3

Metareasoning and Bounded Rationality

Shlomo Zilberstein

This chapter explores the relationship between computational models of rational behavior and metareasoning. Metareasoning is generally considered a crucial component of human intelligence, but its role in computational models of intelligence is less prominent. We describe several approaches to building rational agents and examine the extent to which they rely on metareasoning. While metareasoning is a central component of some approaches, it is not required in others. Despite these differences, we point out an interesting way to reinterpret and unify two of the approaches.

In the pursuit of building decision-making machines or agents, artificial intelligence (AI) researchers often turn to theories of "rationality" in philosophy, decision theory, and economics. According to these theories, an agent is rational when it chooses actions that maximize its performance, given what it currently knows. Rationality is a desired property of intelligent agents since it provides well-defined normative evaluation criteria and since it establishes formal frameworks to analyze agents (Doyle 1990; Russell and Wefald 1991).

An agent is said to be perfectly rational if it chooses optimal actions that maximize its expected performance. Perfect rationality defines the actions that should be taken, but it tells nothing about the reasoning process that leads to selecting these actions. The reasoning process can be as simple as a table lookup that specifies which action should be taken in every situation, or it may involve a complex analysis of the situation and planning. In that sense, perfect rationality does not explicitly require metareasoning. In fact, it may require no reasoning at all.

Ignoring the reasoning process used to select actions—and its associated costs—is a significant drawback of perfect rationality. As early as 1947, Herbert Simon observed that optimal decision making mandated by perfect rationality is impractical in complex domains since it requires one to perform intractable computations within a limited amount of time (Simon 1947, 1982). Moreover, the vast computational resources required to select optimal actions often reduce the utility of the result. Simon concludes that "a theory of rationality that does not give an account of problem solving in the face of complexity is sadly incomplete. It is worse than incomplete; it can be seriously misleading by providing 'solutions' that are without operational significance." Simon suggests that some criterion must be used to determine that an adequate, or satisfactory, decision has been found. He uses the Scottish word "satisficing," which means 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 social sciences and within AI in the areas of problem solving, planning, and search. In the social sciences, much of the work has focused on developing descriptive theories of human decision making (Gigerenzer 2000). These theories attempt to explain how people make decisions in the real world, coping with complex situations, uncertainty, and limited amounts of time. The answer is often based on a variety of heuristic methods that are used by people to operate effectively in these situations (Gigerenzer et al. 1999). Work within the AI community has produced a variety of computational models that can take into account the computational cost of decision making (Dean and Boddy 1988; Horvitz 1987; Russell, Subramanian, and Parr 1993; Wellman 1990; Zilberstein 1993). The idea that the cost of decision making must be taken into account was introduced by Simon and later by the statistician Irving Good who used the term *type II rationality* to describe it (Good 1971). Good says, "when the expected time and effort taken to think and do calculations is allowed for in the costs, then one is using the principle of rationality of type II." But neither Simon nor Good presents any effective computational framework to implement "satisficing" or "type II rationality."

It is by now widely accepted that in most cases the ideal decision-theoretic notion of perfect 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. AI researchers have introduced over

the years a variety of computational models that can be seen as forms of bounded rationality. We examine these models, divide them into four broad classes, and identify the role of metareasoning in each class.

Computational Approaches to Bounded Rationality

There has been a vast amount of work on bounded rationality in the social sciences, decision theory, and AI. This chapter focuses on computational approaches developed by the AI community. What is common to all these approaches is that they perform some form of approximate reasoning. They differ in the way the approximate solution is produced and evaluated.

Regardless of the form of approximation, approximate reasoning techniques can be complemented by some form of explicit or implicit metareasoning. Metareasoning in this context is a mechanism to make certain runtime decisions by reasoning about the problem solving–or object-level–reasoning process. This can be done either explicitly, by introducing another level of reasoning as shown in figure 1.2 in Cox and Raja's introduction, or implicitly, by precompiling metareasoning decisions into the object-level reasoning process at design time. For example, metareasoning has been used to develop search control strategies—both explicitly and implicitly. Thus, metareasoning could play a useful role in certain forms of approximate reasoning, but it is not by definition a required component. In the rest of this section, we examine several approaches to bounded rationality and divide them into four broad classes: heuristic search, approximate modeling, optimal metareasoning, and bounded optimality. We start by describing these classes and the role of metareasoning in each.

Heuristic Search

One of the early computational approaches to bounded rationality has been based on heuristic search. In fact, Simon had initially identified satisficing with a particular form of heuristic search. In this context, heuristic search represents a form of approximate reasoning. It uses some domain knowledge to guide the search process, which continues until a satisfactory solution is found. This should be distinguished from optimal search algorithms that use admissible heuristic techniques such as A*. Search processes that terminate only when they find an optimal solution are an important part of AI, but they have little to do with bounded rationality. When the search process focuses on optimal, rather than

satisfying, solutions the role of heuristics is simply to accelerate the search process by pruning certain parts of the search space from consideration. Simon refers to another type of heuristic function in which heuristics are used to select "adequate" solutions. Such heuristic functions are rarely admissible, and the corresponding search processes are not optimal in any formal sense. Systems based on nonadmissible heuristic functions are often harder to evaluate, especially when optimal decisions are not available. Formal analysis is hard since nonadmissible heuristics do not always have well-defined properties.

Approximate reasoning using heuristic search is a general paradigm, not a specific framework for problem solving. Therefore, it is hard to pinpoint the role of metareasoning. What is clear is that some instances of this paradigm rely on some forms of metareasoning, for example, in order to select the appropriate heuristic for the situation, decide whether the heuristic solution found so far is of sufficient quality, or fine-tune search parameters to try to maximize solution quality within some deadline (Hansen, Zilberstein, and Danilchenko 1997). Other instances of this general paradigm do not rely on metareasoning. Overall, metareasoning is not an essential component of every heuristic search approach.

Approximate Modeling

When it is not feasible to fully model a problem and solve it optimally, an important aspect of the approximation process is embedded in the creation of a suitable model. The challenge is to create a model that retains the main features of the original problem, but is computationally tractable. The hope is that an exact or approximate solution to the simplified problem would still be of similar quality when applied to the original domain. The process of reasoning about the representation of the given problem and choosing a suitable model is a form of metareasoning. Although there has been significant interest in automating this process, it is often handled by the designer of the system using forms of metareasoning that are not yet well understood. One example of such a process is when deterministic action models are used in planning, ignoring the uncertainty about action failures. Combined with suitable run-time execution monitoring, such an approach could be beneficial. In fact, the winner of the International Probabilistic Planning Competition in 2004 was a planner (FF-rePlan) based on these principles. Consequently, the general approach has gained significant attention, and some researchers have been tempted to conclude that probabilistic planning is just too complex. But the important principle that was demonstrated is the benefit of changing models—not specifically eliminating uncertainty. In fact,

in other contexts it might be equally beneficial to introduce uncertainty in order to create a compact model of an otherwise very large deterministic problem.

Treating problem reformulation as a formal reasoning process was started long ago (Amarel 1968). More recently, there have been some successful examples of treating it efficiently as a metareasoning process, particularly when the space of models being searched is restricted. For example, it has been shown that intelligent reformulation or restructuring of a belief network can greatly increase the efficiency of inference. A metareasoning process can be used to optimize the trade-off between the time dedicated to reformulating the network and the time applied to the implementation of a solution (Breese and Horvitz 1990).

Approximate modeling is therefore an important component of bounded rationality, but the ability to formalize and automate this process is still quite limited. The forms of metareasoning that can be used in approximate modeling are very rich, but they are not yet efficiently encodable in algorithmic forms.

Optimal Metareasoning

If one adopts the view that metareasoning is a process that monitors and controls the object-level reasoning process—as shown in Cox and Raja's figure 1.2—one could pose the question of whether the metareasoning process itself is optimal. Optimality here is with respect to the overall agent performance, given its fixed object-level deliberation capabilities. This is a well-defined question that sometimes has a simple answer. For example, metareasoning may focus on the single question of when to stop deliberation and take action. Depending on how the base-level component is structured, the answer may or may not be straightforward. Optimal metareasoning has been also referred to as *rational metareasoning* (Horvitz 1989) and *metalevel rationality* (Russell 1995) to distinguish it from perfect rationality. This offers one precise form of bounded rationality that is relatively easy to achieve. We will further examine this approach in the following sections. Besides being well defined and, in some cases, easily implementable, this approach to bounded rationality has some methodological benefits. It helps decompose the overall problem of bounded rationality into two orthogonal questions: how to design good problem-solving components and how to manage the operation of these components. Improving object-level competence can be a long-term objective, but at any given time it makes sense for agents to try to use their existing capabilities optimally.

It should be noted, however, that optimal metareasoning can result in arbitrarily poor agent performance. This is true because we do not impose up front any constraints on the object-level deliberation process in terms of its efficiency or correctness. Nevertheless, we will see that this presents an attractive framework for bounded rationality and that performance guarantees can be established once additional constraints are imposed on the overall architecture.

Bounded Optimality

Bounded optimality techniques seek to restore a stronger notion of optimality in decision making in the face of computational complexity and limited resources. That is, instead of building systems that can find "sufficiently good" answers, the goal is to find a maximally successful program that can compute these answers. Optimality is defined with respect to a particular space of possible implementations of these programs (Russell 1995; Russell and Wefald 1991).

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." This approach marks a shift from optimization 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. When the space of programs is finite, one can certainly argue that a bounded optimal solution exists. Finding it, however, could be very hard.

Russell, Subramanian, and Parr (1993) give an efficient construction algorithm that generates a bounded optimal program for a particular 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. Thus they propose a weaker notion of asymptotic bounded optimality as a more practical alternative. The latter case requires that the program perform as well as the best possible program on any problem instance, provided that its computational device is faster by a constant factor.

To establish bounded optimality, the designer of the system—not the agent itself—is responsible to identify the agent's reasoning architecture and to prove that the program satisfies the optimality conditions. In that sense, metareasoning does not play any significant role in this framework. Certainly there is no requirement that the agent itself be engaged in any form of

metareasoning. As long as the agent's program is shown to satisfy the optimality conditions, the agent is deemed bounded optimal.

One criticism of bounded optimality is that although the bounded rationality criterion is well defined, it is very difficult to achieve in practice. In fact, there are very few examples in the literature of bounded optimal agents. Bounded optimality may well be the most precise formal approach to bounded rationality, but without further refinement, it is hard to use in practice.

Bounded Rationality as Optimal Metareasoning

We have considered four basic approaches to achieve bounded rationality: heuristic search, approximate modeling, optimal metareasoning, and bounded optimality. The latter two approaches represent specific, well-defined solutions, whereas the former two represent general principles that fall under the broad category of approximate reasoning. From a formal perspective, the first approach is underconstrained, essentially allowing any form of approximate reasoning to count as a solution. The second approach is yet to be fully formalized and effectively automated. The last approach appears to be overconstrained, being difficult to achieve in practice. This leaves us with optimal metareasoning as the most promising approach for further examination.

According to this approach, metareasoning is a process that manages the object-level reasoning process. We consider an agent to be rationally bounded when its metareasoning component is optimal. That is, given a particular object-level deliberation model, we look for the best possible way to control it so as to optimize the expected ground-level performance of the agent. The metareasoning task can take many different forms and can present decisions of various complexities. We identify below the key questions that affect the form and complexity of the metareasoning problem.

1. What object-level decision-making architecture is employed? Is it complete? Is it sound? What trade-offs does it offer between computational resources and quality of results?
2. How does the metareasoning component model the object-level reasoning process? What kind of prior knowledge is available about the efficiency and correctness of the object-level component?
3. What run-time information about the state of the object-level reasoning process is being monitored? What is known about the external environment?

4. What control decisions are being made by the metalevel reasoning process? How do these decisions affect the object-level component?
5. When and how does execution switch between the object level and the metalevel?
6. How much time is consumed by the metalevel reasoning process? How much of the metareasoning strategy is precomputed off-line? What is the online overhead?
7. Is metareasoning optimal? What assumptions are needed to establish optimality?
8. What can be said about the overall performance of the agent? Can a bound be established on how close it is to an ideal perfectly rational agent?

Since the 1980s, several decision-making frameworks have been developed that match this form of bounded rationality. In the next section, we describe two such frameworks and examine the answers to the above questions in these particular contexts. We then mention briefly a number of additional examples of this general paradigm.

Example: Optimal Metareasoning with Anytime Algorithms

One general approach to bounded rationality is 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 metalevel control of computation.

Elementary algorithm construction covers the problem of introducing useful trade-offs between computational resources and output quality in decision making. This fundamental problem has been studied by the AI community, resulting in 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). Although 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. There are now many existing anytime algorithms for standard heuristic search and planning and reasoning tasks.

Performance measurement and prediction covers the problem of capturing the trade-off 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. It has been shown 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 systems with anytime algorithms as their components. The fundamental issue is that composition destroys interruptibility—the basic property that defines anytime algorithms. A two-step solution to this problem has been developed that makes a distinction between "interruptible" and "contract" algorithms (Zilberstein 1993). Contract algorithms offer a trade-off 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, metalevel control covers the problem of run-time allocation of computational resources (sometimes referred to as "deliberation scheduling" Dean and Boddy 1988) so as to maximize the overall performance of the system. In general, metalevel control involves modeling both the internal problem-solving process and the external environment and managing 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 domains characterized by rapid change and a high level of uncertainty, monitoring must be based on the use of interruptible algorithms. An early approach to monitoring anytime algorithms has been based on estimating the marginal "value of computation" (Russell and Wefald 1991). A more recent monitoring approach 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). We use the basic form of this approach as an example and discuss the details below.

1. What object-level decision-making architecture is employed?

The basic assumption about the object level is that it is an anytime algorithm, normally an interruptible one. Some anytime algorithms, such as anytime A* (Hansen, Zilberstein, and Danilchenko 1997), guarantee convergence on the optimal solution, but this is not generally required. If the anytime algorithm produces a result of quality q at time t, the value of that result is described by a time-dependent utility function, $U(q,t)$.

2. How does the metareasoning component model the object-level reasoning process?

Some form of a performance profile is normally used as prior knowledge. It characterizes the trade-off between run time and quality of results. Both deterministic and probabilistic models have been developed. A deterministic performance profile specifies a fixed solution quality per time allocation. A probabilistic performance profile, $\Pr(q_j|t)$, specifies the probability of getting solution quality q_j by running the algorithm for t time units. A more informative modeling tool is the dynamic performance profile, $\Pr(q_j|q_i,\Delta t)$, which specifies the probability of getting a solution of quality q_j by continuing the algorithm for time interval Δt when the currently available solution has quality q_i. This latter model revises the prediction of future quality based on the progress in problem solving made so far.

3. What run-time information about the state of the object-level reasoning process is being monitored?

One approach is to assume that the anytime algorithm used as the object-level reasoning process is a "black box" that does not provide any run-time indications of solution quality. When the quality of the current solution is available, that information together with running time can be used by the metareasoning process. In some cases, the quality of the current solution can only be estimated using certain features of the solution. In that case, the metareasoning component must estimate the quality of the solution using the available features (Hansen and Zilberstein 2001). In our example, we assume that solution quality is observable and that a dynamic performance profile is available.

4. What control decisions are being made by the metalevel reasoning process?

The most basic metareasoning decision is when to stop the anytime algorithm and return the current solution.

One approach is based on the myopic estimate of the expected value of continuing the computation for a period Δt, which is defined as follows:

$$\mathrm{MEVC}(\Delta t) = \sum_j \Pr(q_j \mid q_i, \Delta t) U(q_j, t + \Delta t) - U(q_i, t)$$

where q_i is the current quality and t the current time. The myopic monitoring approach is to continue the computation as long as MEVC(Δt) > 0.

A more general stopping policy can be found by optimizing the following value function (Hansen and Zilberstein 2001):

$$V(q_i,t) = \max_d \begin{cases} U(q_i,t) & \text{if } d = \text{stop} \\ \sum_j \Pr(q_j \mid q_i,\Delta t) V(q_j, t+\Delta t) & \text{if } d = \text{continue} \end{cases}$$

to determine the following policy,

$$\pi(q_i,t) = \arg\max_d \begin{cases} U(q_i,t) & \text{if } d = \text{stop} \\ \sum_j \Pr(q_j \mid q_i,\Delta t) V(q_j, t+\Delta t) & \text{if } d = \text{continue} \end{cases}$$

where Δt represents a single time step and d is a binary variable that represents the decision to either stop or continue the algorithm.

When each activation of the metareasoning component takes a non-negligible amount of computation time that slows down the object-level reasoning process, the decision could also include the frequency of monitoring (Hansen and Zilberstein 2001). An even more complex situation could arise when the metareasoning component uses a variety of features that characterize the state of the environment and the state of the computation. Some of these features could be more costly to compute than others. In that case, the metareasoning decision is what to monitor, at what frequency, and when to stop the entire process.

5. When and how does execution switch between the object level and the metalevel?

In most cases, monitoring of anytime algorithms is done periodically at fixed intervals, say, every Δt time units. When we assume that monitoring itself incurs negligible overhead, the frequency of monitoring can be high with no negative consequences.

6. How much time is consumed by the metalevel reasoning process?

Work on anytime algorithms often relies on precomputed control strategies that are generated off-line using the performance profile of the algorithm and the

overall time-dependent utility function. For example, the above value function and associated control policy can be computed off-line, resulting in a very fast reactive metareasoning component. Simple control strategies, such as the above myopic approach, that stop the computation when the marginal value of computation becomes negative can be computed online with little overhead. When solution quality or the state of the environment must be estimated at run time using nontrivial computations, this could introduce a significant overhead. However, extending the above monitoring technique to factor in this overhead is relatively straightforward (Hansen and Zilberstein 2001).

7. Is metareasoning optimal?

Optimal metareasoning has been introduced for a wide range of scenarios involving anytime algorithms using certain assumptions about the performance profile and the utility function. One common assumption is that metareasoning incurs negligible overhead. The myopic stopping criteria can be shown to be optimal when the expected marginal increase in the intrinsic value of a solution is a nonincreasing function of quality and the marginal cost of time is a nondecreasing function of time. The monitoring policy computed by optimizing the above value function is optimal and does not require the latter assumptions. A range of situations in which optimal metareasoning can be established is described by Hansen and Zilberstein (2001) and Zilberstein (1996).

8. What can be said about the overall performance of the agent?

Even when metareasoning is optimal—and satisfies our definition of bounded optimality—not much can be said about the overall performance of the agents and how close it may be to a perfectly rational agent in the same situation. Generally, no performance bound exists because the anytime algorithm being monitored is not subject to any constraints in terms of its efficiency or correctness. But when the quality measure of the anytime algorithm provides an error bound on how close the result is to the optimal answer, a worst-case bound with respect to a perfectly rational agent can be established.

To summarize, there are many instances of optimal metareasoning involving anytime algorithms as an object-level deliberation method. There are also examples of optimal metareasoning with respect to other object-level components such as algorithm portfolios (Petrik and Zilberstein 2006) and contract algorithms (Zilberstein, Charpillet, and Chassaing 2003). These examples illustrate that this well-defined model of bounded rationality can be implemented in many domains.

Discussion and Conclusion

We examined several different computational approaches to bounded rationality. One approach—based on optimal metareasoning—seems particularly attractive because it is both relatively easy to implement and provides some formal guarantees about the behavior of the agent. We examined several instantiations of this approach using anytime algorithms and provided a characterization of the relationship between the object-level and metareasoning components. These examples show that metareasoning plays an important role in this particular approach to bounded rationality.

Although bounded optimality seems to be a very different approach, under some assumptions it can be unified with optimal metareasoning. If the architecture of the bounded-optimal agent specifies the object-level computations of Cox and Raja's figure 1.2, then the metareasoning problem can be seen as finding the best way to compose the object-level computations and create the most effective agent. In that case, the bounded-optimal program is a solution of the optimal metareasoning problem. One example is the problem of sequencing contract algorithms in order to produce the best possible interruptible anytime algorithm. Contract algorithms offer a trade-off between computation time and solution quality, but the run time must be determined when they are activated. Once activated, no solution is available before the end of the contract. Some reasoning and search methods produce useful contract algorithms. To use such algorithms when the available time is not known in advance, one could run them as a sequence of increasing contracts until the deadline. It has been shown that the best possible way to create such a sequence (in terms of the resulting performance profile) is to use a geometric series of contracts, doubling execution time in each step (Zilberstein and Russell 1996; Zilberstein, Charpillet, and Chassaing 2003). If the task of the metareasoning component is to determine the run time of each contract, then the optimal sequence provides a solution to the optimal metareasoning problem. At the same time, if we consider an architecture in which programs are composed of sequences of contracts, then the optimal sequence is also a solution of the bounded optimality problem. Hence, under certain assumptions the two approaches can be unified, yielding the same solution.

One interesting research challenge is to establish mechanisms to bound the performance difference between the more practical approach based on optimal metareasoning with a given object-level component and a bounded-optimal agent,

using the same architecture. Creating a bounded-optimal agent is hard, but bounding the performance gap might be possible.

Another challenge is to develop models of bounded rationality suitable for multiple decision makers in either cooperative or competitive settings. When agents operate independently and cannot be controlled in a centralized manner, their metareasoning components need to coordinate as well. A simple example is a collaborative setting in which one agent decides to stop thinking and take action, but the other may see a need to continue deliberation. There has been little work so far on coordination between the metareasoning components of collaborative agents. The situation is even more complicated in competitive settings when agents need to monitor or reason about the deliberation processes of other agents, about which they may have little information or prior knowledge.

Acknowledgments

This work was supported in part by the National Science Foundation under Grants no. IIS-0535061 and IIS-0812149 and by the Air Force Office of Scientific Research under Grant no. FA9550-08-1-0181.

References

Amarel, S. (1968). On representations of problems of reasoning about actions. Machine Intelligence, 3, 131–171.

Breese, J., and Horvitz, E. (1990). Ideal reformulation of belief networks. In Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence (pp. 64–72). Boston: Morgan Kaufmann.

Dean, T., and Boddy, M. (1988). An analysis of time-dependent planning. In Proceedings of the Seventh National Conference on Artificial Intelligence (pp. 49–54). Cambridge, MA: MIT Press.

Doyle, J. (1990). Rationality and its roles in reasoning. In Proceedings of the Eighth National Conference on Artificial Intelligence (pp. 1093–1100). Cambridge, MA: MIT Press.

Gigerenzer, G. (2000). Adaptive Thinking: Rationality in the Real World. Oxford: Oxford University Press.

Gigerenzer, G., Todd, P. M., and ABC Research Group. (1999). Simple Heuristics That Make Us Smart. Oxford: Oxford University Press.

Good, I. J. (1971). Twenty-seven principles of rationality. In V. P. Godambe and D. A. Sprott (eds.), Foundations of Statistical Inference (pp. 108–141). Toronto: Holt, Rinehart, Winston.

Hansen, E. A., and Zilberstein, S. (1996). Monitoring the progress of anytime problem solving. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (pp. 1229–1234). Menlo Park, CA: AAAI Press.

Hansen, E. A., and Zilberstein, S. (2001). Monitoring and control of anytime algorithms: A dynamic programming approach. Artificial Intelligence, 126(1–2), 139–157.

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

Horvitz, E. J. (1987). Reasoning about beliefs and actions under computational resource constraints. In Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence[j1] .

Horvitz, E. J. (1989). Rational metareasoning and compilation for optimizing decisions under bounded resources. In Proceedings of the International Symposium on Computational Intelligence (pp. 205–216[j2]).

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

Petrik, M., and Zilberstein, S. (2006). Learning parallel portfolios of algorithms. Annals of Mathematics and Artificial Intelligence, 48(1–2), 85–106.

Russell, S. J. (1995). Rationality and intelligence. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (pp. 950–957). Menlo Park, CA: International Joint Conferences on Artificial Intelligence

Russell, S. J., Subramanian, D., and Parr, R. (1993). Provably bounded optimal agents. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (pp. 338–344). Menlo Park, CA: International Joint Conferences on Artificial Intelligence

Russell, S. J., and Wefald, E. H. (1991). Do the Right thing: Studies in Limited Rationality. Cambridge, MA: MIT Press.

Simon, H. A. (1947). Administrative Behavior. New York: Macmillan.

Simon, H. A. (1982). Models of Bounded Rationality (vol. 2). Cambridge, MA: MIT Press.

Wellman, M. P. (1990). Formulation of Tradeoffs in Planning under Uncertainty. London: Pitman.

Zilberstein, S. (1993). Operational rationality through compilation of anytime algorithms. Ph.D. dissertation, Computer Science Division, University of California, Berkeley.

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

Zilberstein, S., Charpillet, F., and Chassaing, P. (2003). Optimal sequencing of contract algorithms. Annals of Mathematics and Artificial Intelligence, 39(1–2), 1–18.

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