Agent Cooperation Can Compensate For Agent Ignorance In Constraint Satisfaction

Peggy S. Eaton and Eugene C. Freuder
Computer Science Department
University of New Hampshire
Durham, New Hampshire 03824
pse, ecf@cs.unh.edu

Abstract

A team of constraint agents with diverse viewpoints can find a solution to a constraint satisfaction problem (CSP) when the individual agents have an incomplete view of the problem. In this paper we present a method of solving constraint satisfaction problems (CSPs) using cooperating constraint agents where each agent has a different representation of a particular CSP. Agents assist one another by exchanging information obtained during preprocessing and as a result improve problem solving efficiency. Unlike previous distributed and multiagent CSP techniques this agent-oriented technique provides fault-tolerance and redundancy. The technique is illustrated using cooperating constraint agents solving logic puzzles.

Introduction

A team of cooperating constraint-based reasoning agents may be able to solve a constraint satisfaction problem (CSP) even when members of the team have underconstrained representations of the problem. Agents with diverse representations can compensate for incomplete knowledge by sharing information obtained during problem solving.

Constraint satisfaction problem solving techniques can be applied to a variety of problems, including resource allocation, scheduling, planning, and network management. A constraint satisfaction problem (CSP) consists of a set of variables, a domain of values for each variable, and a set of constraints among the variables. A consistent labeling is a solution to the CSP where each variable is assigned a value so that the constraints among the variables are satisfied. A representation of a CSP is underconstrained if it is missing problem constraints. Underconstrained represention are problematic because spurious solutions may be generated in addition to the set of actual solutions. A constraint-based reasoning agent cannot determine which solutions are accurate because of incomplete knowledge.

Each agent in a team of cooperating constraint agents is a constraint-based reasoner with a constraint engine, a representation of the CSP, and a coordination mechanism. Cooperative problem solving agents may solve problems on their own and cooperate with group members when another agent possesses special expertise that can benefit the group, or because an agent has redundant capabilities that increase the reliability of the group, or when the task is simply too large for one agent. A group-level view of problem solving is a way to gain an advantage in managing ever more complicated problems (Jr. & Christensen 1993). Cooperation among constraint agents can compensate for incomplete knowledge and permit a solution because the agents have different representations of a constraint satisfaction problem (CSP). Although communication may be complex because of the diverse representations, cooperating agents with different views may find a solution more efficiently than a single agent.

Constraint satisfaction problems can have many alternate representations. Specifically, the views of the CSP may differ in the choice items selected as variables and items selected as values. For example, there are many ways to represent the classic n-Queens problem where n queens are to be placed on an n x n chessboard in non-attack positions. In the standard CSP representation of the n-queens problem, the rows of the board are selected as variables and the columns as values; (Nadel 1990) shows eight other representations of the n-queens problem.

We present a method of solving CSPs using cooperating constraint agents where each agent has a different representation of a particular CSP. Agents with different, incomplete views of the CSP share information obtained during preprocessing. Cooperation can compensate for incomplete knowledge. The technique is illustrated using cooperating constraint agents solving logic puzzles.

Differing Views

Cooperating agents with diverse representations may be necessary to solve CSPs when constraints are missing.

Different representations may be useful or necessary when:

- An engineer is uncertain as to the best represen-

tation for a particular problem, many agents with alternate representations increases the likelihood of using a good representation.

- Agents exchanging information during problem solving may permit the group to find a solution when no single agent could find a solution.
- An engineer can express some constraints as binary constraints in one representation but not another. The engineer prefers binary constraints since there are many standard binary constraint algorithms that can be used to solve binary CSPs.
- Cooperation among agents with different views may improve the efficiency of problem solving by reducing search effort.
- Although robustness is not guaranteed, the use of multiple agents provides fault-tolerance.
- Agents associated with different companies may be willing to cooperate to solve problems but not willing to share proprietary knowledge representations.

Incomplete CSP representations may occur when:

- Constraints are inaccurately entered into the system.
- An engineer misses constraints in the problem.
- An engineer purposefully leaves out constraints that cannot be easily expressed without using higher order constraints.

Cooperating constraint agents with diverse viewpoints try to compensate for incomplete knowledge by sharing information obtained during problem solving. This agent-oriented technique uses the exchange of partial information, rather than the exchange or comparison of an entire CSP representation when constraints are missing. Agents avoid the complete exchange of representations since agents developed by different companies might contain proprietary knowledge.

(Geelen 1992) presents the dual viewpoint concept for permutation problems (PP-CSP). PP-CSPs are a special class of CSPs where the domains of each variable are the same and the number of variables is equal to the cardinality of the domain. Each variable must be assigned a unique value and the dual view is that each value must be assigned to a unique variable. (Geelen 1992) notes that resource allocation problems can be represented as PP-CSPs. The problem solving technique developed by (Geelen 1992) incorporates heuristics to alternate between these two representations during problem solving.

Logic puzzle problems are an interesting class of problems that appear as entertainment in puzzle books and in some form on the LSAT and GRE exams. Consider the following logic puzzle from *Dell Logic Puzzles*:

> The Flumm Four is a new punk rock group from Liverpuddle. The four musicians (Furble, Icky, Muke, and Slyme) play bass, drums, guitar, and keyboard, not necessarily in that order. They've taken the music world by storm with four hit songs from their first album. Each of these songs - *Grossery Store*, *Heartburn Hotel*, *Love is Garbage*, and *2 Yuk 4 Words* - was written by a different member of the band. Can you match each musician with his instrument and his song?
>
> 1. Icky and Muke are, in one order or the other, the composer of *2 Yuk 4 Words* and the keyboardist
> 2. Neither Slyme nor the composer of *Heartburn Hotel* plays guitar or bass.
> 3. Neither the bass player (who isn't Icky) nor the drummer wrote *Love is Garbage*

Figure 1 shows three ways of representing the Flumm Four logic puzzle problem as a CSP using a constraint network; all constraints in the network are inequality constraints. A constraint network can be used to represent a CSP; the variables are the nodes of the network and the edges of the network are the constraints among the variables.

Logic puzzle CSPs are a more general class of problems than the PP-CSPs described by (Geelen 1992). The PP-CSP constraint network is a single clique. A clique is a constraint network where each node in the network is connected to every other node in the network. The logic puzzle constraint network is composed of a collection of cliques interconnected by constraints. In each clique every value is assigned to one variable and each variable is assigned a unique value from the domain of values. The variables each have the same domain and the size of each clique is equal to the cardinality of the domains. Several representations of a logic puzzle CSP can be easily generated since the variables associated with any clique can be a set of domains for the values in an alternate representation. Logic puzzle CSPs can be viewed as representative of resource allocation problems such as those described by (Geelen 1992) and (Yokoo, Ishida, & Kuwabara 1992).

Sharing Preprocessing Information

Figure 1 shows the representations for a team of agents cooperating to solve the Flumm Four logic puzzle problem. Agents communicate the results of running an arc consistency. Arc consistency (AC) is a preprocessing constraint inference method used to reduce the search effort. The domain size of a variable may be reduced during preprocessing because AC identifies values inconsistent with solutions. Agents share the inconsistent values generated during preprocessing.

Each constraint satisfaction agent uses the problem solving and communication algorithms described below. AC-3, a simple arc consistency algorithm, and a simple search algorithm are combined with fixed organization of the constraint satisfaction agents. The agents continue to send preprocessing results and then process messages received from other agents until there are no changes in the domains, finally the agents search for a solution.

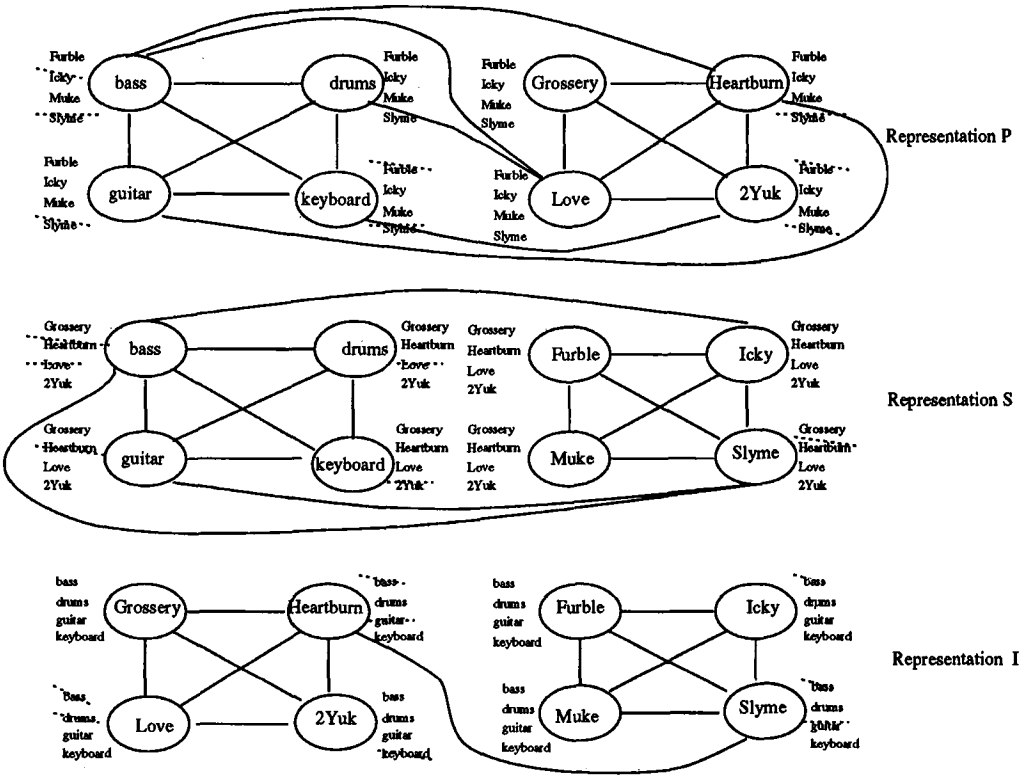

Figure 1: Multiple representations of the Flumm-4 logic puzzle.

Initially, we select a fixed, decentralized organizational structure where agents can communicate with all other agents. A message passing communication model is used rather than a shared memory model since the message passing model allows easy geographic distribution of agents and provides the system with natural redundancy.

Each member of the team runs the following algorithm:

```
csa-1 algorithm (constraint agent)
initialize constraint network
loop
  run arc consistency preprocessing
  if any domain changes then
     send changes to other agents
  if receive(messages)
     propagate message
until no domain changes
if there is a consistent labeling
     print solution
else
     search
```

A single transmission may contain a list of messages in the following format: (message-1, message-2, ..., message-j) and each message is a triple (op item-a item-b) where op is either an equality or inequality operator denoted by == or != respectively. A message is either a unary constraint assigning a value to a variable or a binary constraint between two variables. An agent receiving the message must determine how to add this information to its own knowledge base since the sending agent does not know the representations used by other agents.

```
receiving algorithm (message-list)
for each message
  if item-a and item-b are both variables
    add a constraint of type op
    between item-a and item-b
  else if item-a is a domain value
  if op is ==
       assign variable item-b
       the value of item-a
      else
       remove item-a from
       the domain of item-b
  else ;; item-b is a domain value
      if op is ==
       assign variable item-a
       the value of item-b
      else
       remove item-b from
       the domain of item-a
```

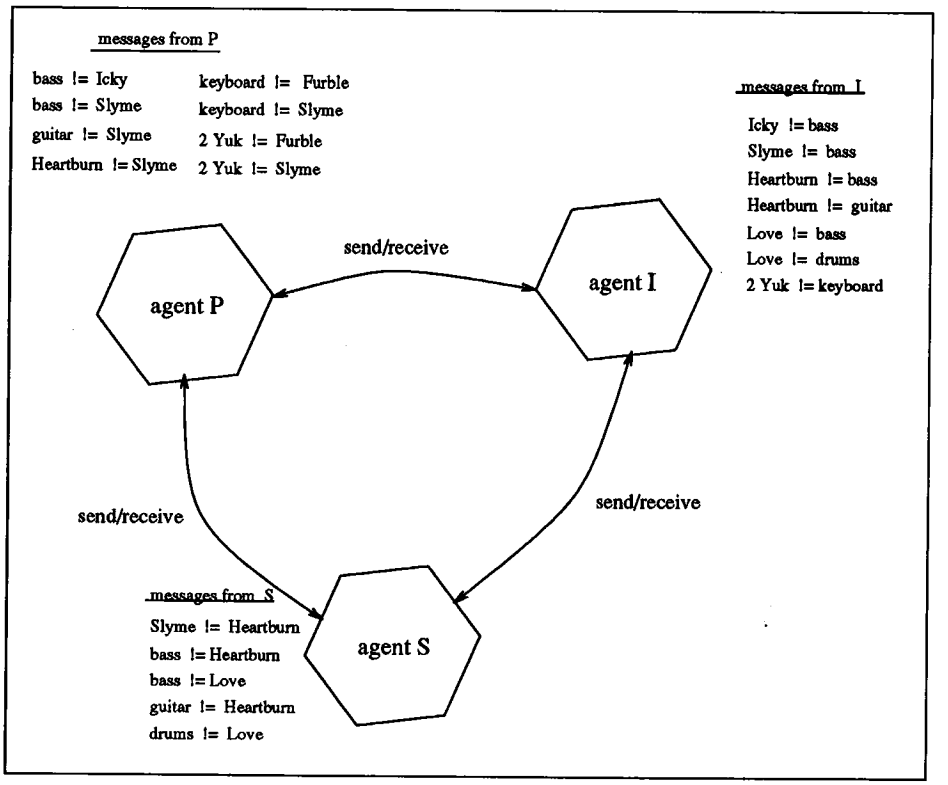

Figure 2: Messages passed after preprocessing.

Compensating For Missing Constraints

Figure 1 shows the representations for a team of agents cooperating to solve the Flumm Four logic puzzle problem. As mentioned in section 2, CSP representations may be missing constraints for a variety of reasons, including constraints that are purposefully left out of the representation. Representations I and S are both underconstrained because rule 1 cannot be represented using a standard binary constraint between the two variables. Without this additional information agents I and S cannot individually solve the problem. Agent P can solve the problem alone using search since rule 1 can be represented using binary constraints in representation P. In representation P we can write binary constraints indicating that the variables *keyboard* and *2 Yuk* can be assigned one of two domain values, either *Icky* or *Muke*. In representations S and I this information is represented as one of two constraints, which is difficult to handle without specialized CSP techniques.

Representation P in figure 1 can be modified so the representation of the agent P is also underconstrained. For example, representation P can be underconstrained by arbitrarily removing a constraint interconnecting the two cliques. As listed in section 2, a representation may be underconstrained for a variety of reasons, consider the situation where an engineer makes a mistake when creating the representation of the problem for agent P. The engineer doesn't enter the constraint between *Heartburn* and *guitar*. Agent P will generate too many solutions to the CSP. However, the team of agents can cooperate and permit a solution. Figure 2 shows the messages exchanged. *Heartburn* != *guitar* is represented as a constraint in agent P but in agent I it is a unary constraint. Figure reffig:flumm-under shows the messages received by agent I when the representations of all agents are underconstrained. After receiving the message agent I performs consistency preprocessing and a solution is found. Cooperation can compensate for incomplete knowledge even when all agents have incomplete information. Removing each of the other constraints connecting the cliques in representation P is similar to this example. Each of the binary constraints connecting the cliques in representation P corresponds to a unary constraint in one of the other representations, so the missing information would be passed back to agent P during the first round of messages.

Related Work and Future Directions

Previous distributed constraint processing techniques divide the variables of a CSP among agents (Yokoo, Ishida, & Kuwabara 1992). (Yokoo, Ishida, & Kuwabara 1992) defines distributed constraint satisfaction problems (DCSPs) and shows that distributed artificial intelligence problems can be represented as DCSPs. The CSP is distributed by placing a vari-

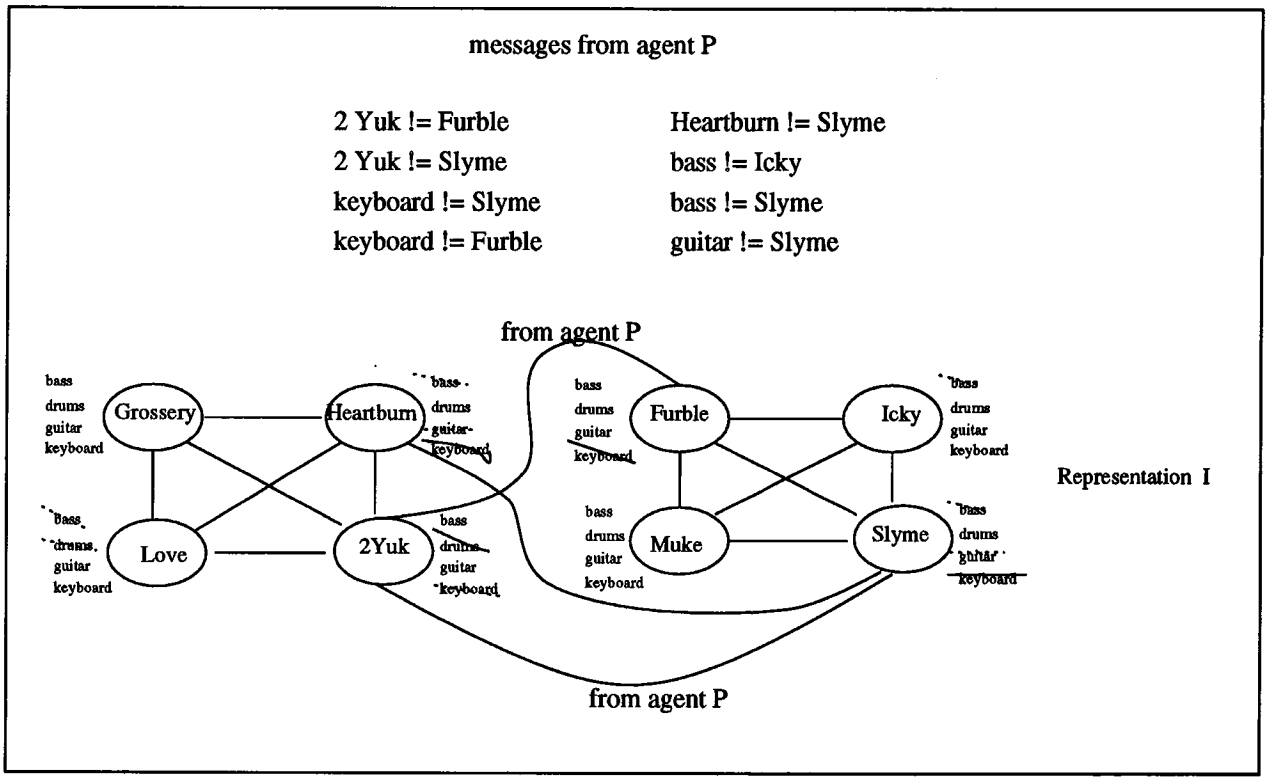

Figure 3: Messages received by agent *I*.

able (and associated domains) at each agent. The constraints of the CSP are distributed among the agents by associating with each agent only the constraints related to its own variable. (Yokoo, Ishida, & Kuwabara 1992) develops an asynchronous backtrack algorithm that allows the agents to work concurrently. Agents exchange locally inconsistent values with each other as search proceeds. (Yokoo, Ishida, & Kuwabara 1992) notes that applying consistency preprocessing algorithms in a DCSP is difficult because the algorithms require synchronization among the agents. If one of the distributed agents is lost, due to a fault in the system, the remaining agents can only produce a partial solution to the DCSP since a variable of the problem has been lost. When a DCSP is underconstrained, agents ignore each other because a constraint between agents is missing. So, DCSP agents cannot handle underconstrained problems.

Related multiagent constraint processing techniques partition the CSP so that separate pieces of the problem are solved and then synthesized. A 'think globally, act locally' approach to distributed meeting scheduling is presented by (Liu & Sycara 1994). Agents exchange meeting scheduling constraints to create a global representation of the problem. Agents agree or disagree to schedule a meeting using their own preferences but they do so given a global representation of the problem and so they can relax local constraints that may not be important given a particular meeting. The focus of the work is handling over-constrained scheduling problems using local information. If a constraint is left out of the problem an agents could be booked for several meetings.

(Nadel 1990) shows that CSPs can be represented in many different ways and demonstrates the difficulty in selecting the best representation. Theoretical complexity expressions are developed to guide the selection of a good representation from among many. Even though this work may assist in the selection of representations that perform well computationally, the work assumes the representation of the CSP is correct and the representation contains all problem the constraints.

(Cheng, Lee, & Wu 1996) represents the exchange of information between dual models of a CSP using constraints that are not related to the actual problem that is being solved. The models are different representations of the same problem and improve problem solving by providing redundant information that is shared.

Future directions for investigating constraint agents with different viewpoints:

- How robust is sharing AC preprocessing information as more information is deleted from the different views?
- Consider whether constraint agents can handle incorrect, inconsistent information. For example when

noisy or erroneous data is present.

- Consider negotiation strategies among the constraint agents when there are competing priorities.
- Explore efficiency issues associated with a team of agents solving large problems as compared to a single solver.
- Is there other information that can be helpful when shared? More complex and efficient consistency algorithms may produce a better set of information to share.
- Consider the use of constraint agents in solving dynamic CSPs.

Acknowledgements: This material is based on work supported by the National Science Foundation under Grant No. IRI-9504316. We thank Nancy Schuster and Dell Magazines for permission to reproduce material from *Dell Logic Puzzles.*

References

Cheng, B.; Lee, J.; and Wu, J. 1996. Speeding up constraint propagation by redundant modeling. In *Proceedings of Second International Conference on Principles and Practice of Constraint Programming (CP96).* To appear.

Geelen, P. A. 1992. Dual viewpoint heuristics for binary constraint satisfaction problems. In *ECAI92, 10th European Conference on Artificial Intelligence.*

Jr., J. R. L., and Christensen, C. 1993. Groups as problem-solving units: Toward a new meaning of social cognition. *British Journal of Social Psychology* 32:5–30.

Liu, J., and Sycara, K. P. 1994. Distributed meeting scheduling. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.*

Nadel, B. A. 1990. Representation selection for constraint satisfaction: a case study using n-queens. *IEEE Expert* 16–23.

Yokoo, M.; Ishida, T.; and Kuwabara, K. 1992. Distributed constraint satisfaction for DAI problems. In *ECAI92, 10th European Conference on Artificial Intelligence.*