situations. In our experiments, we have collected data that measures the amount of work involved in each of the three phases of problem solving that our protocol uses. The amount of message traffic generated during a particular phase is a measure of two factors:

1) the amount of work that agents are doing as they carry out the distributed search and resolve overconstrained problems, and
2) the amount of communication overhead imposed by the distributed protocol.

For these reasons, our discussion of experimental results is focused on observations concerning interagent message traffic.

In sample runs involving the scenario found in Example 1, approximately 50% of the message traffic occurs during the asynchronous search phase, with on the order of 42% of the traffic in the coordinated search phase and the balance during overconstrained resolution. This pattern of message traffic occurs because the number of goals is small, and the impact due to various conflicts is reasonably diffuse (for the size of the problem). Thus, it takes several exchanges of nonlocal impact knowledge for agents to understand that they must coordinate their search. The coordinated search phase takes slightly less work than the asynchronous search because there are only two goals involved in this scenario and because the number of other alternatives whose exploration must be forced is not large. Because only two goals are involved in the overconstrained situation, the number of messages needed to resolve the problem is small.

A scenario involving indirect interactions results in a different pattern of message traffic. In considering a typical situation involving three goals, any two of which can be satisfied, we note that the interactions that cause the problem to be overconstrained can be indirect, so that agents are forced to exchange goal exclusion sets. When this occurred in our experiments, on the order of 28% of the messages were exchanged during the asynchronous search phase, while approximately 48% occurred during the coordinated search and the remaining 24% during overconstrained resolution. Clearly, there is much more effort required to detect that the problem is indeed overconstrained and resolve the situation when the interactions are indirect. The sets of intergoal constraints are more complex and the coupling among agents due to these constraints is not as tight.

It is difficult to accurately assess the sense in which the total number of messages generated has meaning. Clearly, when there is a larger number of global plans to explore, the volume of message traffic is higher than in situations involving fewer global plans. On the other hand when the problem is heavily overconstrained, that is detected relatively quickly and the distributed search space is pruned so that overhead is reduced. Further work is needed to assess the effectiveness of these protocols relative to others that have been developed [19], [22].

A number of other questions can be raised about the protocols presented in this paper. First, does it find a correct solution? The answer is yes, in the following sense. Whenever there is a solution that satisfies all of the system's global goals,

it will be found. On the other hand, when the problem is overconstrained, the multistage negotiation protocol will detect which goals are in conflict so that the agents can act to resolve the conflict in appropriate domain dependent ways. Whether or not the solution is optimal will depend on what measure of optimality is used. If, for example, optimality is achieved when a maximal number of global goals is satisfied, then optimal solutions can be determined using our protocol. On the other hand, if there are other cost functionals imposed on the solutions to various goals, optimality cannot be guaranteed by these protocols alone.

The issue of complexity is also of concern. It is our experience that the complexity of problem solving in these kinds of environments is directly dependent on the degree of conflict among agents and also on the degree of "locality" that these conflicts exhibit. When the goal conflicts are confined to a relatively small number of agents and there are few indirect conflicts, any overconstrained situations can be detected quickly. In circumstances involving large numbers of partial conflicts or significant amounts of indirect goal conflict, it takes longer for agents to learn about the extent of the conflict that is present. In future work, we will investigate a parameterized characterization of the protocol's complexity.

VII. Conclusion

The mechanisms presented in this paper allow agents, each of which has incomplete knowledge about system resources and awareness of only partial solutions to system problems, to intelligently cooperate in solving complex distributed constraint satisfaction problems. In developing these protocols, we have become aware that asynchronous distributed search presents a very difficult set of problem solving issues that have not been previously investigated. Because there is no global view of the world resident at any agent, the set of indirect constraints that are present in the overall system cannot be easily assessed. A form of complex constraint propagation is inherent in problem solving, and devising mechanisms that permit agents to propagate these constraints properly is nontrivial. Agents must be engaged in simultaneously determining a solution to the overall problem and recognizing when parts of the problem are overconstrained, and doing so without falling into cycles of unproductive activity.

Our multistage negotiation protocol deals with these problems by structuring the process in three phases: an asynchronous search phase, a coordinated search phase, and an overconstrained resolution phase. Each of these phases requires agents to handle goals in a progressively more coordinated way, based upon aggregated evidence found in exclusion sets. In the asynchronous search phase, which occurs first, agents try to find solutions independently of other agents. The coordinated search phase is entered when all the choices of an agent with p-goals have been tried and an acceptable solution has not been found. The overconstrained resolution phase, as its name implies, is entered when an overconstrained situation is recognized and goals need to be abandoned in order to resolve the situation.

It is certainly true that distributed constraint satisfaction problems such as those described in this paper could be solved by other systems (such as [22]). Unlike other approaches to distributed constraint satisfaction, however, our strategy makes conflict explicit. This makes it possible to reason about conflict and apply conflict knowledge in determining which constraints should be relaxed in overconstrained situations.

Acknowledgment

The results presented in this paper reflect an effort that has extended over several years. Many people have contributed ideas in a variety of ways. Randy Pope, in particular, was the source of several key ideas that have been extended and incorporated in this paper.

The authors would like to express their appreciation for the efforts that Ed Durfee has expended in contributing to the improvement of this paper. The referees have been very helpful in providing insightful comments that have resulted in substantial improvements as well.

References

[1] P. A. Bernstein and N. Goodman "Concurrency control in distributed database systems," *Computing Surveys*, vol. 13, no. 2, pp. 185–221, June 1981.

[2] S. E. Conry, R. A. Meyer, and J. E. Searleman "A shared knowledge base for independent problem solving agents," *Proc. Expert Syst/ in Gov/t Symp/*, IEEE Comput. Soc., McLean, VA, Oct. 1985.

[3] S. E. Conry, R. A. Meyer and V. R. Lesser, "Multistage negotiation in distributed planning," COINS Tech. Rep. 86–67, Comput. and Inform. Sci. Dept., Univ. Mass., Amherst, 1986. (Also in *Readings in Distributed Artificial Intelligence*, A. H. Bond and L. Gasser, Eds. San Mateo, CA: Morgan Kaufmann, 1988.)

[4] S. E. Conry, R. A. Meyer and R. P. Pope, "Reasoning about nonlocal impact of local decisions in distributed planning," *Workshop on Distributed Artificial Intelligence*, 1988.

[5] S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer, "Multistage negotiation for distributed constraint satisfaction," Tech. Rep., Dept. Elec. Comput. Eng., Clarkson Univ., Potsdam, NY, 1990.

[6] R. Davis, and R. G. Smith "Negotiation as a metaphor for distributed problem solving," *Artificial Intell.*, vol. 20, no. 1, pp. 63–109, Jan. 1983.

[7] E. W. Dijkstra and C. S. Scholten "Termination detection for diffusing computations," *Inform. Processing Lett.*, vol. 11, no. 1, pp. 1–4, Aug. 1980.

[8] N. Francez "Distributed termination," *ACM Trans. Programming Languages and Syst.*, vol. 2, no. 1, pp. 42–55, Jan. 1980.

[9] R. G. Gallager "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Communications*, vol. COM-25, no. 1, pp. 73–85, Jan. 1977.

[10] J. F. Kurose and R. Simha, "A microeconomic approach to optimal file allocation," COINS Tech. Rep. TR-85–43, Dept. Comput. Inform. Sci., Univ. Mass., Amherst.

[11] K. Kuwabara and V. R. Lesser, "Extended protocol for multistage negotiation," *9th Workshop on Distributed Artificial Intelligence*, 1989.

[12] V. R. Lesser and D. D. Corkill "Functionally accurate, cooperative distributed systems," *IEEE Trans. Syst., Man, Cybern.*, pp. 81–96, Jan. 1981.

[13] D. J. MacIntosh and S. E. Conry, "SIMULACT: A generic tool for simulating distributed systems," *Tools for the Simulation Profession*, vol. 18, Apr. 1987.

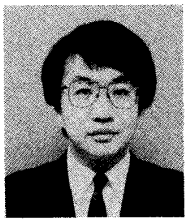
[14] A.K. Mackworth, "Constraint satisfaction" in *Encyclopedia of Artificial Intelligence*, S. C. Shapiro, Ed. New York: Wiley, 1987.

[15] R. P. Pope, R. A. Meyer, and S. E. Conry, "Role recognition in distributed planning," submitted for publication.

[16] R. G. Smith, "The contract net protocol: High level communication and control in a distributed problem solver," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, Dec. 1980.

[17] R. G. Smith and R. Davis, "Frameworks for Cooperation in Distributed Problem Solving," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-11, pp. 61–70, Jan. 1981.
[18] J. A. Stankovic "An application of bayesian decision theory to decentralized control of job scheduling," *IEEE Trans. Computers*, vol. C-34, pp. 117–130, Feb. 1985.
[19] K. Sycara, S. Roth, N. Sadeh, and M. Fox, "Decentralized factory scheduling: Coordinating resource allocation using constrained directed search," in *Proc. Tenth Workshop on Artificial Intell.*, Bandera, TX, 1990.
[20] A. S. Tanenbaum, *Computer Networks* Englewood Cliffs, NJ: Prentice-Hall, 1981.
[21] S. Vinter, K. Ramamritham, and D. Stemple, "Recoverable communicating actions in Gutemberg," *Proc. Int. Conf. Distributed Computing Syst.*, May 1986.
[22] M. Yokoo, T. Ishida, and K. Kuwabara, "Distributed constraint satisfaction for DAI problems," in *Proc. Tenth Workshop on Artificial Intell.*, Bandera, TX, 1990.

S. E. Conry (M'77), for a photograph and biography, please see page 1316 of this TRANSACTIONS.



Kazuhiro Kuwabara received the B.E. and M.E. degrees in electrical engineering from the University of Tokyo, Japan, in 1982 and 1984 respectively. In 1984 he joined Nippon Telegraph and Telephone Corporation (NTT) and has been engaged in research and development on knowledge-based systems.

He was a Visiting Research Scientist at the University of Massachusetts at Amherst for one year from September 1988. Currently he is a Senior Research Engineer in NTT Communications and Information Processing Laboratories.

Victor R. Lesser (A'80), for a photograph and biography, please see page 1183 of the September/October issue of this TRANSACTIONS.

R. A. Meyer (S'69–M'70–S'71–M'72), for a photograph and biography, please see page 1316 of this TRANSACTIONS.