Article

Distributed conflict resolution among cooperating expert systems

Faruk Polat
Shashi Shekhar

University of Minnesota, Computer Science Department, Minneapolis, MN 55455, USA

H. Altay Guvenir

Bilkent University, Department of Computer Engineering and Information Science, Bilkent, 06533 Ankara, Turkey

Abstract: *Cooperating experts approach attempts to integrate and coordinate the activities of multiple specialised problem solvers that come together to solve complex tasks such as design, medical diagnosis, business management and so on. Due to the different goals, knowledge and viewpoints of agents, conflicts may arise at any phase of the problem-solving process. Managing diverse expertise requires well-organised models of conflict resolution. In this paper, a model for cooperating experts is described which openly supports multi-agent conflict detection and resolution. The model is based on the idea that each agent has its own conflict knowledge which is separated from its domain level knowledge, and each agent has its own conflict resolution knowledge which is not accessible and known by others. Furthermore, there are no globally known conflict resolution strategies. Each agent involved in a conflict chooses a resolution scheme according to its self interest. The model is described by using an example in the domain of office design and it is compared with other systems.*

1. Introduction

Distributed Artificial Intelligence (DAI) is a subfield of AI which attempts to integrate existing problem-solving methods used in classical AI in order to develop systems that benefit from multiple agents' points of view. A solution developed by multiple agents incorporates aspects of each agent's problem-solving capabilities and perspectives rather than just the view of an individual agent's analysis of the problem (Cammarata *et al.* 1983; Chaib-Draa *et al.* 1992; Durfee *et al.* 1989; Gasser 1991; Malone & Crowston 1991; Polat & Guvenir 1991a; Smith & Davis 1988). One of the application domains of DAI is cooperating expert systems. The cooperating expert system approach is concerned with solving complex tasks which require diverse expertise to generate comprehensive solutions.

Applications of cooperating expert systems can be seen in human problem-solving tasks such as design, medical diagnosis, research, business management and human relations. Several systems reflecting the cooperating expert systems approach such as Hearsay-II (Erman *et al.* 1980), Contract Net (Smith & Davis 1988), Distributed Vehicle Monitoring Testbed (Lesser & Corkill 1988), MDX (Chandrasekeran 1982, Gomez & Chandrasekeran 1984) and Coop (Shekhar & Ramamoorthy 1992) are described in the literature. Managing diverse expertise is difficult because one has to take into account the problems which will arise in working out solutions in the face of conflicting goals, constraints, viewpoints and knowledge of heterogeneous experts.

In this paper, we describe a model in which a set of knowledge-based agents cooperates to solve design problems. The model is based on the resolution of conflicting solutions generated by experts having different goals, priorities and evaluation criteria. Existing approaches to conflict management (Adler *et al.* 1989; Klein & Lu 1989; Lander & Lesser 1990; Werkman *et al.* 1990) rely on coordinated resolution strategies which require resolution of a conflict based on a globally agreed strategy. In existing systems, conflict resolution knowledge is either maintained centrally or replicated by all agents. In any case, one of the disputants is given the power to take control of the conflict and use a resolution scheme known to everybody. In the approach described in this paper, however, agents are free to choose the most appropriate action, given their understanding of the global and local situations and their own capabilities. They maintain their own set of conflict resolution knowledge which is not globally known. Using their own conflict knowledge, the participants may reach an agreement on a revised solution.

There are several reasons why there is a need for resolving conflicts by means of agents' private conflict resolution knowledge instead of global knowledge about conflict resolution. First of all, this is much more similar to the resolution of conflicts that occur among human beings in solving complex

problem tasks in domains like design, diagnosis and business management. When a conflict is detected, it is not resolved by a central authority with global conflict resolution knowledge; rather, specialists involved in the conflict negotiate a revised solution that will be acceptable to all of them, using their own conflict resolution knowledge and perspectives. Second, forming global conflict resolution knowledge requires merging the conflict resolution knowledge of each agent obtained through knowledge acquisition in a consistent manner. This makes the maintenance of the global knowledge difficult because, when a new agent is added to or removed from the system, or the conflict resolution knowledge of an agent is revised, the global conflict resolution knowledge must be rebuilt accordingly. The last reason is related to the advantages of having distributed knowledge at all agents for reliability and fault-tolerance, instead of maintaining this knowledge in a centrally organised manner.

In Section 2, an overview of conflict management in cooperating expert systems is presented and the existing approaches are summarised. Section 3 describes a new model for cooperating experts and explains how the problem solving proceeds within this model. Section 4 describes how conflict resolution takes place in the new model. Section 5 includes a design example to illustrate how problem solving proceeds in this approach. Finally, Section 6 summarises the new model, emphasising its characteristics.

2. Conflict detection and resolution among cooperating experts

In the cooperating experts approach, several specialised agents combine to solve a common problem. During any phase of the problem-solving process, conflicts might appear as a result of incorrect and incomplete local knowledge, different goals, priorities and solution evaluation criteria. When there are several conflicting proposed solutions for a (sub)problem, the agents involved in a conflict must either agree to choose one proposal, cooperatively revise one, or search for a new solution that will be acceptable to everyone.

A common practice in building knowledge-based systems is to avoid potential conflicting situations by analysis and checking the consistency of the knowledge base at development time (Gingberg 1988; Nguyen *et al.* 1987; Polat & Guvenir 1991b; Rousset 1988; Vignollet & Ayel 1990). This approach, although effective, is very costly as the amount and diversity of knowledge increases. Resolving all conflicts, no matter how unlikely, at development time can be prohibitively time-consuming. Moreover, dividing the domain knowledge into smaller internally consistent collections is difficult.

The problems encountered when resolving conflicts in development time can be avoided by allowing conflicts to occur and be resolved at run-time. In other words, participating agents are allowed to generate conflicting solutions to the subproblems at run-time. In case of a conflict, a set of strategies could be used to resolve the conflict. Some examples of strategies include *backtracking*, *compromise negotiation* (a solution is iteratively revised by sliding a value or set of values along some dimension until a mutually acceptable middle point is found), *integrative negotiation* (identify the most important goals of each agent and find a solution which fulfils all of them), *constraint relaxation*, *case-based and utility reasoning methods*, etc. (Adler *et al.* 1989; Klein & Lu 1989; Lander & Lesser 1990; Polat & Guvenir 1992; Sycara 1989; Werkman *et al.* 1990). Work in this class comes closest to providing conflict resolution expertise with first class status.

Next, we will summarise studies which emphasise the use of conflict resolution within a cooperating expert systems paradigm. The first is conducted by Klein & Lu (1989), who propose a model for cooperative design that emphasises the parallel interaction of design agents. This work addresses the problem of how conflicts among different experts can be resolved. In their model, there are design experts and a particular conflict resolution expert. Given a design problem, design experts solve the subproblems relevant to their expertise. When a conflict is detected, the conflict resolution expert takes control and tries to resolve it. This expert maintains the global conflict management knowledge which contains conflict classes and corresponding resolution strategies.

The second work is introduced by Lander & Lesser (1990), who propose a *Cooperating Expert Framework* (CEF) to support cooperative problem-solving among sets of knowledge-based systems. The participating agents solve subproblems relevant to their specific expertise and integrate their efforts using conflict resolution strategies that are appropriate to the problem-solving context. All of the agents have a global knowledge of conflict resolution strategies. When a conflict is detected, agents involved in the conflict propose their alternative resolution strategies. Eventually they agree on a resolution scheme. Later, the conflict is resolved by one of the chosen agents based on that scheme.

Werkman *et al.* (1990) developed a system called *Design Fabricator Interpreter* (DFI) which is a framework for distributed cooperative problem-solving among construction agents. The DFI system reflects the distributed nature of the construction industry by providing a multi-agent architecture that models design, fabrication and erection processes. Conflicting recommendations issued by design agents are resolved by a *third-party arbitrator* agent, which makes suggestions based on the globally known conflict resolution knowledge. It operates in both passive and active mode. In *passive* mode, the arbitrator monitors the agent proposal process and intercedes when a problem is evident; in *active* mode, it mediates during the agent's proposal process when called upon by the agents.

Adler *et al.* (1989) discuss methods of conflict resolution in the domain of telephone network traffic control. A homogeneous group of agents has geographically divided responsibilities with no overlap. The basic problem that the agents are to solve is excessive demand for the resources in some parts of

the network. Two negotiation protocols are described: *conflict-driven plan merging*, a bottom-up approach to resolving a conflict that has already occurred, and *shared plan development*, a top-down approach to avoiding conflicts as plans are developed and refined. Their research addresses how conflicts on the usage of resources could be resolved.

3. A distributed conflict resolution-based model for cooperating experts

The cooperating experts environment is organised as a community of cooperating problem-solving agents, where each agent is represented as a fully functional and autonomous knowledge-based system. The model is designed for solving problems in the domain of design. This model is based on the idea that each design agent has its own conflict resolution expertise separate from its domain-level design expertise, and that in the context of particular conflicts this expertise can be instantiated into specific advice for resolving these conflicts. The model allows a new problem-solver to be added or an existing one to be removed without requiring any modification to the rest of the system. The model can therefore be considered to achieve Open Systems Semantics (Open Systems deal with large quantities of diverse information and exploit massive parallelism) (Hewitt 1986, 1991) in the sense that it not only allows scaleability (ability to increase scale of commitments) but also robustness (ability to keep commitments in face of conflicts) — two primary indicators of Open Systems Semantics.

3.1. Architecture of the model

The cooperative design environment (Figure 1) is composed of a set of design agents, which are fully functional knowledge-based systems, and a shared blackboard. The agents communicate by posting assertions in a shared language. This requires translation capabilities to be included within the agents. The shared blackboard is a public repository available to all agents; this gives one the ability to store 'global' information, although the information can only be used locally by the agents. Alternatively, it would be possible to convey information directly through point-to-point communication channels or reserved-spot communication (Winston 1984). The shared blackboard is partitioned into four chunks, allowing fast access, delete and update operations of units. They are called *problem*, *solution*, *proposal* and *conflict* areas.

The problem area of the shared blackboard contains the initial problem definition and overall requirements that must be taken into account by the design agents. A problem instance is a tuple of the form $P = <O, G, C, I>$, where O denotes the problem originator (an agent that defines the problem to be solved); G is the set of goals that must be satisfied for a design to be accepted; C is the set of constraints that design agents should not violate (some of the constraints may be violated through negotiation with the problem originator); and I denotes the initial problem information (such as the layout of a room if the problem is to design an office). The solution area of the shared blackboard includes the evolving design template to which non-conflicting design commitments produced by agents are added.

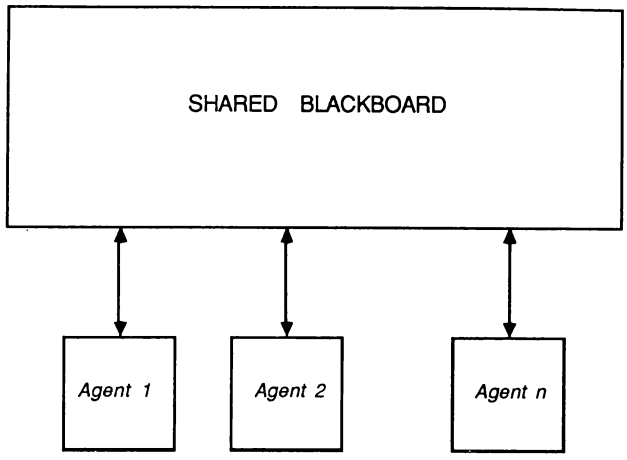


Figure 1: *The architecture of the problem-solving environment.*

The proposal area includes partial and incomplete solutions at several layers of abstraction issued by design agents. Design agents assert their solutions as proposals into this area. A proposal instance is a tuple of the form $Q = <O, A, R, C_f>$, where O denotes the owner of the proposal; A is the set of proposed actions to update the current design template; R consists of the reasons justifying each of the actions in A (R could be empty because an agent may not provide reasons for its proposal due to its inaccurate or incomplete knowledge); and C_f is the confidence factor that indicates the confidence of the owner in generating such a proposal. This information would be useful for other agents if the owner utilised inaccurate or incomplete knowledge in producing its solution.

Conflict area is the place where agents put their critiques related to a new design commitment. A portion of this area provides a communication medium with agents that are involved in a conflict situation. This area holds evaluation results and conflict resolution recommendations issued by design agents. An evaluation result instance is a tuple of the form $ER = <O, Q, A_c, R_a, R_e>$, where O denotes the owner of the evaluation result tuple; Q is the identity of the proposal evaluated; A_c is the set of actions criticised ($A_c \subseteq A$ in Q); R_a is the set of ratings (evaluation results) for each action in A_c; and R_e is the overall result of the proposal Q which is either 'conflicting proposal' or 'nonconflicting proposal'. A conflict resolution instance is a tuple of the form $CR = <O, Q, A_r, R_r>$, where O is the owner of the conflict resolution tuple; Q is the identity of the related proposal; A_r is the set of refinements for those conflicting actions of Q; and R_r includes the reasons for the proposed refinements.

A description of the internal structure of an agent in the model is given in Figure 2. An agent supports a knowledge base, a database and a general controller. The knowledge base

includes domain and control knowledge, just like in a classical knowledge-based system. It also contains conflict resolution knowledge to be used in cooperatively managing conflicts with other agents. This knowledge is not known globally and varies with respect to the agent's beliefs and understanding of the environment. The database includes facts, goals and constraints specific to the domain of that agent. Agents also maintain two types of history information related to the solution generation phase and conflict resolution phase. This not only makes backtracking possible but also allows case-based information to be used later for solving similar problems encountered. The general controller includes procedures for generating and evaluating design commitments, managing conflicts and translating messages into the common language.

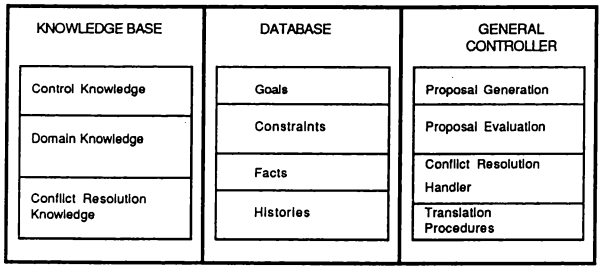


Figure 2: *Internal structure of a design agent in the model.*

The agents in the model are actually heterogeneous agents in the sense that they might use different knowledge representation techniques and inference mechanisms. Each agent is assumed to generate proposals (solutions for subproblems) according to its knowledge. They cooperate to achieve the common goal of solving the global problem. The model can be augmented to support special agents like database systems. For the time being, we assume that each agent is a knowledge-based system which offers to solve subproblems (produce proposals) and cooperates with others in resolving conflicts through negotiation. Local knowledge is represented in whatever language desired and cannot be accessed by any agent except its owner. Several knowledge representation techniques have been developed for the domain of design (Akman *et al.* 1990; Gero 1988; Goel & Pirolli 1989; Smithers & Troxell 1990; Sriram & Adey 1986; Tokoro & Ishkawa 1984). If the internal language is not the same as the shared language, translation procedures are incorporated within agents. In case of a conflict, agents might make some or all of their goals, constraints and even knowledge available to others.

3.2. Problem-solving phases

The problem-solving in the model is initiated by one of the agents asserting a problem definition $P = <O, G, C, I>$ into the problem area of the shared blackboard. All interested agents are instantiated after examining the problem definition and they start producing design proposals related to their expertise, knowledge and viewpoints. When a design agent generates a design proposal $Q = <O, A, R, C_f>$, it is put into the proposal area. The agent producing this proposal also includes explanation information that indicates which of the agent's goals and constraints caused such a proposal. This explanation allows other agents to understand why such a proposal has been asserted.

After the generation of a proposal, all of the agents are signalled. An agent does not interrupt its proposal generation process if it is already working on another proposal, but it immediately awakens another process — the evaluation process — that will run in parallel with the proposal generation process. The evaluation process first informs the owner of the proposal whether it is going to criticise the proposal. If not, it will go to sleep and wait for another proposal to be asserted. If the agent is interested in the proposal, it evaluates and posts the result $ER = <O, Q, A_c, R_a, R_e>$ on the conflict area of the blackboard. The owner of the proposal also evaluates its proposal, usually as part of the solution generation process. It is necessary for the owner to indicate its confidence because it might use incomplete or inaccurate knowledge in producing its solution. The evaluation process results in a rating to be produced which shows the 'quality' of a solution with respect to the goal criteria used to judge. The agents use their internal evaluation criteria and therefore may not share a common rating scale for their findings.

After all the interested parties finish evaluating the newly asserted proposal, those agents which identify the proposal under consideration as conflicting with their beliefs come together to resolve the conflict (those interested agents that put *ER*'s in which R_e part is 'conflicting proposal'). Agents in our model do not have a global knowledge of conflict resolution strategies, though in existing systems agents are assumed to be knowledgeable about the global conflict resolution expertise. In our model, each agent has its own conflict resolution knowledge that allows it to participate in the process of conflict resolution. The result of conflict resolution is either revision or abandonment of the proposed solution.

When none of the interested agents detect any conflict related to a proposal, the partial design template residing in the solution area is updated by using the design contribution existing in the proposal. The process continues until the design template meets requirements specified by the agent that put the initial problem definition in the problem area of the shared blackboard. The design process may also be terminated, although the agent that put the problem definition is not satisfied. This may happen in cases where none of the agents can generate a non-conflicting design proposal any more.

4. Conflict resolution in the model

When an agent detects a conflict, it participates in the resolution process based on its own conflict resolution knowledge. Each agent may utilise different conflict resolution strategies. For example, suppose that we are given the problem of de-

signing an office. The functionality agent suggests that the PC desk be put close to the window so that a PC user could have a look outside when bored. On the other hand, the computer specialist, detecting a conflict, argues that sunshine could damage the PC. The computer specialist uses a conflict resolution strategy which says 'put electrical devices far away from windows'. The functionality agent, however, uses a domain-independent resolution scheme 'try other subgoal alternatives'. Eventually two experts revise the proposal such that the PC desk is put into a place in the office which is not exposed to sunshine, by using different resolution schemes. In deciding which strategy to apply, an agent uses information gathered up to the time the conflict has occurred as well as its conflict knowledge. This information includes:

- Explanation embodied within the proposal (this will allow other interested agents to understand the intent of such a proposal).
- Critiques made by the interested parties to the proposal (after examining outcomes of evaluation procedures of other agents, an agent chooses an appropriate resolution strategy taking into account different viewpoints).
- The relevance of the agent to a particular problem being solved (if an agent is more knowledgeable and capable compared to others, it should participate in resolution of a conflict according to its relevance).
- Flexibility of agents involved in conflicts (this is important for an agent to decide how to behave in a compromise type of conflict resolution).
- Behaviour and actions of other agents in resolving the conflict (by examining this information, an agent might decide to alter the conflict resolution strategy it has been using).
- Conflict resolution history information (if a similar conflict situation was encountered beforehand, an agent could utilise history information in resolving the conflict. This allows case-based reasoning in conflict resolution).
- Number of agents involved in the conflict (depending on the domain, if the number of agents involved in a conflict situation exceeds a certain amount then some of the agents, thinking that they could not be effective for resolving the conflict compared to others, may continue to generate alternative solutions rather than participate in conflict resolution).
- Available problem-solving resources.

When a proposal is issued, each interested agent evaluates it to detect whether or not any conflicting recommendations exist in the proposal. For understanding the kind of conflict that has occurred, each agent examines its conflict knowledge to see which of its conflict situations match (not necessarily a perfect match) the current conflict (if existing). Upon deciding on the conflict situation, an agent uses its conflict resolution knowledge to overcome the conflict from its perspective. Conflict resolution knowledge is composed of general strategies (domain-independent) and specific strategies (domain-dependent). Domain dependent strategies are gathered during the knowledge acquisition phase. Agents prefer to use the most specific strategies first for resolving a conflict. General strategies are resorted to last since they are computationally expensive and may lead to poor solutions.

When an agent detects a conflict and chooses a strategy for resolving the conflict, it does not mean that the agent may not alter the resolution strategy it has chosen. That is, upon observing the actions of other agents during conflict resolution phase, it may improve its understanding of the overall problem and the particular conflict encountered. This allows agents to alter strategies if they think that they will benefit from doing so. When an agent proposes a revised solution based on its resolution scheme, it also explains why the new solution is a good candidate. This enables other agents involved in conflict to choose the most appropriate action on behalf of the resolution process.

5. A cooperating experts' problem: office design

The following example is taken from the domain of office design to exemplify the problem solving process of the cooperating experts that is used in our implementation. The reason for choosing this example is that it is in a concrete rather than an abstract domain and it can be understood easily because of its suitability for simple two-dimensional graphical representation. Here, we present a simplified layout problem for an office design and describe design agents and their interactions. A well-designed office encompasses different areas of expertise concerning aesthetics, functionality, energy efficiency, etc. In this example, we have incorporated four agents in the problem-solving framework. They are

- the client agent;
- the functionality agent;
- the electricity agent;
- the cost agent.

The client agent is the one that puts forth the problem definition specifying general constraints and the global design goal to be satisfied. This agent may be the one to use the office being designed or be the department chairman who is having the office designed for a prospective faculty member. The functionality agent uses specific heuristic search techniques in the area of space planning. The electricity agent is concerned with all the electrical and electronic devices and wiring including computers, telephones, facsimile systems, etc. The cost agent is required to control the overall cost of the design and avoid wasteful use of resources. When a proposal is generated, each interested agent evaluates it to detect a possible conflict from its own perspective. A conflict is detected when an agent finds a conflict situation (upon examining its knowledge base) that matches the proposal under consideration. In this domain, some possible conflict situations are *location of an object, dimensions of an object, quality of an*

object, cost of an object, usage of an object, existence of an object and so on.

The design process is initiated by the client agent who puts the following problem definition into the problem area of the shared blackboard:

```
Problem Definition =
 < Client-Agent,
   { goal: (design office)
     subgoals:  ((minimize amount-of-walking)
                 (customize components-to-the-
                  size-of-office)
                 (maximize efficiency)
                 (must-have PC) ...)) },
   { constraints:  ((to-be-used-by faculty-member)
                    (number-of-occupants 1)
                    (cost-of-design < $6000) ... ) },
   { layout:
     shape: rectangular
       dimensions : ((length 10) (width 8) (height 2.5))
       coordinates: (upperleft (x 0 y 0))
     window:
       type : ((frame wood) (glass glass1))
       dimensions: ((height 1))
       coordinates: ((x 0 y 3) (x 0 y 5))
     door:
       type: ((made-up-of wood))
       dimensions: ((height 2))
       coordinate: ((x 8 y 7) (x 8 y 8))
     electrical-plug:
       coordinates: ((x 6 y 0))
     phone-plug:
       coordinates: ((x 6.5 y 0))
     .
     .
   } >
```

Figure 3a shows the global layout of an office. In this example, we ignore the third (z) dimension; instead the height attribute of objects is used when necessary. Also, we are not concerned with the precise locations of objects. After examining the problem definition, all of the interested parties start producing design commitments. First, the functionality agent, according to its expertise and understanding of the problem, asserts the following proposal into the proposal area of the shared blackboard which updates the template as shown in Figure 3b.

```
Proposal-0 = < Functionality-Agent,
 { (put :object desk1    :type desk   :location (2 2)),
   (put :object chair1   :type chair  :location (3 0.5)),
   (put :object pcdesk1  :type pcdesk :location (2.5 6)),
   (put :object chair2   :type chair  :location (3 5)) },
 { (utilize sunshine)
   (have better-view) } )
 nil >
```

The functionality agent has decided to put a desk and a PC desk along with two chairs nearer to the window so that the occupant can not only have a good view but also utilise the

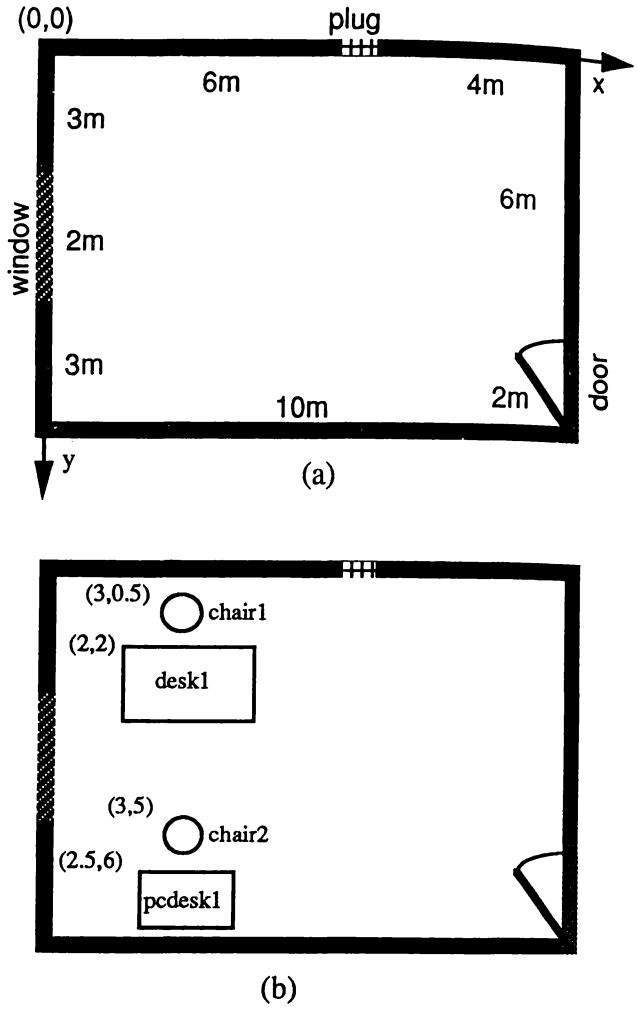


Figure 3: *(a) Global layout of the office; (b) Layout of the office after Proposal–0.*

sunshine. In generating this proposal the agent used the following piece of its domain knowledge:

```
if    number of occupants is 1 and
      the occupant of the room is academic person
then  activate SINGLE-ACADEMIC-KB

SINGLE-ACADEMIC-KB (only utilized piece of knowledge)

      find a place to put a desk (desk1)

if    the occupant is to use computer
then  include a computer desk (pcdesk1) and
      find a place to put the computer desk (pcdesk1)

if    a desk (desk1, pcdesk1) is to be put
then  put it near to the window
      reasons: utilize sunshine
               have a better view
```

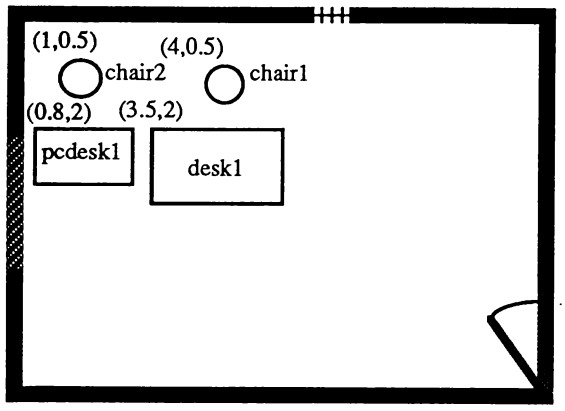
This proposal triggers the evaluation procedures within other interested agents. The client agent detects a conflict after evaluating the proposal. With this configuration, the client

agent notices that occupants must walk too much because they might need to use the PC (which will be put on the PC desk) a lot. The functionality and client agents combine to resolve the conflict encountered. The client agent uses a specific resolution scheme which states 'keep frequently used objects close to each other', and the functionality agent uses a general conflict resolution strategy which is 'try other location alternatives'. The client agent has two alternatives to resolve the conflict from its perspective; it may put the PC desk either to the left or to the right of the other desk. Taking into account the explanation within the functionality agent's proposal, the client agent proposes to put the PC desk to the left of the other desk so that the PC desk will be close to the window and hence the occupant can utilise sunshine and have a better view. The revised and agreed solution is shown in Figure 4a. Below is the description of how conflict is detected and resolved by both agents:

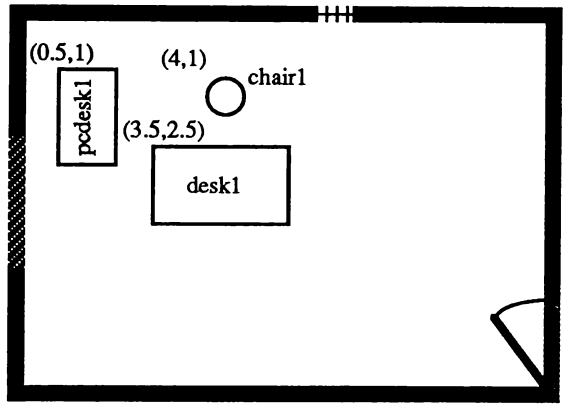
```
Conflict Detection Tuple =
 < Client-Agent, Proposal-0, { (all-actions) },
   { (increased amount-of-walking) },
   "Conflicting-Proposal" >

The Client Agent:
 Conflict Situation: Location of Objects
 Resolution        : If   the objects involved in
                          conflict are to be used
                          very often
                     then try to keep them close
                          to each other

The Functionality Agent:
 Conflict Situation: Location of Objects
 Resolution        : try other configuration
                          alternatives

The agreed resolution actions:
 (move :object desk1      :position (3.5 2))
 (move :object chair1     :position (4 0.5))
 (move :object pcdesk1    :position (0.8 2))
 (move :object chair2     :position (1 0.5))
```

Then the cost agent realises that there is no need to have two chairs close to each other. The cost, client and functionality agents decide to remove one of the chairs as shown in Figure 4b. In the resolution phase, agents agree on rotating the PC desk such that a chair could be used for both desks:

```
Conflict Detection Tuple =
 <Cost-Agent, Proposal-0',
 { (existence-of :object chair2) },
   { }, "Conflicting-Proposal" >

The Cost Agent:
 Conflict Situation: Cost of Objects
 Resolution        : remove them

The Client Agent:
 Conflict Situation: Existence of Objects
 Resolution        : If   existence of some objects
                          causes some problems
                     then remove them
```



(a)



(b)

Figure 4: *(a) Layout of the office after resolving the conflict in Proposal–0. (b) Layout of the office after resolving the conflict in Proposal–0'.*

```
The Functionality Agent:
 Conflict Situations :    Location of Objects -
                          Existence of Objects
 Resolution               try other configuration
                          alternatives
                          upon removing unwanted objects

The agreed resolution actions:
 (remove  :object chair2)
 (move    :object desk1   :location (3.5 2.5))
 (move    :object chair1  :location (4 1))
 (rotate  :object pcdesk1 -90)
 (move    :object pcdesk1 :location (0.5 1))
```

Later, the electricity agent decides to put the PC on the PC desk and use an extension cord to connect the PC to the plug, and puts a proposal related to these modifications into the proposal area (Figure 5a):

```
Proposal-1 = < Electricity-Agent,
               { (put :object pc1 :type ibmpc
               :location (0.7 1.5)),
                 (put :object cord1 :type cord :path
                   ((0.8 1.5) (0.8 0.5) (6 0.5) (6 0.5)) },
                 { },
                 nil >
```

```
The knowledge used to generate Proposal-1:

   if   there is a suitable place (pcdesk1) to put
        an electrical device (pc1) on top of
   then put the electrical device (pc1) on top of it
        (pcdesk1)

   if   an electrical device (pc1) is not connected to
        the plug and
        the electrical device's (pc1's) own cable
        is short
   then use an extra and long enough cord (cord1)
        to connect
        the electrical device (pc1) to the plug
```

The cost agent detects another conflict related to the electricity agent's proposal, which is the cost of using an extension cord. The electricity, cost and functionality agents are involved in the resolution of the conflict. The cost agent decides to remove the extension cord and is very inflexible in its decision. The functionality agent does not have a specific resolution scheme in its mind, but uses a general one and

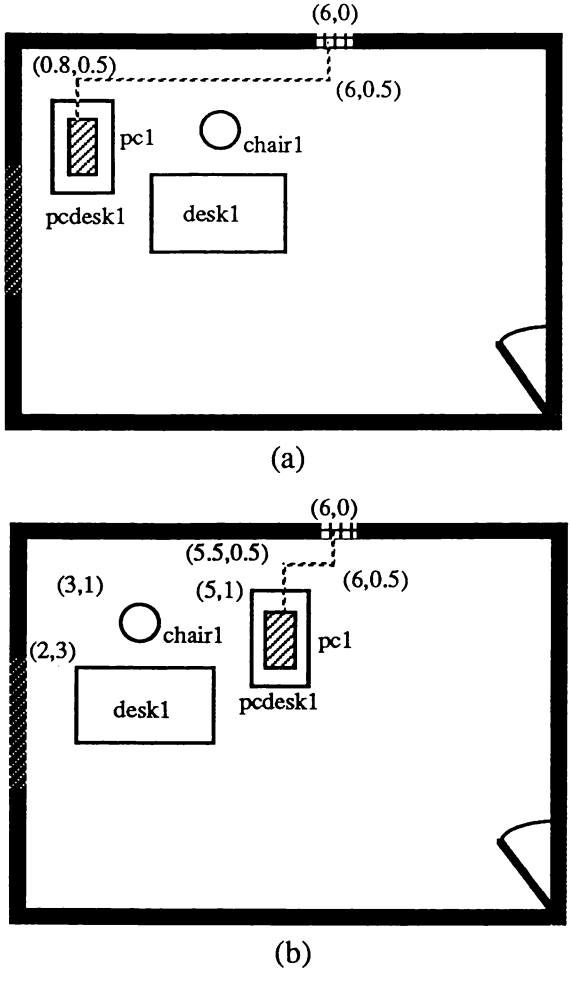


Figure 5: *(a) Layout of the office after Proposal–1. (b) Layout of the office after resolving the conflict in Proposal–1.*

decides to put PC desk to the right of the other desk. Although the functionality agent knows that it is better to keep the PC desk close to the window for reasons of utilising sunshine and having a better view, it does not insist on this preference for resolving the conflict. The electricity agent has a domain specific resolution strategy which states 'keep all of the electrical devices close to the plugs'. Below is how this particular conflict is detected and resolved:

```
Conflict Detection Tuple =
 < Cost-Agent, Proposal-1, { (cost-of :object cord1) },
   { }, "Conflicting-Proposal" >

The Cost Agent:
 Conflict Situation     Cost of an Object
 Resolution             - remove the object
                          (preferred resolution
                           alternative)
                        - if   some other object of the
                               same type but cheaper
                               exists
                        then   try to remove the
                               expensive one
                        - if   there exists some other
                               unnecessary or less
                               important object in
                               the template
                        then   try to remove that object

The Electricity Agent:
 Conflict Situations :  Location of Objects and
                               Existence of Objects

 Resolution             - try  other configuration
                               alternatives
                               upon removing unwanted
                               objects
                        - if   an electrical device
                               (pc1) is to be put
                        then   put the device (pc1)
                               close to  the plug
                               (preferred resolution
                               alternative)

The Functionality Agent:
 Conflict Situations :  Location of Objects and
                               Existence of Objects

 Resolution             try other configuration
                               alternatives

The agreed resolution actions:
 (remove  :object cord1)
 (move    :object desk1        :location (2 3))
 (move    :object chair1       :location (3 1))
 (move    :object pcdesk1      :location (5 1))
 (move    :object pc1          :location (5.5 1.5))
 (plug-in :object cord of pc1  :to plug)
```

By explaining their resolution steps the three agents come to an agreement to put the PC desk close to the plug, as shown in Figure 5b. Later, the electricity agent connects the PC's cable to the plug without using an extension cord. The design pro-

ceeds in this manner until reaching the requirements specified by the client agent.

In this example, we only gave a segment of the problem-solving process, emphasising the resolution of conflicts without considering precise locations of objects.

6. Conclusions

The cooperating experts' approach has an important role in the field of Distributed Artificial Intelligence because many of the problems that are being encountered in real life require the application of complex and diverse expertise. One of the important problems faced in a cooperating community of experts is how to detect and resolve conflicts occurring at any phase of problem solving. Existing approaches to conflict resolution rely on coordinated conflict resolution strategies (Adler *et al.* 1989; Klein & Lu 1989; Lander & Lesser 1990; Werkman *et al.* 1990). In these approaches, each agent is assumed to have a global knowledge of conflict resolution information. When conflicts happen, agents agree on a conflict resolution scheme and one agent resolves the conflict using a globally agreed resolution strategy.

In this paper, we introduce a new cooperating experts environment for solving problems that openly support multi-agent conflict detection and resolution. In this environment, each agent is free to choose the most appropriate action, given its understanding of the global and local situation and its own capabilities. Each agent has its own conflict resolution knowledge which is not accessible and known by others. Furthermore, there are no globally known conflict resolution strategies. Each agent involved in a conflict chooses a resolution scheme according to its self-interest. Agents might use different strategies of their own and might still agree on a solution.

The model achieves flexibility in its problem-solving, which is the most compelling argument for building modular multi-agent systems. A new agent can be added or an existing one can be removed without any modification of the rest of the system. This characteristic of the model satisfies the requirements of Open Systems Semantics. However, in the existing approaches, addition or removal of an agent requires that the global conflict resolution knowledge be reformed accordingly.

This approach is very similar to conflict resolution in human problem-solving. Existing approaches are too restrictive and applicable only to the problems where experts must agree on a known strategy for resolving conflicts. The new approach also allows agents to alter strategies in the resolution phase if they think it is wise to do so. The model also requires agents to explain why a particular action is made in order for other agents to reason about it. This means that agents may explain the reasoning behind their proposed actions in putting their proposals and in detecting and resolving conflicts.

Currently, we are implementing the model on interconnected SUN–4 workstations. All the problem-solvers — agents — are modelled as processes running on different workstations that communicate over Ethernet. In order to illustrate the problem-solving phases in the model, we have chosen to solve design problems. We will test the conflict resolution-based model on various other examples in the domain of design and medical diagnosis.

References

ADLER, M.R. *et al.* (1989) Conflict resolution strategies for non-hierarchical distributed agents, *Distributed Artificial Intelligence*, **2**, M.N. Huhns, pp. 139–161.

AKMAN, V., P.J.W. TEN HAGEN and T. TOMIYAMA (1990) A fundamental and theoretical framework for an intelligent CAD system, *Computer-Aided Design*, **22**(6), 352–367.

CAMMARATA, S., D. MCARTHUR and R. STEEB (1983) Strategies for cooperation in distributed problem solving, in *Proc. Int. Joint Conf. Artificial Intelligence*, Karlsruhe, Germany, August. pp. 767–770.

CHAIB-DRAA, B. *et al.* (1992) Trends in distributed artificial intelligence, *Artificial Intelligence Review*, No.6, pp. 35–66.

CHANDRASEKERAN, B. (1982) Decomposition of domain knowledge into knowledge source: The MDX approach, *Fourth National Conf. of Canadian Society for Computational Studies of Intelligence*, Canada.

DURFEE, H.D., V.R. LESSER and D.D. CORKILL (1989) Trends in cooperative distributed problem solving, *IEEE Transactions on Knowledge and Data Engineering*, **1**(1), March, 63–83.

ERMAN, L.D., F. HAYES-ROTH, V.R. LESSER and D.R. REDDY (1980) The Hearsay-II speech understanding systems: integrating knowledge to resolve uncertainty, *Computing Surveys*, **12**, June, 213–253.

GASSER, L., (1991) Social conceptions of knowledge and action: DAI foundations and open systems semantics, *Artificial Intelligence*, **47**, 107–138

GERO, J.S. (Ed.) (1988) *Artificial Intelligence in Engineering Design*, Elsevier, UK.

GINGBERG A. (1988) Knowledge base reduction: a new approach to checking knowledge bases for inconsistency and redundancy, *AAAI*.

GOEL, V. and P. PIROLLI (1989) Design within information-processing theory: the design problem space, *AI Magazine*, Spring, 19–36.

GOMEZ, F. and B. CHANDRASEKARAN (1984) Knowledge organization and distribution for medical diagnosis, *Technical Report*, 84–FG–FGBC, Department of Computer and Information Science, The Ohio State University.

HEWITT, C. (1986) Offices are open systems, *ACM Transactions on Office Information Systems*, **4**(3), 271–287.

HEWITT, C. (1991) Open information systems semantics for distributed artificial intelligence, *Artificial Intelligence*, **47**, 79–106.

KLEIN, M. and S.C-Y. LU (1989) Conflict resolution in cooperative design, *Artificial Intelligence in Engineering*, **4**(4), 168–180.

LANDER, S. and V.R. LESSER (1990) Conflict resolution strategies for cooperating expert agents, *International Conference on Cooperating Knowledge-Based Systems*, Keele University, October.

LESSER, V.R. and D.D. CORKILL, (1988) The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks, in R.S. Engelmore and A. Morgan (Eds.), *Blackboard Systems*, Addison-Wesley, pp. 353–386.

MALONE, T.W. and K. CROWSTON (1991) Towards an interdisciplinary theory of coordination, *Technical Report*, CCS–TR–120, Center for Coordination Science, MIT, Cambridge, MA.

NGUYEN, T.A. *et al.* (1987) Verifying consistency of production systems, in *Proceedings of the Third Conference on Artificial Intelligence Applications*, pp. 4–8.

POLAT, F. and H.A. GUVENIR (1991a) Coordination issues in distributed problem solving, in *Proceedings of the Sixth International*

Symposium on Computer and Information Sciences, Elsevier, Vol. 1, Antalya, pp. 585–594.

POLAT, F. and H.A. GUVENIR (1991b) A unification based approach for knowledge base verification, *Expert Systems*, **8**(4), 251–259.

POLAT, F. and H.A. GUVENIR (1992) A conflict resolution based cooperative distributed problem solving model, in *Proceedings of AAAI–92 Workshop on Cooperation among Heterogeneous Intelligent Agents*, San Jose, CA, July.

ROUSSET, M.C. (1988) On the consistency of knowledge bases, in *Proceedings of European Conference on Artificial Intelligence*.

SHEKHAR, S. and C.V. RAMAMOORTHY (1992) Coop: a self-assessment based approach to cooperating expert systems, *International Journal on Artificial Intelligence Tools*, **1**(2), 175–204.

SMITH, R.G. and R. DAVIS (1988) Frameworks for cooperation in distributed problem solving, in A.H. Bond and L. Gasser (Eds.), *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, pp. 61–70.

SMITHERS, T. and W. TROXELL (1990) Design is intelligent behaviour but what's the formalism? *AI EDAM*, **4**(2), 89–98.

SRIRAM, D. and R. ADEY (1986) *Applications of Artificial Intelligence in Engineering Problems*, Vols. 1–2, Springer-Verlag.

SYCARA, K.P. (1989) Negotiation in design, in *Proceedings of the MIT-JSME Workshop on Cooperative Product Development*, MIT, Cambridge, MA.

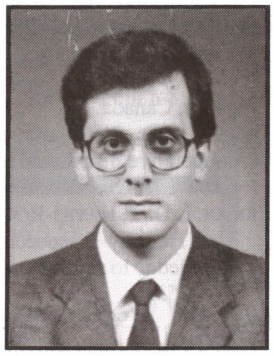
TOKORO, M. and Y. ISHKAWA (1984) An object-oriented approach to knowledge systems, in *Proc. of the Int. Conf. on Fifth Generation Computer Systems*, pp. 623–631.

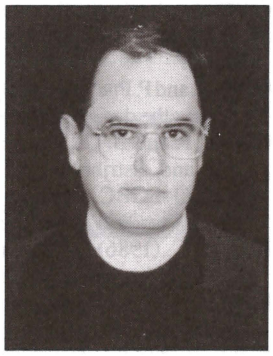
VIGNOLLET, L. and M. AYEL (1990) SYCOJECT: A tool for building automatically sets of test for knowledge bases, *Applications of Artificial Intelligence*, No. 8, pp. 192–201.

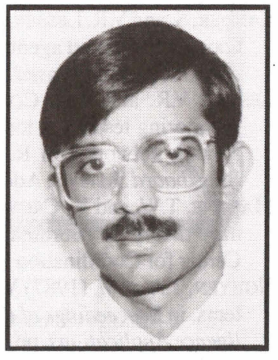
WERKMAN K. *et al.* (1990) Design and fabrication problem solving through cooperative agents, NSF–ERC–ATLSS *Technical Report* 90–05, Lehigh University, Bethlehem, PA.

WINSTON, P.H. (1984) *Artificial Intelligence*, Addison-Wesley.

The authors







Faruk Polat

Faruk Polat is a PhD candidate in Computer Engineering and Information Science at Bilkent University in Ankara, Turkey. He has been working on his dissertation as a visiting NATO science scholar at the University of Minnesota, Minneapolis, for the 1992–93 academic year. His research interests include knowledge-based systems, knowledge base verification and distributed artificial intelligence. He received his BS in Computer Engineering from the Middle East Technical University, Ankara, in 1987 and his MS degree in Computer Engineering and Information Science from Bilkent University in 1989.

H. Altay Guvenir

H. Altay Guvenir is an Assistant Professor of Computer Engineering and Information Science at Bilkent University in Ankara, Turkey. His research interests include expert systems, machine learning and computational linguistics. He received his MS in Electrical Engineering from Istanbul Technical University in 1981 and his PhD in Computer Science from Case Western Reserve University in 1987. He is a member of AAAI, ACM, ACM SIGART and the International Association of Knowledge Engineers.

Shashi Shekhar

Shashi Shekhar received the B.Tech degree in Computer Science from the Indian Institute of Technology, Kanpur, India in 1985, and the MS degree in Business Administration and the PhD degree in Computer Science from the University of California, Berkeley in 1989. He is currently an Assistant Professor in the Department of Computer Science at the University of Minnesota, Minneapolis. His research interests include databases, artificial intelligence and software engineering with emphasis on important applications such as manufacturing. Dr Shekhar is a member of the IEEE Computer Society, ACM and AAAI.