2004

Reasoning and Learning with Imperfect Casebases: An Agent Perspective with An Expert Model

Leen-Kiat Soh
University of Nebraska, lsoh2@unl.edu

University of Nebraska-Lincoln, Computer Science and Engineering
Technical Report # TR-UNL-CSE-2004-0002

Reasoning and Learning with Imperfect Casebases: An Agent Perspective with An Expert Model

Leen-Kiat Soh
Department of Computer Science and Engineering
University of Nebraska
115 Ferguson Hall, Lincoln, NE 68588-0115, USA
Tel: (402) 472-6738 Fax: (402) 472-7767 E-mail: lksoh@cse.unl.edu
December 21, 2003

1 Introduction

Traditionally, case-based reasoning (CBR) (e.g., Watson and Marir 1994) assumes that the cases in the casebase are correct, useful in both time and space. Otherwise, the cases would not have been stored in the casebase in the first place. Cases are supposed to be useful in guiding us to a successful solution, or in preventing us from repeating the same failure.

- But these suppositions are valid when the problem specifications that the cases were stored for are relatively static. Furthermore, even if the problem specifications are relatively static, the types of problems that the CBR module actually encounters may be varying over time. For example, suppose the casebase contains three types of cases, A, B, and C. In a certain period, the CBR module may only encounter problems of type C, and not the others. Thus, in a way, during the best case retrieval, the CBR module should not consider cases of types A and B. In traditional setting, the CBR module, in its search for the best case, will consider all types. When the casebase is large, this presents a possible scalability problem.
- Also, these suppositions are valid when the casebase is assumed to be available and complete. This of course is not always true, and has been documented in the literature (Kolodner 1993). Cases may not readily available and thus a CBR module has no choice but to start with an approximated casebase. Researchers thus have introduced case-based learning (CBL) to allow new cases to be added to the casebase as the CBR module operates in the problem domain. This solution is a good approach when the problem domain is noiseless and certain. For example, suppose the CBR module encounters a new problem and finds a new solution that results in a successful outcome. Does the CBR module learn this case? What if the success was actually one of the possible outcomes due to noise in the problem domain? Traditionally, we store cases that are deemed "expertise". But now, if the above new case is a combination of "chance encounter" and "expertise," then should the case be learned? If so, how should the case be learned? In a way, traditional CBR does not learn new expertise, it only refines expertise already stored in the casebase.
- Finally, these suppositions are valid when there is a single decision maker. However, there are environments where there are multiple decision makers, such as a multiagent system. Each agent has a different experience, and may maintain a different casebase even if each is given the same initial casebase. Now, in traditional CBR and CBL, a CBR module has some allowance to make mistakes in its choice of solutions until it finally learns the correct solution to a problem. This process may be short if the solution that the module has already in its casebase is not far off from the correct solution, and the adaptation step is able to bring the that solution to the correct solution in a small number of iterations. However, is it possible to speed up this process? In a

multiagent system, theoretically it is possible. Since each agent is a decision maker, it is possible for agent *A* to have found the solution to a problem *P* while agent *B* to have just seen the problem *P* a couple of times. Thus, instead of "struggling" to solve the problem *P* on its own, agent *B* could contact agent *A* for a possible solution. Traditional CBR does not address that.

In the following, we outline agent-based novice-to-expert CBR and CBL framework.

2. Agent-Based Novice-to-Expert Case-Based Reasoning and Learning Framework

2.1. Expert Model

According to (Bransford *et al.* 1999):

> "*By definition, experts have developed particular ways to think and reason effectively. Understanding expertise is important because it provides insights into the nature of thinking and problem solving. It is not simply general abilities, such as memory or intelligence, nor the use of general strategies that differentiate experts from novices. Instead, experts have acquired extensive knowledge that affects what they notice and how they organize, represent, and interpret information in their environments. This, in turn, affects their abilities to remember, reason, and solve problems.*"

Bransford *et al.* (1999) identified six key conclusions:

- Experts notice features and meaningful patterns of information that are not noticed by novices.
- Experts have acquired a great deal of content knowledge that is organized, and their organization of information reflects a deep understanding of the subject matter.
- Experts' knowledge cannot be reduced to sets of isolated facts or propositions but, instead, reflects contexts of applicability, i.e., it is "conditionalized."
- Experts are able to retrieve important aspects of their knowledge with little attentional effort.
- Though experts know their disciplines thoroughly, this does not guarantee that they are able to instruct others about the topic.
- Experts have varying levels of flexibility in their approaches to new situations.

The above conclusions were based on numerous studies on school students, and have been adopted in educational studies to improve student learning.

In our research, we see the above conclusions as key to guide our framework of an agent or a multiagent system reasoning and learning with imperfect casebases. We see a casebase as expertise; that is, each case is a problem-solution pair that one may want to re-use for the next problem, or may want to avoid for the next problem. We see an agent as a semi-novice equipped with an initial casebase that may be noisy, incomplete, or inappropriate. Also, we see an agent capable of learning to reason more like an expert as it interacts with its environment and other agents.

2.2. Agent Characteristics

Each agent has the following characteristics:

(1) Autonomous – Each agent runs without interaction with human users. It maintains its own knowledge base, makes its own decisions, and interacts with its sensor, neighbors and environment.

(2) Rational – Each agent is rational in that it knows what its goals are and can reason and choose from a set of options and make an advantageous decision to achieve its goal (Wooldridge and Jennings 1995). In the resource allocation domain, each agent is selfish as each attempts to preserve its control of CPU resources and conserve its power usage, for example. Yet, each agent is bound by a global objective that demands each agent *cooperate* to share resources. The two balancing goals drive the rationality of our agents.

(3) Communicative – Each agent is able to communicate with others, by initiating and responding to messages, and carrying out conversations.

(4) Reflective (or Aware) – According to Brazier and Treur (1996), a reflective agent reasons based on its own observations, its own information state and assumptions, its communication with another agent and another agent's reasoning, and its own control or reasoning and actions. By being reflective, each agent is time aware and situationally aware. When an agent is time aware, it observes time in its decision making and actions. Its reasoning takes time into account, and thus, the outcome of a reasoning process is partially dependent on time. When an agent is situationally aware, it observes its current situation, the situation of its neighbors, and that of the world and makes decisions based on these observations. In general, an agent that is situationally aware observes the resources that it shares with other agents, its current tasks, messages, profiles and actions of its neighbors, and the external changes in the environment. In this paper, we require a stronger level of situational awareness. An agent also observes its own resources that *sustain the being* of the agent. For a hardware agent, these resources may be the battery power, the radio frequency links, etc. For a software agent, these resources may be CPU, RAM, disk space, communication channels, etc. In general, discussions on agent-based negotiations ignore, for example, the CPU allocation and usage that enables the execution of a negotiation, assuming such resources as unlimited or always-accessible. Note that, for example, in (Sandholm & Lesser 1995), a *bounded rationality* model is used where each agent has to pay for the computational resources (CPU cycles) that it uses for deliberation, assuming that the resources are available. In our model, however, we require an agent to be aware of whether the resources are available before even starting a negotiation.

(5) Honest – Each agent does not knowingly lie or intentionally give false information. This characteristic is also known as veracity (Galliers 1988).

(6) Adaptive – Each agent is able to adapt to changes in the environment and learns to perform a task better, not only reactively but also from its past experience.

(7) Cooperative – Each agent is motivated to cooperate if possible with its neighbors to achieve global goals while satisfying local constraints.

Generally, the agents in a multi-agent system may be controlling different resources and use different reasoning and negotiation techniques. In our approach, we require (1) that all agents be capable of negotiation in which they share a common vocabulary that enables message understanding, and (2) that each agent knows what resources may be used or controlled by a non-empty subset of the other agents in the environment so that it can determine whom to negotiate with. In our particular domain of application, each agent controls the same resources, since each one controls the same type of sensor. Also, each agent uses the same negotiation methodology based on case-based reasoning, but the individual case bases differ.

2.3. Casebase Assumptions

We assume the following for a casebase in our framework.

(1) Incomplete – The casebase may be incomplete in its coverage. That is, not all key problem-solution pairs encountered in a system are encoded cases in the casebase.

(2) Noisy – The casebase may be noisy. The problem or solution description of a case may not be accurate. Some descriptors may contain inaccurate values. From an agent's standpoint, the quality of its sensors may not be noiseless, resulting in inaccurate problem description. Likewise, when an agent learns a solution, the goodness of the solution may have been influenced by the noise in the observed outcome.

(3) Uncertain – The solutions of the cases may not yield expected outcomes with certainty due to the dynamic environment with its noticeable probability of events. Therefore, a case that has worked once before not work when the same problem occurs.

(4) Unstable – The solutions of the cases may stop to work or start to work after a certain period due to long-term changes in the environment. For example, a case may be useful with a solution that yields good outcomes for the first 100 times it is retrieved. However, because of a change in the environment (e.g., a new agent joins the system), the case may cease to be successful. This implies that a case cannot be assumed to be always useful even though it has been shown to be useful consistently before. Similarly, a case cannot be assumed to be always harmful as well.

(5) Evolutionary – The casebase may have to evolve to adapt to changing focus or applications in the environment where the agent is situated. That is, cases that have worked before may no longer be useful because the agent has started to encounter a new set of events. Thus, cases that have not been used in a long while may be filtered out or partitioned out from the casebase.

(6) Positive and Negative Expertise – The casebase may contain both types of cases, one with good solutions and one with bad solutions. A case with a good solution helps us solve similar problems by providing a good solution to adapt from. A case with a bad solution helps us solve similar problems by providing a bad solution to adapt *away* from. This assumption allows the casebase to store both positive and negative expertise, increasing the flexibility of our CBR and CBL framework. Note that, traditionally, when a casebase is assumed to be *good*, it does not contain cases with bad solutions.

(7) Distributed – The casebase may be distributed among different agents and evolved individually. Each agent may start with the same initial casebase but eventually populate and refine its casebase based on its own experience. Agents may choose to share their expertise and experience.

(8) Malleable – The casebase may be modified. Cases may be removed from the casebase due to their inappropriateness to the problem currently encountered by the agent. A case may also be replaced if another case has been found to cover the same problem space better. A new case may be added because of its potential impact in future problem solving of the agent.

The above assumptions about the casebase define an *imperfect* casebase. We see that these assumptions motivate agent learning from novice to expert. An agent should strive to cover more completely its problem space, should learn to monitor and observe problem descriptors and outcomes better, should identify the uncertainty in its cases, should be flexible enough to change its previously, empirically proven beliefs, should adapt to changing environments, and should be able to recognize good and bad cases. In a multiagent system, each agent's casebase is *imperfect* in the sense that each only represents a subset of the entire casebase of the system, and agents may choose to share their cases. Finally, we assume that the casebase is malleable. An agent may modify its casebase as it interacts with its environment overtime. This allows the agent to learn. Thus, a malleable casebase implies that the casebase is imperfect; it also implies that an agent with a malleable casebase can improve the casebase.

2.4. Integrated Framework: Novice-to-Expert Agent CBR and CBL

In this section, we present an integrated framework for CBR and CBL, from an agent's point of view, based on the expert model, agent characteristics and assumptions of the casebase as described in previous sections. The rationales behind this integration are the following.

The expert model (Bransford *et al.* 1999) provides an empirical, cognitive model on how novices become experts, how experts teach novices, and how experts differ from novices in reasoning. The empirical model not only describes how experts reason, it also prescribes to some extent how novices should learn to become experts. The model also shows that experience plays a monumental role in how humans gain their expertise. The cognitive model, on the other hand, provides guidelines on how knowledge or expertise could be organized, represented, accessed, and retrieved.

CBR—and CBL—is an AI paradigm naturally suitable for the application of the expert model.

First, in CBR, a casebase is seen as a collection of expertise, in which each case is an item of expertise—with a problem situation and its solution. Each case does not explicitly state the actual causal link between a situation and its solution. It simply indicates a useful association between the two. This fits the cognitive model of experts. An expert establishes useful associations between situations and solutions, even though he or she possesses the implicit or explicit knowledge justifying why a particular solution is appropriate for a particular problem.

Second, a case is a flexible representation of expertise. Unlike rules or heuristics, a case does not require an exact match to be useful because of the adaptation step of CBR. Thus, changing its problem situation parameters does not necessarily render it un-useable. This flexibility allows the expert model to play a role in *refining* the problem space without the system worrying about losing its applicability. (Of course, if the problem space is significantly changed, then the system may no longer to solve what it has been design to do.)

Third, a case is a modular representation of expertise. A case does not in general rely on other cases when used. CBR may occasionally look at several good cases to obtain a consensus, but in general, these cases can be used independently of each other. However, in a rulebase, the results of fired rules have to be synthesized to obtain a consistent set of decisions. Thus, some changes in a rule will impact the usefulness of other rules. This is because rules are usually not independent of each other. The modularity of the cases allows each case to be dynamically changed in the casebase at runtime. And that facilitates the integration of the expert model.

Fourth, a case is a sharable representation of expertise. This sharability stems from a case' modularity. Agents, for example, can exchange and use cases from other agents more easily than rules. When incorporating a piece of foreign expertise into the local knowledge base, an agent has to *break in* the piece, to ensure the this new piece does not disturb the existing knowledge base and lead to undesired results. A case, because of its modularity, can be more easily incorporated. This opens up a host of possibility for distributed CBR and CBL, and provides a convenient platform for the expert model: a novice agent can learn from another expert agent, and an expert agent can teach a novice agent.

In the following, we describe our proposed integrated framework using the six conclusions of the expert model as our design principles of reasoning and learning with imperfect casebases.

2.4.1. Meaningful Patterns

The first conclusion of the expert model states that experts notice features and meaningful patterns of information that are not noticed by novices. To incorporate into our CBR model, we emphasize on two components of the conclusion: noticeability of features and meaningfulness of patterns of information. As an agent learns and moves from being a novice to being an expert, that means it should be able to notice more and more features and meaningful patterns of information. To capture the notions of noticeability and meaningfulness, we propose the following design principles. The key here is to allow an agent to develop sensitivity to patterns of meaningful information.

Principle 1. Noticeability Measurement I An agent should increase the noticeability of useful problem-solution pairs and useful adaptation heuristics of its casebase.

A case that is used often and used successfully often should have a higher degree of noticeability. It does not necessarily enable the case to be retrieved often; however, it should increase the chance of the case being used in finding meaningful information. Furthermore, since the casebase is imperfect, the adaptation step, even with correct adaptation heuristics, may not find the correct solution. As a result, some heuristics have to be modified. Heuristics with good noticeability will be preserved while those with poor noticeability will be modified or replaced.

To measure noticeability, we may keep track of the usage history of each case—documenting the number of times a case is used, and the number of times the use of the case leads to a successful outcome. We may also keep track of the adaptation heuristics in terms of each heuristic's contribution to the success or failure of a solution. For example, suppose a heuristic *h1* moves the best case' solution to the adapted solution, by *d*. If the solution eventually leads to a success, then the heuristic *h1* is rewarded with a noticeability measure proportionally to *d*. Likewise, if it leads to a failure, then the heuristic is penalized accordingly.

Note that the noticeability measure influences neither how a case is retrieved nor how an adaptation heuristic is used, at least not directly. The noticeability measure simply provides a measure that agent may use to look for meaningful patterns.

Principle 2. Noticeability Measurement II An agent should reduce the noticeability of problem-solution pairs or adaptation heuristics that have not been used in a long time.

To measure this, we may tag to each case a *recency* measure—a last-used time stamp. The noticeability of a case decays as the difference between the current time and the last-used time stamp increases.

Principle 3. Noticeability Entailment An agent should observe the noticeability of a case or an adaptation heuristic over time and entail *significant* trends (using techniques such as linear regression, moving average, etc.) from the observation.

We see also intrinsic and extrinsic trends. Intrinsic trends are observations based on only a single case. If a case' noticeability curve is monotonically increasing, for example, that means the case has been used often and use successfully often. Extrinsic trends are observations resulting from comparisons among several cases. For example, suppose a case C_1's noticeability decreases over time while all other cases in the casebase maintain their noticeability. That means this trend is meaningful—it may imply that C_1 be replaced or modified. Similar entailments can be generated for the adaptation heuristics.

Principle 4. Meaningfulness Annotation An agent should annotate how meaningful a case is to its casebase based on the case' noticeability and its trends. Similarly, an agent should do the same for an adaptation heuristic.

The above principle follows from Principle 3. Here we use an annotation, as simple measurements will not be rich enough to capture the meaningfulness of a case or an adaptation heuristic.

Principle 5. Meaningfulness Entailment An agent should increase its sensitivity to meaningful annotations found in its casebase as it gains expertise.

First, as an agent gains expertise, the overall noticeability of its casebase should also increase, meaning that it is getting better at problem solving. Thus, we see this sensitivity measure proportional to the noticeability of the casebase. When an agent is a novice, its cases and adaptation heuristics will have low noticeability. This leads to meaningful annotations with little confidence. Given these annotations, the agent will not act upon them. And vice versa.

Thus, with a noticeability value that decays with time and increases with usefulness, an agent will be able to gain sensitivity to useful situation-problem pairs and heuristics and meaningful patterns. In the subsequent sections, we will see how this awareness is used in the organization of cases and heuristics.

2.4.2. Organization

The second conclusion of the expert model states that experts have acquired a great deal of content knowledge that is organized in ways that reflect a deep understanding of their subject matter. In Section 2.4.1, we define noticeability of knowledge and its meaningfulness. Here, we use these characteristics to help an agent decide how to organize knowledge.

Principle 6. Adaptive Resolution An agent should organize its cases such that frequently used cases have a higher resolution of coverage, centered around most noticeable and meaningful cases.

In an expert's mental model, knowledge is organized around big ideas. As an agent learns, it gradually identifies with useful cases and increases the coverage resolution of these cases. We see this as an effective approach to *adaptive* CBL. Generally, a new case is learned only when it provides *enough* diversity to the existing casebase. To achieve this, static thresholds are used to determine whether a case is diverse or useful enough. Principle 6 suggests that these thresholds be dynamically determined so that the diversity threshold is lower around a noticeable and meaningful case. In that manner, an agent will become increasingly precise in solving problems around these areas. In other words, it will be more difficult to learn a new case in a region where existing cases are not noticeable and meaningful.

We see this Adaptive Resolution Principle as a good learning approach, especially for CBL, as it has the following advantages. First, it does not rely on a set of static thresholds that may become obsolete or not appropriate as the events encountered by the agent change. Second, it allows problems in close proximity of a frequently used case to be addressed more precisely—with the increasing number of problem-solution pairs stored. This cuts down the risks in adapting a solution that may or may not work from a representative case. Third, it does not learn new cases as easily around a less noticeable and less meaningful case. This prevents the casebase from learning cases that will not likely be used often. Fourth, it will also provide a measure of authority of the agent, based on the number of cases in the proximity of a meaningful and noticeable case, in a particular problem, and also provide clues for meta-level learning (abstraction).

We also see several drawbacks of the principle. That is, it does not prevent an agent from learning too many cases and that would lead to a reduced importance of the adaptation heuristics, and would also lead to a larger search space during the best case retrieval process. Also, when a case gains in its noticeability and meaningfulness, its diversity threshold decreases, allowing more cases that may be less different to be learned. We will address this drawback using the notion of authority.

Principle 7. Authority Measurement An agent has should increase its authority in solving a particular problem as it has a high number of cases in a *problem neighborhood*. Here a *problem neighborhood* is defined with a *core*—the most noticeable and meaningful case in the neighborhood, a pre-defined *neighborhood distance,* $D_{Neighborhood}$, and a set of cases within $D_{Neighborhood}$ from the core.

Given the above principle, we see that an agent will gradually obtain a high authority in a particular problem if it keeps encountering and solving the similar problems. The neighborhood distance $D_{Neighborhood}$ is an adjustable design parameter. A large value allows the agent to generalize more easily as it finds a core case for a large neighborhood and vice versa. Note that this value is not the same as the *K* value in *k*-nearest neighbors approaches. In our framework, we choose to keep the neighborhood a constant for each case, and the number of neighbors within each neighborhood a variable.

Note also that given this authority measurement, an agent is able to fine-tune its knowledge of representative problems. That is, a case that starts out as the *core* of its neighborhood may relinquish that role after a more definitive case is learned. Such a case will be used more frequently and more successfully, thus assuming the role of the core of the neighborhood and dominating the original core. In essence, the problem-solution pair is refined.

Principle 8. Authority Entailment I An agent should be able to cluster related cases into units.

According to the expert model, experts appear to possess an efficient organization of knowledge with meaningful relations among related elements clustered into related units that are governed by underlying concepts and principles. With the definition of authority—the core and the neighborhood, our framework accomplishes the clustering in the following manner. For each case in the casebase, determine whether each is a core—the most meaningful and useful case within its neighborhood. If yes, then stop. If no, then reduce the neighborhood distance by 1, and iterate, until cases are cores—some will be a core of a neighborhood of only one member, i.e., itself. Each neighborhood is then a cluster. The above procedure is admittedly a simple hierarchical clustering approach.

Note that the procedure allows for multiple clusters to overlap. This would lead to confusion or indeciveness in the generalization of knowledge. We will touch upon this in Principle 9 below, as we will address that more completely when we discuss the context and access principles in Section 2.4.3. Note that this clustering principle allows an agent to identify the core cases and contextualize the problem-solution pairs to improve access.

Principle 9. Authority Entailment II An agent, as it gains knowledge, should gradually learn clearly defined clusters with fewer and fewer, and ultimately, with no overlapping cases.

A case that belongs to two clusters is confusing in terms of context and access (Section 2.4.3 later). However, initially, an agent without adequate authority in its solutions to a particular problem does not have the confidence to address this issue. Subsequently, it may have cases that belong to different problem neighborhoods. To resolve this, we look to the authority of the competing cases. When the authority of a case increases, its diversity threshold decreases. If the confusing case's location and utility are qualified (above the thresholds of one of the competing cases), then it joins the neighborhood of that case. If it is qualified for both, then the confusion remains. With this entailment, the agent will try to better identify its cases with core problems. But it does not necessarily require a clean partition of cases if it does not have sufficient authority to do so.

Principle 10. Clarity Measurement The *clarity* of an agent's knowledge is based on the number of overlapping cases in its clusters.

This principle stems from Principle 9 above. The *clarity* of a casebase is proportional to the number of cases in the casebase, inversely proportional to the number of pairs of cases C_i and C_j—and their corresponding neighborhoods, N_{C_i} and N_{C_j}—where $N_{C_i} \not\subset N_{C_j}$, $N_{C_j} \not\subset N_{C_i}$, and $N_{C_i} \cap N_{C_j}$ is not empty, and inversely proportional to the size of $N_{C_i} \cap N_{C_j}$.

Note that an agent with a good authority in its solutions will likely have a good clarity of its casebase.

Principle 11. Authority Entailment IV An agent should be able to better judge how good a case is as it gains knowledge.

According to the expert model, experts can critique better. To incorporate this into our CBR and CBL framework, an agent should be self-aware and reflective, at least in terms of knowing which cases in its casebase are good, which are not as good. Once again, we rely on the authority measurement. A case that has a high authority and that is a core of its neighborhood is a good case.

Thus, with the above five principles, we now have a notion of authority of a case in its solving a representative problem, an organization strategy based on this authority notion, and a notion of clarity of a casebase in terms of the organization of the cases. In conclusion, an agent, when it gains expertise, will have cases with higher authority values and a casebase with better clarity. The clusters, as a result of the increased authority and clarity, will facilitate the *conditionalization* of experts' knowledge, as we discuss the next step of our framework in Section 2.4.3.

Principle 12. Authority Entailment V An agent should treat a critique of its solution to a problem when the critique comes from an agent with a high authority in solving that problem.

This principle follows from the previous ones and is important for agents in a multiagent system to exchange solutions and critiques. With this, an agent will be able to fuse solutions from other agents of different authority values, giving more weight to solutions coming from more authoritative agents.

2.4.3. Context and Access.

In this section, we incorporate the third conclusion of the expert model. This conclusion has to do with the context and access of knowledge. According to (Bransford *et al*. 1999), an expert's knowledge cannot be reduced to sets of isolated facts or propositions. Instead, an expert's knowledge reflects the context of applicability of that knowledge; that is, the knowledge is *conditionalized* on a set of circumstances.

Principle 13. Diversity Measurement An agent, as it gains expertise, should increase the diversity of its casebase.

We have mentioned the notion of *diversity* earlier in Section 2.4.2. It measures how different the cases are in a casebase. This can be the average *difference* between each pair of cases in a casebase, for example, by computing the normalized and weighted distance between each pair of attribute values. As observed in (Bransford *et al.* 1999), experts have a vast repertoire of knowledge that is relevant to their domain or discipline—meaning that an expert is able to solve a lot of distinct problems. Through case-based learning, an agent will be able to add diversity to its casebase (see Principle 6 in Section 2.4.2).

Principle 14. Size Measurement An agent, as it gains expertise, should be able to reduce the size of knowledge relevant to any particular problem.

Though an expert possesses a vast repertoire of knowledge, only a subset of that knowledge is relevant to any particular problem. Applied to our framework, this seems to be in conflict of the principles outlined in Section 2.4.2. In Principle 6 where we outline adaptive resolution, we propose that an agent adds more cases to support most noticeable and meaningful cases to increase the precision of the solution to the problems addressed by these noticeable and meaningful cases. But Principle 14 states that the size of knowledge relevant to any particular problem should be reduced as an agent gains expertise. Actually, the two principles can co-exist since they focus on the different spaces of a case. Principle 6 focuses on the problem space as it aims to increase the resolution of similar problems near the useful problem-solution pairs. Principle 14, on the other hand, focuses on the solution space as it aims to reduce the number of solutions for a problem. Thus, we introduce the notion of *solution possibility* of a problem as the number of solutions for a problem.

Principle 15. Size Entailment I An agent, as it gains expertise, should reduce the solution possibility of a problem.

Basically, this has two implications. First, recall that in our framework, an agent stores new cases that are different enough from those existing in the casebase. Thus, it is possible for a new case that has the exactly the same problem description with one of the cases in the casebase but a very different solution to be learned. Second, it is also possible to measure the goodness of the solutions since now the problem description is the same. To measure the goodness, we refer back to the noticeability value of a case, based on its usage history, as discussed in Section 2.4.1. Third, this principle does not prevent new cases from being learned; however, it does influence how these new cases are used in the future. Thus, to reduce the solution possibility of a problem, we look at two possible approaches. Experts do not have to search through everything they know in order to find what is relevant; such an approach would overwhelm their working memory (Bransford *et al.* 1999). To achieve this, one can narrow the search space as in the first approach, or can increase the chance of finding a good solution, as in the second approach.

(a) The first approach reduces the solution possibility of a problem directly. Find all cases with the same problem description. Retain the case with the best noticeability and remove others from the casebase, or transfer those to a repository. This approach may remove occasionally useful solutions and may end up re-learning them. This approach also needs to address the *incubation* period for a newly learned case, since a new case needs time to prove its usefulness to the agent.

(b) The second approach reduces the solution possibility of a problem indirectly. All cases with the same problem description are retained. However, the retrieval of a solution is based on a probabilistic selection. Suppose that given a problem *P*, there are *n* cases, meaning *n* possible solutions. Each case has a noticeability value. Based on similarity, all these cases will be retrieved as the best case to the problem *P*. However, only one solution is to be selected from the retrieved cases. A solution from a case with a higher noticeability value will have a higher chance to be selected, and vice versa. This approach is more flexible but it does not narrow the search.

We will see in the following principles how experts are able to make their search more efficient and effective using a combination of the above two approaches.

Principle 16. Conditionalization An agent, as it gains expertise, should be able to conditionalize more and more of its most noticeable and meaningful cases.

According to the expert model, experts have not only acquired knowledge, but are also good at retrieving the knowledge that is relevant to a particular task. Experts' knowledge is *conditionalized*—it includes a specification of the contexts in which it is useful (Bransford *et al.* 1999). With this principle, we introduce the notion of *conditionality* of a case. To measure the conditionality of a case, we examine the quality of problem-solution pairs to determine the cause-effect patterns between the problem and the solution. A problem with many different good solutions has a low conditionality value. A solution that works for many problems, however, has a high conditionality value. Thus, our framework promotes the applicability of a solution. As an agent gains expertise, it will have more and more noticeable and meaningful cases, as well as more authority. This implies that it will have more and more solutions that have worked. This thus leads to a higher confidence in determining the conditionality of a case. This conditionalization process is similar to data mining.

Principle 17. Conditionalization Entailment An agent, as it gains expertise, should access conditionalized cases more often and more conveniently.

Knowledge that is not conditionalized is often *inert* because it is not activated, even though it is relevant (Bransford *et al.* 1999). To simulate this mental behavior, an agent should first search for a solution in cases that have been conditionalized, and should prefer solutions from conditionalized cases. A novice agent will have few conditionalized cases and thus will still rely on non-conditionalized cases for solutions. An expert agent will have sufficient conditionalized cases such that it access unconditionalized cases only infrequently. Thus, in effect, those cases become inert.

At this point, we have discussed three groups of principles, from defining meaningful patterns to how to organize these patterns, to how to wrap these patterns in context for access. In Section 2.4.4, we see how the expert model guides our framework in retrieving cases.

2.4.4. Fluent Retrieval

According to (Bransford *et al.* 1999), experts are able to flexibly retrieve important aspects of their knowledge with *little* attentional effort. People's retrieval of relevant knowledge can vary from being *effortful* to *relatively effortless* (*fluent*) to *automatic*. Automatic and fluent retrieval are important characteristics of expertise. Note that under this conclusion, it does not necessarily mean an expert has a better solution to a problem; however it does imply that an expert is likely to obtain a solution faster as it is able to search faster—think less. Fluency is important because effortless processing places fewer demands on conscious attention. In Section 2.4.3, we introduce the notion of conditionalization. Here, we extend conditionalization to the notion of fluency.

Principle 18. Fluency An agent, as it gains expertise, should access more and more conditionalized cases in solving the problems that it encounters. This follows from Principle 17.

Given Principle 18, an agent is able to search a narrower space—only the conditionalized cases first. However, this principle does not reduce the amount of attentional effort in terms of the number of attributes or heuristics to look at. Thus, we have the following principles.

Principle 19. Importance Measurement An agent, as it gains expertise, should increase the importance of an attribute if the difference between its values in two cases impacts the quality values of the solutions. Similarly, an agent, as it gains expertise, should increase the importance of an adaptation heuristic if its application impacts the quality value of a solution.

To find out the importance of an attribute, for example, an agent may perform an association discovery between the attribute's values in all the cases stored in the casebase and the corresponding solution quality or outcome of those cases. An attribute that yields a high variety of outcome quality values indicates that the attribute is important. As an attribute in the problem description of a case, it means the attribute may factor more in the retrieval, giving the attribute a higher weight. As an attribute in the solution description, it means the attribute may factor more in adaptation, lessening the degree of adjustments during the adaptation process when the solution quality of the best case was high; and increasing the degree of adjustments during the process when the solution quality was low. The same argument also applies to the adaptation heuristics.

Principle 20. Importance Entailment I An agent, as it gains expertise, should be able to reduce the size of problem description and the number of useful adaptation heuristics for specific problems.

Through Principle 19, an agent will be able to identify attributes and heuristics that are useful. Combining the two, an agent should be able to ignore non-important attributes or heuristics for specific problems.

Principle 21. Importance Entailment II An agent, as it gains expertise, should be able to come up (retrieve and adapt) with a solution by looking at a smaller set of problem description (smaller number of problem attributes) and using a smaller set of adaptation heuristics.

This principle follows directly from Principle 20. Note that for Principles 19, 20, 21, we can further qualify them using the notion of authority as discussed in Section 2.4.2. That is, if a case has a higher noticeability and meaningfulness, it has a higher influence on determining the importance of the case' attributes. Moreover, we can also use the notion of conditionalization as discussed in Section 2.4.3 to qualify the above principles as well, giving those conditionalized cases more weight in the measurement of importance for an attribute or a heuristic. While the conditionalization principles define the useful problem-solution pairs, the importance principles focus on the useful attributes and heuristics, yielding a higher resolution. This reflects the fact that as an agent gains expertise, it has a better sense of problem description and appropriate solutions.

Principle 22. Categorization An agent, as it gains expertise, should be able to categorize problems into different types.

This principle follows from Authority and Conditionalization principles. In Section 2.4.2, we define the notion of authority of a case in its solving a representative problem, an organization strategy based on this authority notion, and a notion of clarity of a casebase in terms of the organization of the cases. An agent, when it gains expertise, will have cases with higher authority values and a casebase with better clarity. In Section 2.4.3, cases are conditionalized after achieving high noticeability and meaningfulness. This type categorization is thus performed on conditionalized cases with high authority. As an agent becomes an expert, it will be able to categorize these cases into distinct case types. With this principle, an agent is thus able to model its own cases—achieving meta-cognition.

Principle 23. Categorization Entailment I An agent, as it gains expertise, should be able to categorize a new problem into one of the case types that it has with increasing confidence.

In general, it is straightforward to categorize a new problem into one of the case types. The confidence measure is thus based on the authority of the case type.

Principle 24. Categorization Entailment II An agent, as it gains expertise, should be able to generate a set of rules of thumb for the solution of a particular case type.

On the other hand, based on the categorization and the attribute and heuristic importance, an agent should be able to generate the significant attributes and heuristics for a particular problem type. In a way, an expert agent moves away from specific cases, and looks at the models of the cases. It does not try to obtain a solution based on a case for a problem that it knows how to solve with high confidence and it has seen many times previously. Instead, it tries to use the rules of thumb to come up with the appropriate solution. Note that with this, we are in a way shifting from CBR to rule-based reasoning as an agent gains expertise. The following principle is used to guide the solution.

Principle 25. Categorization Entailment III An agent, as it gains expertise, should be able to guide the retrieval and adaptation for the solution of a new case using the rules of thumb for the particular case type of the new case.

Under the above principle, an agent should be able to obtain the appropriate solution without going through the retrieval process. It is now a pattern matching process. Note that the set of important attributes and the set of adaptation heuristics are both smaller than the original ones after Principles 21 and 22, under the fluency conclusion. Thus, it is easier to match a new problem against these attributes as well as apply these heuristics to obtain an appropriate solution. This entailment gives an agent a natural approach to store its expertise—from noisy cases (as assumed in our premise) to organized cases to conditionalized cases and finally to case types with reduced sets of attributes and adaptation heuristics. As the number of attributes decreases, that means the number of attributes that the agent needs to observe is smaller. As the number of heuristics decreases, that means the number of decision points that the agent needs to consider is smaller too. This in turn contributes to the fluency of the retrieval.

In a way, as an agent becomes an expert, its reasoning process becomes more reactive. The wide-area search that a novice performs is time consuming and often not targeted. When an agent can notice the usefulness of its cases better, it then is able to find more authoritative cases; when it has more authoritative cases, it is able to conditionalize them to find the problem-solution *templates*. Given the conditionalization, importance of attributes and heuristics, and problem categorization, an agent will then be able to come up with a solution more fluently, in a more targeted path. This targeted path is supported by the agent's experience in using the cases in its casebase.

2.4.5. Teaching

According to (Bransford *et al.* 1999), experts are not always able to teach others even though they know their disciplines thoroughly. Translating this conclusion to our framework, the decision among the agents is thus on how to share their knowledge—or cases. In a multiagent system, due to the varied experience that the agents encounter, they are bound to gain expertise at different rates. Some may be good at solving particular problems while some may not. Those that have become good are considered the experts and those that have not are novices. In a multiagent learning framework, the novices may request help from the experts to solve a particular problem. The experts supply their knowledge to the novices and we consider that *teaching*.

We see two approaches to our framework of multiagent *teaching*: prescriptive and descriptive. In *prescriptive teaching*, the expert agent informs the novice agent what the novice agent should do. In *descriptive teaching*, the expert agent informs the novice agent what the expert agent has done.

Principle 26. Prescriptive Teaching An agent, as it gains expertise in solving a particular problem, should tend to teach prescriptively more often other agents that request help in that problem.

As an agent gains in its expertise in solving a particular problem (the authority value of the problem increases, the conditionalization of the problem, and/or the categorization of the problem), it becomes more willing to share this knowledge (conditionalized cases or rules) with other agents. Since this knowledge is meta-cognitive, the agent in effect tells other agents what they should do.

Principle 27. Prescriptive Learning I An agent, as it receives different knowledge from other agents, weighs the importance (and/or usefulness) of the knowledge based on the authority of the source and importance of the knowledge.

As an agent gains in its expertise in solving a particular problem (the authority value of the problem increases, the conditionalization of the problem, and/or the categorization of the problem), it becomes more willing to share this knowledge (conditionalized cases or rules) with other agents. Since this knowledge is meta-cognitive, the agent in effect tells other agents what they should do.

Principle 28. Prescriptive Learning II An agent, as it receives a piece of prescriptive knowledge from another agent, *justifies* the foreign knowledge with its local knowledge.

The justification process is simply to determine whether the agent's local knowledge *supports* the foreign knowledge.

- If the agent has gained a certain degree of expertise—some of its local knowledge is in the form of conditionalized cases and categorized case types (or rules), then, the justification process looks for undercuts and rebuttals. If the local knowledge undercuts or rebuts (Parsons *et al.* 1998) the foreign knowledge, then that means the foreign knowledge is not justified. Otherwise, it is justified.
- If the agent is a pure novice—it has no conditionalized cases or categorized case types (or rules), then the justification process does nothing.

In the framework, a novice agent is considered to not have the adequate knowledge to rebut or refute a piece of prescriptive knowledge. (It is not so when it receives a piece of descriptive knowledge, however, as we shall see later.)

It is also important to note that an agent may be an expert and a novice at the same time, in different types of problems. This resolution is key in the interpretation of the principles discussed in this section. Thus, when we say "the agent is a pure novice," we actually mean that "the agent is a pure novice in solving problem *p*."

Principle 29. Prescriptive Learning III An agent *absorbs* prescriptive knowledge at different levels:

(1) As a pure novice, the agent accepts the prescriptive knowledge, files it away, and only comes back to it after it has gained enough expertise to justify the knowledge.

(2) As a semi-novice, semi-expert:

   a. If the prescriptive knowledge is justified, then the agent uses the knowledge as one of its own.

   b. If the prescriptive knowledge is not justified, then the agent files the knowledge and uses it to help in the agent's conditionalization and categorization processes.

(3) As an expert:

   a. If the prescriptive knowledge is justified, then the agent uses the knowledge as one of its own.

   b. If the prescriptive knowledge is not justified, then the agent discards it.

The idea here is even if the agent does not have enough expertise to justify and absorb a piece of prescriptive knowledge, it should still be able to use the prescriptive knowledge to help its conditionalization, as in the scenarios with a pure novice or a semi-novice, semi-expert.

Principle 30. Descriptive Teaching An agent, when it does not have enough expertise (no appropriate rules of thumb or conditionalized, categorized cases), should be able to provide appropriate cases to requesting agents. Some cases may have low noticeability and meaningfulness.

The above principle allows a novice agent to still contribute to another agent's problem solving. When an agent receives a request, it checks its level of expertise indicators for the expertise related to the request—i.e., the number of rules of thumb, the number of conditionalized, categorized cases, for example—and if the numbers are low or zeros, then it sends over a case instead. Note that we call this *descriptive teaching* since the agent itself is not confident whether the case that it has been using is "what an agent should do." It is simply what the agent has been using. This reflects the expert model truly. In Principle 26, an expert agent teaches prescriptively since such rules of thumb or conditionalized cases are less noisy—those pieces of information have been derived after numerous episodes of experience in solving a particular task. In Principle 29, a novice agent realizes that what it knows is noisy and thus teaches only descriptively.

Principle 31. Descriptive Learning I An agent, as it receives a case from another agent upon its request to help solve a problem, justifies the case and *adopts* the case if it does not undercut or rebut existing rules of thumb or conditionalized cases. Otherwise, it is thrown away.

Note that an agent is willing to adopt a case as long as it does not weaken its own expertise in other solving other problems. So, the above principle serves a filter for that. Note also that with this principle, it is possible that an agent corrupts its own casebase with the imported cases that may not be useful at all. However, as discussed in our previous sections, a case that is not useful will be systematically forgotten. Also, since this is descriptive learning, the agent basically learns about what other agents have been doing to solve a particular problem. So, the above arrangement does not require the agent to use the imported case. Depending on how the case has been used by the teaching agent, the learning agent may adopt it with different degrees, as outlined in the following principle.

Principle 32. Descriptive Learning II After deciding to adopt a case, the agent incorporates the case into its casebase based on the noticeability and meaningfulness of the case:

(1) If the case has low noticeability, regardless of its meaningfulness, then it is incorporated by resetting all its usage history, as if it was newly created case.

(2) If the case has high noticeability, then it is incorporated by reducing all its usage history in proportion to the case's meaningfulness.

Suppose that the learning agent is a_L and the teaching agent is a_T. The above principle cautions that a useful case in a_T's reasoning activities may not be necessarily useful in a_L's. In our framework, we have a conservative approach. If the case has low noticeability, the learning agent simply resets the case and stores it as a new case. If the case has high noticeability, the learning agent reduces its usage history in proportion to the case's meaningfulness. Recall, from Section 2.4.1, that a case's meaningfulness indicates the trend of a case's noticeability. And vice versa. In this manner, the learning agent does not blindly adopt a case fully.

Note that it is possible for the learning agent to incorporate a profile of the teaching agents that it has come into contact with. If an agent has been able to give out good, useful cases, the learning agent may update a *teaching authority* of that agent accordingly and factor cases adopted from that particular agent accordingly. This discussion, however, is not within the scope of this paper, which focuses on designing a CBR and CBL framework using the expert model outlined in (Bransford *et al.* 1999).

2.4.6. Adaptive Expertise

Finally, Bransford *et al.* (1999) outlined the last characteristic of experts. In general, experts have varying levels of flexibility in their approach to new situations. An expert is usually more resourceful than a novice. To obtain flexibility, we look to analogy-based reasoning.

Moreover, experts have a higher level of metacognition. This refers to one's ability to recognize the limits of his or her current knowledge. To recognize the limits of one's current knowledge, he or she must be able to evaluate how well his or her current knowledge is solving the problems encountered. And this can be based on what we have defined in the previous five sections.

Principle 33. Metacognition Measurement An agent, as it becomes an expert, should increase its level of metacognition.

Principle 34. Applicability Measurement Following Principles 22-25, a rule's *applicability* measure depends on the number of cases that fall under the case type that the rule describes.

To measure the level of *metacognition*, we look at three different types of knowledge: ordinary cases, conditionalized cases, and rules. As an agent gains expertise, it begins to have conditionalized cases and ultimately may have rules. Each ordinary case has a usage history (noticeability and meaningfulness). So does a conditionalized case or a rule. The key here is to measure the noticeability or applicability of each case or rule. An agent's metacognition is a function that is proportional to each rule's *applicability* (how many cases does it cover), each conditionalized case's *conditionality* (Principle 16), and each ordinary case's *noticeability* (Principles 1-2).

Principle 35. Metacognition Entailment I An agent, as it gains in its level of metacognition, should be *less patient* regarding problems that it does not know how to solve or it has failed to solve.

Remember in the previous sections, we allow an agent to be patient before labeling a problem-solution pair to be useless and requesting help from other agents. For an agent with a high level of metacognition, it immediately addresses problems that it fails to solve. Therefore, the agent is more likely to realize its weakness and immediately ask for help through either descriptive learning or prescriptive learning (Section 2.4.5). The measure of *patience* is simply a function inversely proportional to an agent's metacognitive level.

Principle 36. Metacognition Entailment II An agent, as it gains in its level of metacognition, should re-examine and address its past failures, including cases covered or used in the conditionalization and categorization that have led to conditionalized cases and rules.

An expert with a high level of metacognition should be able to re-examine its own knowledge. As discussed in the previous sections, it is possible for an agent to have cases that have poor solutions yet generate conditionalized cases or rules that are likely useful. An expert will find those cases and ask for help to fix them. If such a case is fixed, then the agent will accordingly update the conditionality and applicability of its cases and rules.

Principle 37. Flexibility Measurement An agent, as it gains expertise, should gain flexibility (or resourcefulness) in its reasoning.

With this principle, we look to analogy-based reasoning. According to Bransford *et al.* (1999), an expert is more resourceful and able to think across domains and applications to find a solution. Within a small scope, we see a good solution to a particular case type to be possibly useful to other case types, as well as a bad solution to a particular case type to be possibly poor as a solution to other case types. Within a larger scope, we extend the discussion to domain types. Here, we measure the flexibility of an agent based on the agent's success rate in applying a solution to a particular case type to another case type.

Principle 38. Flexibility Entailment I An agent, as it gains flexibility (or resourcefulness) in its reasoning, should apply conditionalized cases or rules to problems that the cases or rules do not originally cover.

Note that an agent performs the above for all new problems based on a frequency value determined by flexibility proportionally. Basically, it should become more aware of its past successes and try to generate analogous solutions that may yield a good solution.

Principle 39. Flexibility Entailment II An agent, as it gains flexibility (or resourcefulness) in its reasoning, should become more aware of its past failures and try to avoid using the same solution that may yield a poor solution.

In general, an agent in CBR may have outcome-driven adaptation to avoid making the same mistakes given the same problem. This is covered in our previous sections. With Principle 37, however, we propose to heighten the cautiousness of an agent as it gains expertise in evaluating its possible solution. Basically, all failed solutions will be stored, indexed by their corresponding problems. When a new problem comes in, the agent will generate a solution, say, *S*, based on CBR. It will then compare the solution with the failed solutions stored, regardless of the problems matching the new problem or not. Note that this cautiousness directly affects the impact of these failed solutions on *S* proportionally. That is, if an agent has no metacognition, then it does not take into account that failed solutions for problems not similar to the new problem that it is trying to solve. In fact, what we propose here is akin to analogy-based reasoning. The key here is to identify the common theme in the failed solutions—a theme that has a high number of parameters (attribute-value pairs) is more useful, for example.

Principle 40. Metacognition and Flexibility Entailment I An agent, as it performs flexibility-driven reasoning (or analogy-based reasoning), can encounter failures.

Note that as an agent gains in its metacognition and flexibility, it performs more analogy-based reasoning. It is possible that the agent may encounter failures. These failures will be recorded to prevent the agent from performing similar analogy-based reasoning steps in the future. If a poor solution for a problem P_1 is found to be very useful in other problems, then the poor solution will not be used in analogy-based reasoning. Similarly, we determine the suitability of a good solution as well.

Principle 41. Metacognition and Flexibility Entailment II An agent, as it fails in its flexibility-driven reasoning (or analogy-based reasoning), will decrease in its expertise.

This principle allows an agent to become a novice if it becomes too aggressive in exercising its expertise. If the number of failures overwhelms the agent, the quality of its conditionalized cases and rules will go down. This will lead to a decrease in the level of metacognition. This basically keeps the agent in check.

3. Conclusions

We have proposed and described a framework for reasoning and learning with imperfect casebases. The framework is based on an expert model that deals with selective retrieval, conditionalization, fluent retrieval, sharing, adaptivity, and metacognition. We have outlined the framework from an agent perspective, situated in a dynamic, noisy environment where there are multiple agents. We have also defined the notion of imperfect casebases in terms of completeness, noise, uncertainty, stability, evolution, positive and negative expertise, distributedness, and malleability.

Our immediate next step is to perform a thorough literature review of current CBR and CBL techniques in terms of the above framework. In fact, we have started work on the above in the past few years (Soh and Tsatsoulis 2001; Soh and Luo 2003a, 2003b; Luo 2003).

Following that we will build a comprehensive multiagent system satisfying the above six sets of principles as an infrastructure, to further investigate the feasibility of the framework and apply the framework to real-world problem domains.

4. References

Bransford, J. D., A. L. Brown, and R. R. Cocking (eds.). (1999). How Experts Differ from Novices, in J. D. Bransford, A. L. Brown, and R. R. Cocking (eds.) *How People Learn: Brain, Mind, Experience, and School*, pp. 19-38, Chapter 2, Committee on Developments in the Science of Learning, National Research Council.

Brazier, F., J. Treur (1996). Compositional Modelling of Reflective Agents, *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*, Banff, Alberta, Canada, pp. 13/1-13/12.

Galliers, J. R. (1998). A Strategic Framework for Multi-Agent Cooperative Dialogue, *Proceedings of ECAI'88*, Munich, Germany, pp. 415-420.

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.

Luo, J. (2003). *Case-Based Learning Behavior in a Real Time Multi-Agent System*, MS Project Report, Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE.

Parsons, S., C. Sierra and N. R. Jennings (1998). Agents that Reason and Negotiate by Arguing, *Journal of Logic and Computation*, **8**(3):261-292.

Sandholm, T. W. and V. R. Lesser (1995). Coalition Formation among Bounded Rational Agents, *Proceedings of IJCAI-95*, Montreal, Canada, pp. 662-669.

Soh, L.-K. and J. Luo (2003a). Muiltiagent Case-Based Reasoning through Individual and Cooperative Learning, *Proceedings of the IJCAI 2003 Workshop on Agents and Automated Reasoning*, Acapulco, Mexico, 44-51.

Soh, L.-K. and J. Luo (2003b). Combining Individual and Cooperative Learning for Multiagent Negotiations, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03),* Melbourne, Australia, 1122-1123.

Soh, L.-K. and C. Tsatsoulis (2001). Combining Genetic Algorithms and Case-Based Reasoning for Genetic Learning of a Case Library: A Conceptual Framework, *Proceedings of the Late-Breaking Papers* of *Genetic and Evolutionary Computation Conference (GECCO'01),* July 7-11, San Francisco, CA, pp. 376-383.

Watson, I. and F. Marir (1994). Case-Based Reasoning: A Review, *Knowledge Engineering Review*, **9**(4):327-354.

Wooldridge, M. and N. Jennings (1995). Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review*, **10**(2):114-152.