Using Advice to Influence Problem-Solving

Karen L. Myers
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com

Abstract

Difficulties in developing fully automated AI problem-solving systems have prompted interest in interaction methodologies that support increased user involvement in the solution-generation process. This paper considers the merits of interactive problem solvers grounded in the metaphor of *advice-taking*. An advice-taking system allows users to specify high-level characteristics at runtime of both the desired solution and the problem-solving process to be employed for the task at hand, with the underlying problem-solver applying those directives to guide solution construction. Advisability enables users to influence the behavior and output of problem-solving systems by making recommendations in terms that are natural and meaningful for them. This paper presents a model of advice-taking and characterizes applications for which advisability is best suited. In addition, it describes an ongoing effort to develop an *advisable planner* that marries an advice-taking interface to state-of-the-art planning technology.

Overview

In recent years, there has been growing recognition of the difficulty in building fully automated, stand-alone AI problem-solving systems. One consequence of this realization has been increased interest in semi-automated paradigms that facilitate user participation in the problem-solving process. These paradigms are distinguished both by the specific problems that they address (*e.g.*, control of problem-solving, the acquisition of strategic knowledge) and the models of user involvement that they employ.

Machine learning of problem-solving knowledge represents one important paradigm. In this paradigm, the steps undertaken by a user in solving a given problem are observed and analyzed to produce a generalized theory of how to solve related tasks. The user does not contribute explicitly to the knowledge acquisition process but rather serves as an indirect source of information. Furthermore, the user is a 'temporary' knowledge source in the sense that the overall objective is to acquire sufficient problem-solving knowledge so that human participation can be eliminated entirely.

The paradigm of *intelligent assistance* (IA) adopts a markedly different role for users. In the IA model, a user solves a problem in conjunction with an automated assistant who monitors the activities of the user, providing guidance (such as the identification of problems in individual user actions or the automated validation of a proposed solution) where appropriate. Humans are essential to the problem-solving process within the IA paradigm — automated assistants will never replace them.

This paper focuses on the paradigm of *advice-taking*. An advice-taking system allows users to specify high-level characteristics at runtime of both the desired solution and the problem-solving process to be used for the task at hand. The underlying problem-solver employs these directives to guide the search for a solution to the assigned task. Advice-taking enables users to provide guidance to the problem-solver in terms that are natural and meaningful for them, resulting in increased user influence on the solutions produced.

Advice-taking and intelligent assistance share the tenet that users should play a substantive role in the problem-solving process but the two paradigms differ in the nature of that involvement. In essence, IA embodies a philosophy of *advice-giving* rather than *advice-taking*. Furthermore, the IA paradigm treats the user as the driver of the problem-solving process, whereas the advice-taking paradigm puts the user in more of a supervisory role. As such, the advice-taking paradigm does not impose as much of a problem-solving burden on the user. The paradigm of machine learning of problem-solving knowledge is essentially dual to that of advisability: while machine learning seeks to generalize from concrete, individual solutions to abstracted problem-solving knowledge, advice-taking converts high-level descriptions of partial solutions into complete, low-level solutions to a problem.

The advice-taking paradigm has received little attention in recent years, despite its great potential for improving the accessibility and usability of AI problem-solving systems. This paper discusses sev-

eral of the advantages of an advice-based interface for problem-solvers and attempts to characterize the class of applications for which advice-taking is both valuable and viable. The abstract describes an ongoing project focused on adding an advice-taking interface to an AI planning system, in order to illustrate some of the issues involved in designing advisable problem-solvers. Finally, several open questions related to interactive problem-solving are presented that have particular relevance for advice-based interfaces.

Advisable Problem-Solvers

We view advice-taking as a process by which a user interacts with an automated problem-solving system in order to guide and influence the solution-generation process. Advice consists of specifications of desired characteristics of both the solution and the problem-solving process for a given task, generally expressed in high-level, abstract terms. The underlying problem-solver is responsible for formulating solutions that satisfy the task-specific constraints imposed by user-supplied advice.

Advice-taking augments the capabilities of the underlying problem-solving system in the sense that the system does not require advice for its operation. Rather, advice simply influences the set of solutions that the system will provide for a given task.

The advice-taking paradigm supports user provision of significant, high-level input to the problem-solving process. This design has several advantages over a fully automated approach.

Increased User Influence Users in many application areas are reluctant to cede responsibility of problem-solving to a fully automated system. A solution generated by a black-box module is unacceptable for several reasons. First, users often resist a solution when they do not understand the process by which it was derived. Second, there is often no single 'correct' solution to a problem. Rather, users want to customize solutions to match their individual needs and desires. For example, military campaign plans generally reflect the biases and preferences of the commanders in charge, despite the fixed doctrine that underlies much of the planning process (Thaler & Shlapak 1995).

Simplification of Domain Modeling Success for many AI problem-solving methods requires complete and correct knowledge of the targeted application domain. Providing this information is daunting (if not impossible) for even moderately sized applications. The advice-taking paradigm eliminates the need for complete and error-free domain knowledge by enabling users to interact with the problem-solver during the search for a solution. This interactive approach is imperative for most applications, as it is impossible to replicate fully the domain and commonsense knowledge of human experts.

Solution Quality For most complex AI applications, it has proven extremely difficult to elicit and quantify explicit evaluation metrics. For this reason, few AI systems employ meaningful evaluation criteria to guide the problem-solving process. The advice-taking framework allows users to provide evaluations directly, through the prescription and rejection of various approaches and partial solutions. As such, the advice-taking model makes it easier to exploit the insights and knowledge that users have accumulated from years of domain experience, leading to greatly improved solution quality.

Efficiency The constraints and strategies introduced by user-supplied advice can, in some cases, reduce the time required to solve a given problem. For example, a user-supplied sketch of a solution for a planning problem might eliminate a large portion of the search space that would otherwise have to be explored by a generative planner.[1]

The Advisable Planner

We have recently embarked on a three-year effort to make AI planning technology more accessible and controllable through the integration of an advice-taking interface with an automated planning system (the SIPE-2 generative planner (Wilkins 1988)). The resultant *advisable planner* (AP) will accept a variety of instructions and advice from a user and employ those directives to guide plan construction. Advice will include partial sketches of plans or subplans for achieving a set of goals, specific subgoals to be used in pursuit of the overall objectives, restrictions and prescriptions on the use of specific objects and actions, and desired attributes of the final solution. These advice-taking capabilities will provide users with the means to influence the shape of the plan being generated for a particular set of goals, enabling them to participate in the planning process in a meaningful way.

Motivation

The AP project was motivated in large measure by the lack of success to date in transitioning AI planning technology to appropriate user communities. A major reason for the lack of technology transfer stems from the difficulty of using planning systems. AI planners have traditionally been designed to operate as 'black-boxes': they take a description of a domain and a set of goals and automatically synthesize a plan for achieving those goals. This design explicitly limits the amount of influence that a user can have on the generated plans. Furthermore, it requires complete and accurate formalizations of the domain, since the system is expected to operate without user intervention. Providing such

[1] Advice does not always improve the efficiency of problem-solving. In fact, it has been shown that the use of initial plan sketches can increase the complexity of planning tasks under certain conditions (Nebel & Koehler 1995).

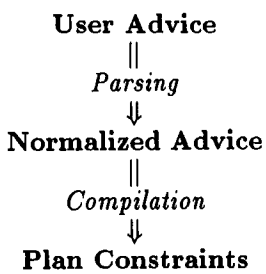

Figure 1: Advice Representation and Translation

comprehensive domain information is time-consuming and expensive, and represents a significant investment for each new application.

Recent trends toward *mixed-initiative* styles of planning have led to support for certain low-level interactions on the part of humans, such as ordering goals for expansion, selecting operators to apply, and choosing instantiations for planning variables (Tate, Drabble, & Kirby 1994; Drabble & Tate 1995; Wilkins 1993). While a step in the right direction, these interactions are too fine-grained to suit users, who generally want to be involved with the planning process at a higher, more strategic level.

The AP model of planning embodies a shift in perspective on how planning systems should be designed: an automated planner is viewed as a tool for enhancing the skills of domain users, not as a replacement for them. The Advisable Planner will enable a user to play a more substantial role in the plan generation process. In particular, the user will be able to interact with a planning system at high levels of abstraction in order to guide and influence the planning process, with the planning system performing the time-consuming work of filling in necessary low-level details and detecting potential problems.

AP Architecture

Overall, the AP contains two distinct phases: the *advice translation* phase, and the *problem-solving* phase.

Advice translation involves mapping from user-supplied advice into appropriate internal representations for the planner. The translation process involves several stages, as illustrated in Figure 1. *User advice*, specified in some natural (or pseudo-natural) language is parsed into an intermediate normalized representation. The normal form provides a planner-independent representation of the advice, thus enabling a clean semantic definition and portability amongst different planners. *Advice compilation* is the planner-dependent translation from normalized advice to internal constraints defined in terms of planner-specific operators, goals, and objects. These constraints will be used to direct the plan construction process.

The problem-solving phase takes the compiled advice representations, and generates only solutions that satisfy the advice (referred to as *advice enforcement*). Planning proceeds in a mixed-initiative style whereby the user can make planning-time requests to modify current plans or previously stated advice. The AP may request additional domain information from the user during planning to aid in resolving detected trade-offs in the plan, to recover from planning failures, or to clarify user-supplied advice.

The main thrusts of the AP project are (1) to identify advice idioms for planning that are natural and expressive yet amenable to computationally efficient processing, and (2) to develop advice-sensitive planning algorithms that enforce compiled advice. The many interesting and important issues related to managing interactions with users (such as natural language understanding, dialog management, *etc.*) are beyond the scope of the project.

Basic Advice Idioms

We have identified three basic *advice idioms* that provide broad coverage of the kinds of influence required by users of planning systems.

Task Advice designates specific goals to be achieved and actions to be performed. As such, task advice amounts to partial specification of a solution to a planning task. Examples include Defend the Northeast Sector and Secure Air Superiority in Sector A before Air Superiority in Sector B. Sketches of partial plans constitute another form of task advice.

Strategic Advice consists of recommendations on how goals and actions are to be accomplished, for instance, Use an air-based strategy to defend the Northeast sector and Don't send troops through cities with populations greater than 100,000.

Evaluational Advice encompasses constraints on some metric defined for the overall plan. Common metrics include resource usage, execution time, and solution quality. For example, the directive Don't employ more than 5 sorties in Region H constitutes an example of evaluational advice.

Our model of advice does not include *control* of problem-solving, such as ordering goals for expansion, or choosing instantiations for planning variables (Currie & Tate 1991; Stefik 1981). While control is an important issue for automated planning systems, it is our view that effective control of the planning process requires deep insights into the mechanics of the planning system itself. As such, it is not a responsibility that should be borne by the user (who generally will not be an expert in AI planning) and thus is not appropriate as a topic for user advice.

Structured Advice

The basic idioms are inadequate to model the full range of influence over the planning process required by users. Greater influence can be attained by enriching

advice with additional structuring mechanisms. We describe several such constructs that the AP prototype will support.

One important structuring mechanism is the situationalization of advice to restricted *contexts*. Consider the statement

> Use a feint with strong air support if there are no other feints in progress.

This directive expresses a piece of strategic advice to be applied only in certain situations. Such contextualization of advice is critical to providing adequate expressivity for users. We model contexts in terms of the *problem-solving state* of the planner, using a *Belief-Desire-Intention* approach. *Beliefs* correspond to the expected world state at a given point in a plan, *Desires* correspond to the goals within a plan, and *Intentions* correspond to commitments to actions. This model supports surface-level constructs such as

"if X is true" *(Belief context)*
"to achieve X" *(Desire context)*
"when doing X" *(Intention context)*

Preferences constitute a second important structuring mechanism for advice. Users frequently have multiple, conflicting desires, possibly with situation-specific preferences among them. For instance, a user may want to minimize both the cost and execution time of a plan under construction; these two objectives generally conflict. A user may express conditions under which one objective is more important than the other; for instance, execution time should take precedence when it exceeds some threshold. The AP will enable users to express conditional preferences among advice. The system will identify sources of conflict when all advice cannot be satisfied and inform the user of the space of trade-offs available. This approach will enable the users to make informed choices about alternative solutions to pursue.

A third advice structuring mechanism to be supported in the AP is *advice rationale*, which captures a user's objective in asserting a piece of advice. For example, the advice

> Complete deployment in 10 days in order to protect Route 5.

includes the motivation for the basic advice that it expresses. Rationales will enable treatment of advice that is more likely to match the user's intent. For example, the temporal constraint in the above advice could be ignored if some other aspect of the plan ensures adequate protection for Route 5.

Technical Approach

In operational terms, advice is compiled into constraints that influence decisions made by the core planning technology. In addition to satisfying standard planning notions of goal/operator applicability, the associated advice constraints must also be satisfied. While traditional planning constraints are grounded in projections about the *world* state, advice constraints further include conditions on the the current plan and open goals (*i.e.*, the *problem-solving* state). This model provides great flexibility while supporting a broad range of advice constructs.

A plan and each of its nodes has an accompanying set of compiled advice constraints. We are developing advice-sensitive planning algorithms in which advice constraints accumulate during processing, being passed downward through the plan refinement structure. We refer to this style of planning as *constraint planning*, given the similarity to Constraint Logic Programming (CLP) (Jaffar & Lassez 1987): both consist of a problem-reduction search augmented with constraints on the overall structure being defined. For CLP, the constraints restrict instantiations of variables; for advisable planning, the constraints further restrict the choice of problem-reduction rules (*i.e.*, the operators to be applied).

Related Work on Advice-Taking

There have been few attempts to develop domain-independent advice-taking systems. The most significant work in this area is Mostow's FOO system, which provides a framework for advice-taking in the context of playing the card game Hearts (Mostow 1981). Mostow's work defines advice as a high-level description of a task for which there is no explicit method to achieve it. The work focuses on *operationalization* – the transformation of advice into directives that can be executed directly by the problem-solving system being advised. FOO employs a reductionist approach by which advice is successively rewritten (using both sound and heuristic methods) until it grounds out in executable actions. Its notion of advice-taking amounts to 'filling in' gaps in planning operators, in contrast to the AP approach of restricting how operators can be applied. Furthermore, the work does not address the issue of general-purpose advice languages suitable for human interactions, which is a key component of the AP effort.

Issues in Task Automation

The topic of task automation raises a number of open issues, some of which are of particular relevance to advice-taking systems.

1. *When are fully automated problem-solvers attainable? When are they desirable?*

   As noted above, fully automated problem-solving is neither desirable nor possible in all cases. Developers of AI technology need to understand better the limits of what can and should be automated in order to better focus their efforts. This issue will become more critical as researchers undertake larger-scale applications in the future.

2. *Is it possible to layer different interactive approaches on top of core problem-solving capabilities in a modular fashion?*

   Ideally, there should be a clean separation between a problem-solver and the methods used to interact with the problem-solver. It is unclear to what extent such modularity is possible for complex AI systems. For instance, the use of an advice-based interface for a planning system necessitates changes to the fundamental planning algorithms.

3. *When should domain knowledge and processing capabilities be built into a problem-solving system rather than left for users to specify?*

   For domain knowledge, it is necessary to take into account both the cost of soliciting the knowledge and encoding it in adequate representation structures, and the extent to which comprehensiveness and correctness can be ensured. Relevant issues for processing capabilities include computational efficiency and the availability of adequate domain knowledge.

4. *How can we bridge the gap between problem-solver semantics and user semantics?*

   Interaction tools must support communication with users either directly (*e.g.*, for advice-taking) or indirectly (*e.g.*, for machine learning of problem-solving strategies). The conceptual models employed by users often differ substantially from the computational models that underlie automated problem-solvers, making this communication difficult. For instance, individuals typically don't frame planning tasks in terms of operators with add and delete lists, which is the basic representational approach underlying most generative planners. For users to communicate with automated problem-solvers (and *vice versa*), it is necessary to provide a mapping between the differing models and their semantics.

5. *Can interaction tools be used effectively by non-experts?*

   Many current tools require users to have a deep understanding of the processing and models that underlie both the tool and the problem-solver. This knowledge prerequisite has made it difficult to transition AI problem-solving technology from research environments to appropriate user communities.

Conclusions

The notion of problem-solving systems that can take advice from humans has been around since the beginning of AI itself (McCarthy 1985). Despite the conceptual appeal, there has been little success to date in building automated advice-taking systems because of the intractability of the task in its most general form. We believe that the paradigm can be made tractable for specific classes of applications and tasks by grounding advice in a focused set of problem-solving activities. The Advisable Planner project presents a step toward this goal for the paradigm of generative planning.

Advice-based problem-solvers are well-suited to applications in which users want to influence solutions, or for which domain knowledge or problem-solving methods are too weak to enable efficient, fully automated generation of solutions. They provide users with the means to influence and direct the problem-solving process in meaningful ways. This contrasts sharply with the kinds of interactions currently supported in most problem-solving systems, which are at too low a level to be of value to users. Overall, the use of an advice-taking paradigm will make automated problem-solving systems more accessible and appealing to users by making it easier to generate high-quality solutions that are well-suited to their needs.

Acknowledgements

This research has been supported by ARPA Contract F30602-95-C-0259.

References

Currie, K., and Tate, A. 1991. O-Plan: the open planning architecture. *Artificial Intelligence* 32(1).

Drabble, B., and Tate, A. 1995. O-Plan mixed initiative planning capabilities and protocols. Technical report, University of Edinburgh.

Jaffar, J., and Lassez, J.-L. 1987. Constraint logic programming. In *ACM Symposium on Principles of Programming Languages.*

McCarthy, J. 1985. Programs with common sense. In Brachman, R. J., and Levesque, H. J., eds., *Readings in Knowledge Representation.* Morgan Kaufmann.

Mostow, D. J. 1981. *Mechanical Transformation of Task Heuristics into Operational Procedures.* Ph.D. Dissertation, Computer Science Dept., Carnegie-Mellon University.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence* 76:427–454.

Stefik, M. 1981. Planning and meta-planning. *Artificial Intelligence* 16(2).

Tate, A.; Drabble, B.; and Kirby, R. 1994. O-Plan2: An open architecture for command, planning and control. In Fox, M., and Zweben, M., eds., *Knowledge Based Scheduling.* Morgan Kaufmann.

Thaler, D. E., and Shlapak, D. A. 1995. Perspectives on theater air campaign planning. Technical report, Rand Corporation.

Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann.

Wilkins, D. E. 1993. *Using the SIPE-2 Planning System: A Manual for Version 4.3.* Artificial Intelligence Center, Menlo Park, CA.