Problem-Solving Stereotypes for an Intelligent Assistant

Christopher Owens
Department of Computer Science, The University of Chicago
1100 East 58th Street, Chicago, IL 60637
E-mail: owens@cs.uchicago.edu

Abstract

This paper examines the role of case-based reasoning in a *problem-solving assistant* system, which differs from an autonomous problem solver in that it shares the problem-solving task with a human partner. The paper focuses on the criteria driving the system designer's (or the system's) choice of cases, of representation vocabulary, and of indexing terms, and upon how the assumption of a human in the problem-solving loop influences these criteria. It presents these theoretical considerations in the context of work in progress on IOPS, a case-based intelligent assistant for airline irregular operations scheduling.

Introduction

While most work on AI problem-solving has been directed towards the goal of building autonomous systems, capable of reasoning independently from an initial problem description to a successful solution, a growing body of work has begun focusing on the practical and scientific role of *intelligent assistants*: systems that do not solve problems autonomously, that instead enter into a problem-solving partnership with a human user.

This paper examines the role of episodic memory and case-based reasoning in the context of an intelligent assistant system. It focuses on a kind of knowledge that either an autonomous problem solver or an intelligent assistant system might embody: knowledge linking the commonly-occurring threats, opportunities, and failures of the problem-solving domain with appropriate responses to those situations. It explains how the assumption of a cooperative, as opposed to autonomous, problem solver changes the functional constraints on which stereotypical situations to represent, how to represent them, and which predictive features to associate with the stereotypes. It demonstrates how the assumption of a human in the problem-solving loop relaxes certain representational requirements and enables a new kind of feature acquisition, but also how it places additional requirements upon the choice and representation of cases or stereotypes. These theoretical considerations are described against the background of our work in progress on IOPS (Irregular Operations Planning System), an intelligent assistant for the task of airline irregular operations scheduling.

Intelligent assistant systems

The goal of an intelligent assistant systems is to assist the human user in detecting, diagnosing, and analyzing problems and in generating, selecting, and implementing solutions. An assistant might help by performing any or all of the following functions:

- The assistant might perform some specific computation, calculation, or inference at the behest of the human problem solver. A trivial example here is an electronic calculator; a less trivial example is a simulator that lets the user predict the results of some action. Note that the user chooses what calculation to perform and when to perform it; the system responds to specific user requests.
- The assistant might store information that the user would otherwise need to memorize, and provide it to the user at the appropriate time. A trivial example is an on-line reference manual, a more complex example is the "ask-" series of systems [Schank, 1991]. Here, the system provides information in response to the user's request; it is up to the human to interpret the relevance of the information to the current situation.
- The assistant might spontaneously advise the human user. If the system has access to a description of the current situation, the system may detect the applicability of one or more of its stored problem-solving strategies, and suggest it or them to the user.
- The assistant might request additional information from the user, prompted by a need to discriminate among competing strategies to apply to the current situation, or by a need to discriminate among competing hypotheses to explain the origin of the current problem. The behavior of medical diagnostic reasoning systems (e.g. [Shortliffe, 1976]) in suggesting

appropriate laboratory tests is typical of this activity.

- The assistant might perform some of the bookkeeping necessary to help the user carry out a plan. If the user selects a particular abstract, high-level problem solving strategy (e.g. repair a schedule failure by substituting one resource for an unavailable one), the system can fill in some of the details (selecting an appropriate resource to substitute, tracking the state of the old and new resource, etc.)

In a more sophisticated system, combinations of these behaviors are possible. The system might, for example, detect the applicability of several of its problem-solving strategies, request additional information to determine which few are most applicable, partially predict the results of implementing each of the strategies, and present the set of choices to the user. Once the user selects one of the strategies, the system can implement it and update its model of the state of the world.

Underlying each and all of these behaviors is the system's critical need to learn new problem classes, repair strategies, and descriptive features as it interacts with the user and acquires more knowledge about the problem-solving domain. Recent work at Chicago [Hammond, 1992] has described a life cycle of "apprentice" to "assistant" to "advisor" as the system acquires knowledge about the domain, and spends less time asking the human partner questions and more time offering advice and suggestions.

Case-based planning

One mechanism for solving problems is by noticing and exploiting the similarities between the current situation and a case — either a specific prior experience or a commonly recurring stereotype — selected from memory. (See, e.g. [Alterman, 1986; Bareiss, 1989; Barletta and Mark, 1988; Hammond, 1989; Kolodner and Simpson, 1989; Schank, 1982] for descriptions of typical case-based problem solvers). For the purposes of this paper, the task of an autonomous case-based planner can be described as:

- **Describe** the current problem in terms of the system's indexing vocabulary,
- **Retrieve** from memory a case whose stored problem description matches the description of the current problem,
- **Analyze** the differences between the current situation and the retrieved situation, and
- **Modify** the solution stored with the old problem description to fit the current situation.
- **Store** the new solution with a description of the current problem, placing particular representational focus on the features that differentiate the current situation from the old case retrieved from memory.

A commonly-articulated argument for the case-based approach is that it is particularly appropriate in situations where a system cannot reasonably proceed by chaining through the problem-solving operators that one would expect to find in a complete descriptive theory of the domain — either because a complete domain theory is unavailable, or because access to it is expensive, where not enough is known about the current situation to precisely determine the applicability of the theory's operators, or where the operator space of the domain theory is so large that the computational costs of searching it are prohibitive. Several of these features characterize the airline irregular operations domain, described below.

When dealing specifically with intelligent assistant systems as opposed to autonomous systems, several more arguments come into play:

- **Case-based advice:** Suggesting relevant cases is an effective mechanism for the system to offer advice to the user. Even if the system lacks sufficient inference capabilities to autonomously derive an appropriate sequence of actions by transforming the case to fit the current situation, it can still assist by presenting the case to the user, and exploiting the user's ability to apply the case.
- **Case-based knowledge acquisition:** The task of case retrieval forms a natural mechanism to control the system's requests for information from the user. The system in effect plays a kind of "20 questions" game with the user, asking for additional descriptive information about the current situation only when that information would play a clear role in discriminating among multiple cases, each of which potentially applies.
- **Case-based feature learning:** The systems retrieval failures (i.e. inability to discriminate between different cases) provides an opportunity to acquire new elements of a descriptive vocabulary from the user. The system can ask, in effect, "How do these two cases differ" and the user can provide, and name, a new descriptive feature.

Case-based knowledge acquisition and case-based feature learning are described further below, in the context of our ongoing work on IOPS, which solves problems in the domain of dynamic schedule repair.

IOPS

Airline irregular operations

The problem solving domain of this research is *airline irregular operations scheduling*. An airline wants to meet anticipated passenger demand over the routes it flies with an efficient allocation of its capital and human resources. To this end, it develops an operations schedule: an assignment of aircraft to scheduled flight operations and scheduled maintenance stops, and of crew to flight legs and rest periods. The schedule is carefully optimized to achieve efficient utilization

and distribution of aircraft and crew over the airline's routes.

Unfortunately, schedule disruptions due to weather, traffic congestion, unscheduled equipment maintenance, crew illness, or unanticipated requests for charters or other additional flight operations are inevitable but unpredictable. Because of the massive internal interdependencies inherent in an airline schedule, even a small single-point failure, such as an aircraft temporarily delayed for replacement of a burned-out light bulb, could potentially result in a snowballing sequence of downstream delays, disruptions, and missed connections if actions were not taken to mitigate the consequences of the failure.

To deal with these unexpected events, airlines employ operations controllers: experienced individuals whose job it is to monitor the airline's flight operations and to take steps to minimize passenger delay and inconvenience and cost to the airline. The controllers have access to information about the airline's current and planned operations, and knowledge of current and forecast conditions. Based on the information they receive, they order changes to the airline's operating schedule in an attempt to mitigate the effects of unexpected disruptions.

A content theory

An essential set of decisions in the design of an case-based planner revolves around a **content theory** of the domain: determining what cases ought to be put in memory, what descriptive features ought to be part of the system's representational vocabulary, and how the system ought to extract descriptions of new situations so that those descriptions will be useful in determining the applicability of old cases to new situations. Typical criteria for selecting cases and indices are discussed in [Owens, 1991], [Owens, 1990], and [Birnbaum *et al.*, 1989].

Our initial content theory of failure and repair in the irregular operations domain derives from our observation of several experienced operations controllers over multiple sessions as they detect, diagnose, and solve problems. The result of this analysis has been:

- to categorize, to the extent possible, the different **classes of problems** that the controllers are asked to solve. Examples of such categories include:
  - Unscheduled maintenance delay at a hub airport during peak travel time.
  - Weather-induced bottleneck at a non-hub airport.
  - Traffic congestion restricting outbound flights from a non-hub airport.
- to identify the **primitive operators** the controllers have at their disposal. Examples of primitive operators include:
  - Cancel a flight segment
  - Advance or delay the departure time of a flight segment
  - Add an unscheduled stop to a flight, or skip a scheduled stop.
  - Substitute one aircraft for another
  - Divert a flight to a different destination
  - Ferry an empty aircraft from one airport to another.
- to identify the higher-level **strategies** that the controllers use to solve problems. Higher-level strategies are built from sequences of primitive operators, and they appear to address goals such as:
  - Localize a problem: prevent a disruption at one airport from propagating to the rest of the system, e.g. by rerouting flights around the affected airport.
  - Distribute the impact of a problem, e.g. create small delays across the system to avoid a major bottleneck at one airport.
  - Delay the effects of a problem to increase the chance that an opportunistic solution will present itself, e.g. "borrow" an aircraft from a later flight to cover a shortfall on a current one; cover the later flight by borrowing yet a later aircraft, etc.

The importance of this content theory is that it defines the functional criteria for selecting a problem solver's case library, representation vocabulary, and indexing terms. The cases stored in the system should cover the classes of problems that experienced controllers appear to solve. The representation vocabulary should represent the features necessary to detect the applicability of those cases, and the primitive operators involved in the solutions. The indexing terms should be sufficient to discriminate between the existing cases in determining their applicability to new situations.

While this type of content theory is appropriate to the design of an autonomous problem solver, an additional set of criteria come to bear in the design of an intelligent assistant system.

Some representational problems are easier in a system that can count on a human user to help it diagnose and learn. Difficult tasks of detection and situation assessment, for which no good theory exists, can be deferred to the user. On the other hand, the presence of a human in the problem-solving loop also additional demands upon the system and upon its knowledge representation. Not only must the system's case library and descriptive vocabulary cover the range of expected problems and solutions, but it must do so in such a way as to facilitate communication of partial results and explanations to user, and to enable requests for additional information to be made and understood.

The IOPS system

The knowledge gathered from our observation of schedule controllers is being represented as the case library of IOPS, a case-based intelligent assistant for irregular operations scheduling currently under development.

In addition to the case library, IOPS has access to operating data regarding the current and planned state of an actual airline schedule, including the assignment of aircraft to flight legs, information about passenger loads, connections, and destinations, and the current and planned locations and movement of aircraft.

The function of IOPSis to provide the following kinds of assistance to the human problem solver:

- Given a description of a problem, like "Aircraft 2854 will be out of service for 1 hour for unscheduled maintenance", or "Bad weather is expected to close Denver for 4 hours this evening", use the description, plus all the current operating data, to select potentially applicable repair strategies.
- Given a set of potentially applicable repair strategies, ask the user for information that would discriminate among them. This is not only an opportunity to narrow the current search space, it is also an opportunity, as described below, to acquire new descriptive features about the domain.
- Given a repair strategy, selected either by the search process above or by the user, perform the computation necessary to implement it. If, for example, the strategy is "Delay a flight and use its aircraft to fill in for the temporarily unavailable one", then generate a list of potentially acceptable flights to delay.

Memory and learning for an intelligent assistant

A key role for a case memory in an intelligent assistant system is to direct the interaction between the user and the system. This can effect *short-term* knowledge acquisition, in which the system, over the course of a single problem-solving session, asks the user for help in detecting the presence or absence of abstract properties that it itself cannot detect, and in *long-term* knowledge acquisition, in which the system, through structured interaction with the user acquires new descriptive features for subsequent use in representing and indexing cases. In both short-term and long-term knowledge acquisition, A library of prior cases presents a baseline against which new descriptive information can be acquired from the user.

Short-term knowledge acquisition

Presenting potentially applicable cases to the user along with a request for clarification is a powerful mechanism for managing the interaction between the user and the system. Since, in a cooperative problem-solving context the system cedes some of the feature detection responsibility to its human partner, a mechanism is necessary for the system to request the information it needs. The system cannot simply ask the user "Tell me something about the current situation" — the question is too open-ended.

On the other hand, if the system has a partial description of the current situation, it can use its case library to request additional information. It can accomplish this by retrieving cases that match the current situation, comparing them with each other to identify descriptive features that, if their presence or absence could be determined in the current situation, would discriminate among the potentially-matching cases. The system can then ask the user about the missing features.

For an example from the airline irregular operations scheduling domain, consider a system trying to repair a schedule whose partial description indicates that 5 off-peak flights inbound to a hub airport are reported delayed by over 30 minutes each. The system, searching its library of cases matching these descriptors, finds multiple possible matches. One feature that discriminates between the matches is that some of the prior cases cover situations in which the weather was deteriorating and others covered situations in which the weather was not deteriorating. Although the system lacks sufficient data and inference capability to determine whether or not the weather is deteriorating at the affected airport, it can ask the user and, based upon the user's response, further narrow the search space of relevant cases.

Long-term knowledge acquisition

In contrast to using its case base to acquire knowledge about the current situation as described above, an intelligent assistant system can also use its case base as a mechanism for acquiring new descriptive features about the domain by asking the user. Again, this information can be requested in the context of a failure to find a prior case that satisfactorily matches the current situation. This failure can manifest itself either as:

- The system's inability to find any case that matches the current situation on the basis of the existing description, or
- The system's "best match" case being rejected by the user as inappropriate to the current situation, or
- The system retrieving two or more cases that match the current situation, and the system being unable to identify any features which, if known about the current situation, would discriminate among the cases.

The first manifestation represents a fundamental lack of cases in the system's library; the solution here is not in principle different from the solution adopted by autonomous case-based reasoners: Solve the problem from first principles and store the solution as a case. (The difference being, for an autonomous agent, solving from first principles can include letting the human partner solve the problem.)

The second two manifestations of retrieval failure can be dealt with by requesting the user to identify a new descriptive feature not previously known to the system. The system asks, "As nearly as I can tell, this

case exactly matches the current situation, but you don't seem to agree. Please identify a feature that is present in the current situation and absent in the case, or vice versa."

An example of this feature acquisition strategy is taken from our observations of the airline controllers, in which the observer played the role of the intelligent assistant system. The problem being solved is a shortage of baggage cannisters in Toronto:

> Controller: *I'm going to order Vancouver to put some extra empty baggage cannisters onto the next flight to Toronto.*
>
> Observer: *The strategy I know about for solving this problem is to fly in some baggage cannisters from the closest airport having frequent flights to the affected airport and having excess baggage cannisters available. In this case, that might be Detroit or Chicago. Why isn't that appropriate here?*
>
> Controller: *Detroit to Toronto involves crossing a national boundary. Shifting assets across national boundaries involves an additional delay and paperwork expense associated with Customs. In this case, the added time to ship them from Vancouver is not significant relative to these administrative delays and costs.*

At this point, an actual system could have used the interaction to acquire a specific new descriptive feature that was not previously part of the domain theory, that feature being whether or not two stations were in the same country. In future problem-solving sessions, the system could use the new feature to characterize situations and decide among potentially applicable cases.

Naming and detection

Simply because the system has learned about a new feature from the user (like whether or not two cities are in the same country, as above) does not, of course, mean that the system has developed an inference mechanism for detecting whether or not that feature applies in any given situation. While this would likely be an insurmountable problem for an autonomous problem-solver, it is less of a problem for an intelligent assistant because it retains the option of asking the user whether or not the feature describes the current situation.

But this easing of the detection task imposes an additional functional constraint on descriptive features: that they be nameable or describable so as to make communication possible with the user. When the system acquires a new descriptive feature, it must also acquire a name for that feature or a mechanism for describing it so that it can, in the future, ask that user (and other users) whether or not the feature characterizes situations.

Similarly, the system and the user must be able to communicate about the system's cases. It is unreasonable for the system to ask "Do you think `CASE-T0073` is relevant here?", but it is probably also unreasonable to require the system to describe every detail of a case in order to refer to it. This naming and reference problem remains an open one.

Evaluation

Evaluating hybrid systems involving a human-computer partnership is difficult. While there are good objective measures for success in the airline irregular operations domain (typically some function taking into account passenger delay and inconvenience and cost to the airline), IOPS's goal is not to solve problems autonomously. Consequently, the evaluation question is not "Which classes of problems does the system solve?" or "How well (quickly, effectively, cheaply) does it solve these problems?" For the scientific issues discussed in this paper, the evaluation questions are:

- Are the stereotypical situations represented here an appropriate set?
- Are they well represented?
- Are the features used for detection and diagnosis appropriate?

There are several bases on which to evaluate the goodness of the case library and representation/indexing vocabulary:

Are the failure types and repair strategies meaningful? Do experienced controllers recognize them? Can controllers readily answer the question "Is this failure characterization appropriate to the current situation?"

Are the failure types and repair strategies useful to an individual? Can a controller, using the IOPS, develop better solutions than without the system? Or can the controller consider more solutions in a given time, or develop solutions more quickly?

Are the failure types and repair strategies useful across individuals? Can one controller use failure diagnoses and repair strategies developed by observing the behavior of another? Can the failure categories and repair strategies be named or otherwise presented to make this process easier?

Are the failure types and repair strategies an effective mechanism for transferring knowledge? Can novices use the system to obtain results that approximate the results obtained by experts? Can novices learning anything useful about the domain by interacting with the system?

Conclusions

A crucial set of issues in the design of a case-based reasoner revolves around the question of what kind of representation vocabulary should be used to describe the system's cases, and what descriptive features should form the basis for judging the applicability of cases to new situations. The basis for resolving these questions remains one of function: A case-based reasoner whose job it is to repair schedule failures, for example,

should describe, categorize and index cases based upon the failures it is able to detect and upon the repairs it is able to perform.

The assumption of a human partner in the problem-solving loop alters these functional criteria in specific ways. While the system may be partially relieved of the burden of detecting every failure itself, it takes on the burden of extracting from the human user an operational description of those failures that the human has detected. While the system may be partially relieved of the burden of ultimately deriving a solution from a retrieved case, it takes on the burden of communicating a partial solution or describing a potentially relevant case. And, if the system is to acquire domain knowledge from the user in the form of new descriptive features, it needs a mechanism for communicating with the user about the success or failure of matches, and for using the failures to prompt the user for new descriptive features.

Another characterization of the functional requirements that derive from a case-based system's interaction with a human partner is to add more tasks to the existing *retrieve*, *debug*, *modify* and *apply* that lie at the core of case-based problem-solving. These new tasks involve asking the user about the presence or absence of a feature in the current situation, asking about the applicability or non-applicability of a case, and asking for a name and sketchy description for a new descriptive feature previously unknown to the system.

Acknowledgments

This work is supported in part by the Air Force office of Scientific Research under contract AFOSR-91-0112, and in part by the Defense Advanced Research Projects Agency and Rome Laboratory under contract F30602-91-C-0028. The author gratefully acknowledges the assistance of United Airlines in providing data and observer access to live operations. Nothing in this paper represents any policy, position, or opinion of United Airlines. John Borse has been an active participant in the data gathering and ongoing system development described in this paper.

References

[Alterman, 1986] R. Alterman. An adaptive planner. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 65–69, Philadelphia, PA, August 1986. AAAI.

[Bareiss, 1989] Ray Bareiss. *Exemplar-Based Knowledge Acquisition*, volume 2 of *Perspectives in Artificial Intelligence*. Academic Press, San Diego, CA, 1989.

[Barletta and Mark, 1988] R. Barletta and W. Mark. Explanation-based indexing of cases. In J. Kolodner, editor, *Proceedings of a Workshop on Case-Based Reasoning*, pages 50–60, Palo Alto, 1988. Defense Advanced Research Projects Agency, Morgan Kaufmann, Inc.

[Birnbaum *et al.*, 1989] Lawrence Birnbaum, Gregg Collins, and Bruce Krulwich. Issues in the justification-based diagnosis of planning failures. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 194–96. ONR/NSF, 1989.

[Hammond, 1989] Kristian Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*, volume 1 of *Perspectives in Artificial Intelligence*. Academic Press, San Diego, CA, 1989.

[Hammond, 1992] Kristian J. Hammond, editor. *Agency: Planning and acting in a dynamic world.* 1992. In preparation.

[Kolodner and Simpson, 1989] Janet L. Kolodner and Robert L. Simpson. The mediator: Analysis of an early case-based problem. *Cognitive Science Journal*, 1989.

[Owens, 1990] Christopher Owens. *Indexing and retrieving abstract planning knowledge.* PhD thesis, Yale University, 1990.

[Owens, 1991] Christopher Owens. A functional taxonomy of abstract plan failures. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 1991.

[Schank, 1982] R.C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, 1982.

[Schank, 1991] R. C. et al Schank. Ask (incomplete citation). Technical report, Institute for the Learning Sciences, Northwestern University, 1991.

[Shortliffe, 1976] E.H. Shortliffe. *Computer-based medical consultations: MYCIN.* American Elsevier, New York, 1976.