Satisficing and bounded optimality

A position paper

Shlomo Zilberstein*
Computer Science Department
University of Massachusetts

It appears probable that, however adaptive the behavior of organisms in learning and choice situations, this adaptiveness falls far short of the ideal "maximizing" postulated in economic theory. Evidently, organisms adapt well enough to "satisfice"; they do not, in general, "optimize".

– Herbert Simon, 1958

Abstract

Since the early days of artificial intelligence there has been a constant search for useful techniques to tackle the computational complexity of decision making. By now, it is widely accepted that optimal decision making is in most cases beyond our reach. Herbert Simon's approach based on *satisficing* offers a more realistic alternative, but it says little on how to construct satisficing algorithms or systems. In practice, satisficing comes in many different flavors, one of which, *bounded optimality*, restores a weak form of optimality. This paper demonstrates this form of satisficing in the area of anytime problem-solving and argues that it is a viable approach to formalize the notion of satisficing.

Satisficing

In the pursuit of building decision-making machines, artificial intelligence researchers often turn to theories of "rationality" in decision theory and economics. Rationality is a desired property of intelligent agents since it provides a good evaluation criteria and since it establishes a formal framework to analyze agents (Doyle 1990; Russell and Wefald 1991). But in general, rationality requires making *optimal* choices with respect to one's desires and goals. As early as 1957, Simon observed that optimal decision making is impractical in complex domains since it requires one to examine and evaluate all the possible alternatives (Simon 1982). However, the vast computational resources required to select optimal actions often reduce the utility of the result. Simon suggested that some criterion must be used to determine that an adequate, or satisfactory, decision has been found. He has revived the Scottish word "satisficing" (=satisfying) to denote decision making that searches until an alternative is found that is satisfactory by the agent's aspiration level criterion.

Simon's notion of satisficing has inspired much work within the artificial intelligence community in the area of problem solving and search. It is by now widely accepted that in most cases the ideal (decision-theoretic) notion of rationality is beyond our reach. However, the concept of satisficing offers only a vague design principle that needs a good deal of formalization before it can be used in practice. In particular, one must define the required properties of a satisficing criterion and the quality of behavior that is expected when these properties are achieved.

One of the first approaches to satisficing has been heuristic search. In fact, Simon has initially identified heuristic search with satisficing. It is important to distinguish in this context between two different ways in which heuristics can be used. In many problem domains heuristic evaluation functions are used to substantially reduce the search space. Moreover, admissible heuristic functions allow such algorithms as A^* to always return the optimal answer. Admissible heuristic search is an important part of AI, but it has little to do with satisficing. The focus on optimal, rather than satisfying, solutions makes this type of heuristic search a more efficient way to find exact answers. Simon refers to another type of heuristic functions in which heuristics are used to select "adequate" solutions. This type of heuristic functions is rarely admissible or even near-optimal. Systems based on non-admissible heuristic functions are harder to evaluate, especially when optimal decisions are not available. Formal analysis is hard since non-admissible heuristics do not always have well-defined properties. The purpose of this paper is to propose an alternative approach to satisficing in which the notion of adequate

*Author's address: University of Massachusetts, Computer Science Department, LGRC, Box 34610, Amherst, MA 01003-4610. Email: shlomo@cs.umass.edu. Phone: (413) 545-4189. URL: http://anytime.cs.umass.edu/shlomo/.

solution has a well-defined meaning.

Bounded optimality

Bounded optimality techniques seek to restore some notion of optimality to decision making in the face of computational complexity and limited resources. That is, instead of building systems that find "sufficiently good" answers, the goal is to find "optimal" answers. Optimality, however, is defined with respect to a particular space of possible implementations.

Russell and Wefald (Russell and Wefald 1991) say that an agent exhibits bounded optimality "if its program is a solution to the constraint optimization problem presented by its architecture." Their approach marked a shift from optimizations over actions to optimization over programs. The program is bounded optimal for a given computational device for a given environment, if the expected utility of the program running on the device in the environment is at least as high as that of all other programs for the device. Russell, Subramanian, and Parr (Russell *et al.* 1993) give an efficient construction algorithm that generates a bounded optimal program for a restricted class of agent architectures, in which a program consists of a sequence of decision procedures. The decision procedures are represented using condition-action rules. The authors admit that bounded optimality as defined above may be hard to achieve for most problems. They propose a weaker notion of *asymptotic bounded optimality* as a more practical alternative. The latter case requires that the program performs as well as the best possible program on arbitrary hard problems if its computational device is faster by a constant factor.

In this paper, I propose a somewhat more general notion of bounded optimality. Bounded optimality is the selection of the best action subject to an *arbitrary* set of *architectural constraints*. Every agent operates within a particular architecture that often imposes constraints on the behavior and performance of the agent. The architectural constraints include the knowledge representation scheme, the base-level and meta-level inference mechanisms, and the action execution and monitoring techniques. Optimal or satisfactory action selection may be achieved within a given architecture for action selection, but the architecture defines the space of choices. These constraints limit the alternatives that an agent may consider. The alternatives available to the agent do not necessarily correspond to all the choices in the "real-world".

The special case in which the architectural constraints include *only* the description of the computational device and the environment, we get bounded optimality in the sense defined by Russell and Wefald. But for most problems, this goal is extremely hard to achieve.

Multiple levels of satisficing

Computer problem-solving can be viewed as simply running a particular program on a computational device in order to compute the "correct" answer to a problem. But developing a complex problem-solving system, such as an automated information gathering and extraction system, is a process that involves several stages before the actual system can be constructed. Satisficing can be applied (typically must be applied!) to each one of the problem-solving stages. The following list shows the four primary stages of problem-solving and the aspects of the problem that can be determined using a satisficing approach.

1. **Modelling the environment** (level of abstraction, background knowledge, the state space, the dynamics of the environment, Markov assumption)
2. **Modelling the problem** (the available inputs, the goals, the reward structure, timing and resource constraints)
3. **Modelling the computation** (the algorithms to be used, the data structures, the part of the problem that can be solved off-line, distributed versus centralized computation)
4. **Modelling meta-level control** (organizing the search process, meta-level knowledge, evaluating partial results and progress, centralized versus distributed control, stopping criteria)

Each one of the issues listed above will affect the behavior and performance of the resulting agent. Each one of these problems can be approached and solved using optimizing or satisficing techniques. Each stage involves complex tradeoffs that can be addressed off-line or on-line. For example, using a more precise model of the environment may complicate the problem definition and may force the system to compute less precise answers to the problem.

The key question is whether bounded optimality is a useful approach to all or some of these problems. In other words, the question is whether there are any advantages to making optimal decisions within an approximate model, rather than making approximate decisions within a more precise (or even perfect) model.

Simon argues that neither approach has an absolute advantage over the other. When comparing satisficing with optimizing, he claims that in complex real-world situations, optimization becomes approximate optimization since the description of the real-world is radically simplified until reduced to a degree of complication that the decision maker can handle. Satisficing seeks simplification in a somewhat different direction, retaining more of the detail of the real-world situation, but settling for a satisfactory, rather than approximate-best, decision. According to Simon, one cannot predict in general which approach leads to the better decisions as measured by their real-world consequences.

I argue that for certain aspects of problem-solving, bounded optimality has some methodological advantages over satisficing (in the more general sense). Bounded optimality forces the designer of the system to spell out the architectural constraints. Within those constraints, the system must make optimal decisions. So the justification of the actions selected by the system is reduced to examining the architectural constraints.

Example: anytime decision making

Fortunately, bounded optimality is not just desirable – it is a viable approach to satisficing. This can be shown by examining one general approach to bounded optimality based on composition and monitoring of anytime algorithms.

Methodologically, problem solving with anytime algorithms is based on dividing the overall problem into four key subproblems: elementary algorithm construction, performance measurement and prediction, composability, and meta-level control of computation.

Elementary algorithm construction covers the problem of introducing useful tradeoffs between computational resources and output quality in decision making. This fundamental problem has been studied by the AI community developing a variety of "anytime algorithms" (Dean and Boddy 1988) or "flexible computation" methods (Horvitz 1987) whose quality of results improves gradually as computation time increases. The same problem has been studied within the systems community in the area of "imprecise computation" (Liu *et al.* 1991). While iterative refinement techniques have been widely used in computer science, the construction of "well-behaved" anytime algorithms is not obvious. To serve as useful components of a resource bounded reasoning system, such algorithms should have certain properties: measurable objective output quality, monotonicity and consistency of quality improvement, and marginal decrease in the rate of quality improvement over time. Constructing good, reusable anytime algorithms is an important active research area.

Performance measurement and prediction covers the problem of capturing the tradeoff offered by each system component using a "performance profile". A good performance profile is a compact probabilistic description of the behavior of the component. A typical representation is a mapping from run-time to expected output quality. Recent results show that conditioning performance profiles on input quality and other observable features of the algorithm can improve the precision of run-time quality prediction.

Composability covers the problem of building modular resource bounded reasoning system with anytime algorithms as their components. The fundamental issue is that composition destroys interruptibility – the basic property of anytime algorithms. A two step solution has been developed to this problem that makes a distinction between "interruptible" and "contract" algorithms (Zilberstein 1993). Contract algorithms offer a tradeoff between output quality and computation time, provided that the amount of computation time is determined prior to their activation. The idea is to first compose the best possible contract algorithm and then make it interruptible with only a small, constant penalty (Zilberstein and Russell 1996).

Finally, meta-level control covers the problem of run-time allocation of computational resources (or "deliberation scheduling" (Dean and Boddy 1988)) so as to maximize the overall performance of the system. In general, meta-level control involves modeling both the internal problem solving process and the external environment and manage computational resources accordingly. In domains characterized by high predictability of utility change over time, the monitoring problem can be solved efficiently using contract algorithms and a variety of strategies for contract adjustment. In domain characterized by rapid change and a high level of uncertainty, monitoring must be based on the use of interruptible algorithms and the marginal "value of computation" (Russell and Wefald 1991). More recently, a new approach to monitoring has been developed that is sensitive to both the cost of monitoring and to how well the quality of the currently available solution can be estimated by the run-time monitor. The technique is based on modeling anytime algorithms as Markov processes and constructing an off-line monitoring policy based on a stochastic model of quality improvement (Hansen and Zilberstein 1996).

In what sense the methodology presented above is an example of bounded optimality? To answer this question, we need to identify the architectural constraints under which the system makes optimal decisions. The specific assumptions depend on the specific implementation, but they generally cover the following aspects:

- The computational resources that can be traded for increase in solution quality (time, memory, information, communication, etc.).
- The specific problems or sub-problems to be solved using an incremental/anytime approach.
- The type of performance profiles used to characterize the computational tradeoffs (probabilistic, conditional, dynamic, etc.).
- The approach to monitoring (myopic versus non-myopic, interruptible versus contract, etc.).

Subject to these *assumptions*, composition and monitoring can be formalized as optimization problems and in many cases they can be solved efficiently (Zilberstein 1996b). The overall approach is therefore an instance of bounded optimality.

Summary

Satisficing is an important principle that must be used in order to successfully solve complex decision problems. It is typically applies to different stages of problem solving, both by the designer of the system (off-line) and by the system itself (on-line). Although no single approach to satisficing is likely to be universally accepted, one particular approach – bounded optimality – appears to have some advantages. Bounded optimality forces the designer of the system to spell out the architectural constraints. Within those constraints, the system must make optimal decisions. The applicability of this approach is evident by recent progress in the area of anytime problem solving.

Acknowledgments

Support for this work was provided in part by the National Science Foundation under grants IRI-9624992 and INT-9612092.

References

T. Dean and M. Boddy. An analysis of time-dependent planning. *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 49–54, Minneapolis, Minnesota, 1988.

J. Doyle. Rationality and its roles in reasoning. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 1093–1100, Boston, Massachusetts, 1990.

E. A. Hansen and S. Zilberstein. Monitoring the progress of anytime problem solving. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 1229–1234, Portland, Oregon, 1996.

E. A. Hansen, S. Zilberstein, and V. A. Danilchenko. Anytime heuristic search: First results. Technical Report 97-50, Computer Science Department, University of Massachusetts, 1997.

E. J. Horvitz. Reasoning about Beliefs and Actions under Computational Resource Constraints. *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, Washington, 1987.

J. W. S. Liu, K. J. Lin, W. K. Shih, A. C. Yu, J. Y. Chung and W. Zhao. Algorithms for scheduling imprecise computations. em IEEE Computer, 24:58-68, 1991.

S. J. Russell, D. Subramanian and R. Parr. Provably bounded optimal agents. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 338–344, Chambery, France, 1993.

S. J. Russell and E. H. Wefald. *Do the Right Thing: Studies in limited rationality.* Cambridge, Massachusetts: MIT Press, 1991.

H. A. Simon. *Models of bounded rationality, Volume 2.* Cambridge, Massachusetts: MIT Press, 1982.

S. Zilberstein. Operational Rationality through Compilation of Anytime Algorithms. Ph.D. dissertation, Computer Science Division, University of California at Berkeley, 1993.

S. Zilberstein. Resource-bounded sensing and planning in autonomous systems. *Autonomous Robots*, 3:31–48, 1996.

S. Zilberstein. The use of anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

S. Zilberstein and S. J. Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82:181–213, 1996.