[4608]

Reprinted from THE BELL JOURNAL OF ECONOMICS
Vol. 9, No. 2, Autumn 1978

On how to decide what to do

Herbert A. Simon

Reprinted from THE BELL JOURNAL OF ECONOMICS
Vol. 9, No. 2, Autumn 1978

On how to decide what to do

Herbert A. Simon

On how to decide what to do

Herbert A. Simon
Professor of Psychology
Carnegie-Mellon University

Economics, which has traditionally been concerned with what decisions are made rather than with how they are made, has more and more reason to interest itself in the procedural aspects of decision, especially to deal with uncertainty, and more generally, with nonequilibrium phenomena. A number of approaches to procedural rationality have been developed in such fields as operations research and management science, artificial intelligence, computational complexity, and cognitive simulation which might be of considerable value to economics as it moves in this new direction.

1. Introduction

■ It is commonplace to observe that economics has traditionally been concerned with *what* decisions are made rather than with *how* they are made—with substantive rationality rather than procedural rationality. In other places (Simon, 1976, 1978), I have tried to explain why this is so, and to suggest some reasons why it is changing—why economics is becoming increasingly interested in how people go about deciding what to do, or, in a normative mode, how they *should* go about it. Economics is not the only domain of science that is concerned with questions of procedural rationality; indeed, such questions have been at the very center of attention of several other disciplines, and important progress has been made in finding answers.

Let me start with the most familiar realms, and then go on to those that are more remote. First, I shall take up operations research and management science, then artificial intelligence and the new discipline of computational complexity, and finally, cognitive simulation. I shall also discuss briefly how the new developments might apply to traditional problems in economics. For the most part, however, I shall leave those applications to the reader and to the future, after stating my reasons for thinking that they will be needed for the continuing progress of economic analysis.

2. Operations research and management science

■ What is the goal of basic theory in operations research (OR) and management science? It is to specify good (or best) methods for finding good (or best)

This research was supported by Research Grant MH-07722 from the National Institute of Mental Health.

decisions in complex managerial situations. OR theory is a theory of computation, of procedural rationality. It is, to paraphrase Clausewitz (1832), the continuation of classical numerical analysis by other means. The "other means" are such tools as linear, dynamic, integer, and geometric programming, queuing theory, combinatorial analysis, simulation, and search theory. OR theory is normative theory: it tells managers how they ought to use their computers. Of course, as more and more managers follow its advice, it also becomes a positive science, describing some of the actual decision procedures employed by organizations.

I observed in passing that OR theory is normative in a dual sense: it is concerned with good methods for reaching good decisions. Thus, a linear programming (LP) algorithm is a procedure for reaching an optimum in a situation represented as a system of linear equations and inequalities. A good LP algorithm is one that determines the optimum at a relatively low computational cost. Research in LP is directed at inventing new algorithms or new computational paradigms that will reduce this cost.

Conceptually, of course, there is no reason why we need to treat the substantive decision problem and the procedural problem at arm's length, as though they were independent of each other. The global optimization problem is to find the least-cost or best-return decision, *net* of computational costs. We formulate this problem in terms of a tradeoff between the marginal computational cost and the marginal improvement in the substantive decision it is expected to produce (Simon, 1955; Stigler, 1961; Marschak and Radner, 1972).

In practice, this is not always the way we proceed. In writing LP algorithms, we almost always assume that we want to discover the optimal solution, while disregarding computational costs. We then try to devise an algorithm that will find this solution as cheaply as possible. This is a reasonable way even for an optimizer to proceed, since computers are rather cheap these days, and getting cheaper. For $300 one can purchase an hour's attention from a fairly large computer, and if one's interest is in scheduling an oil refinery, the payoff from making correct decisions is several orders of magnitude larger than that.

Why, then, be concerned with the $300 at all? Well, as managers say, it is still money. If one algorithm can compute the solution in an hour, while another takes two, it is worth pocketing the change. But there is a more important reason: there are not only cost constraints on the solutions of problems; there are also real-time constraints. If the refinery has to be rescheduled daily, a scheduling algorithm that takes two weeks to run provides little help or comfort. With each modest increase in the size of the problem the cost of finding a solution with any given algorithm is likely to increase by orders of magnitude. For problems above some limiting size, the algorithm becomes not just expensive but impracticable—it simply will not yield solutions in any reasonable span of time. For this reason, OR researchers concerned with linear programming (or other) algorithms have usually held that their goal is to expand the upper limits of problem size for which solutions can be found, rather than to reduce the cost of finding solutions for problems of a given size. Of course, these are simply the two sides of the same coin, and, in fact, alternative algorithms are frequently evaluated by comparing their speeds of solution on standard sets of test problems.

The evaluation of algorithms is still largely a pragmatic matter, with little theoretical foundation. That is to say, there are today few theorems that prove that a particular LP algorithm is the most efficient, or that one class of algorithms

is better than another. Algorithms are usually evaluated, as indicated above, by running them on test problems and recording the amount of computation required for solution. This state of affairs has been a source of understandable unhappiness in OR, and is showing some signs of impending change. Within computer science, a new subdiscipline—computational complexity—has emerged. It addresses problems of precisely this kind. Operations researchers (and a few economists) have begun to make contact with it, and I shall have more to say about it later.

When we shift our attention from LP to the domain of integer problems, or combinatorial problems in general, the criterion of procedural rationality takes a different shape. In these domains, it is easy to frame problems of practical import whose exact solutions are well beyond present and prospective computational resources, even with the most efficient algorithms we have available. Here we must turn again to heuristic procedures that give us reasonable approximate solutions with reasonable amounts of computation, and hence we are faced with the tradeoff of solution quality against computational cost. The classic example of an immense problem is choosing a move in the game of chess. Since chess is a finite, zero-sum game of perfect information, it is trivial to prove that a (not necessarily unique) best strategy exists; but it is also easy to estimate that finding it by systematic, unselective, search would require exploring a tree of some 10^{120} branches. Algorithms for solving problems by selective, heuristic search of large spaces have been studied both in OR and in the discipline of artificial intelligence, and I shall discuss them under the latter heading.

In summary, OR theory is a part of the normative theory of procedural rationality specifying both algorithms for finding optimal or good decisions and procedures, usually empirical and pragmatic, for evaluating such algorithms. Conceptually, these theories are concerned with the tradeoff between the quality of the solution and the cost of finding it, but in most cases, the tradeoff is only implicit. On the one hand, when it is possible to find optimal-solution algorithms that demand only reasonable amounts of computing time, the quality of the solution drops out of the equation, and we are concerned only with the cost of computation. At the other extreme, when the problem spaces to be searched are very large, and we are unable to discover structure in them that would permit the search to be conducted in an efficient way, then costs hardly enter in at all, for we are constrained to seek *any* algorithm that will find acceptable solutions with acceptable amounts of computing effort. In these realms, the exponential explosion of computation with increasing depth of search is a familiar and melancholy fact of life, and "efficiency" means the difference between getting there at all or remaining lost in the maze of possible paths. And that brings us to the topic of artificial intelligence.

3. Artificial intelligence

■ Artificial intelligence is the discipline that is concerned with programming computers to do clever, humanoid things—but not necessarily to do them in a humanoid way. The closely allied field of cognitive simulation (or "cognitive science" as it is more and more being called) is concerned with programming computers to do the clever things that people do, but to do them by using the same information processes that people use. The close relation between artificial

intelligence and cognitive science is, in a sense, accidental. Experience of the past twenty years has shown that often (not always) the best way to program a computer to solve complex problems, discover concepts or patterns, or understand natural language is to imitate as closely as possible the way humans handle the same tasks.

Humans and computers do not, however, have the same strengths. People have very small short-term memories but indefinitely large long-term memories (see Section 6); no such distinction need be made in computer memory. People are very poor at doing large amounts of simple arithmetic, but very good at carrying out highly selective searches, using complex criteria of selection; almost the converse is true of computers—they are bears for arithmetic, but short on subtlety. Of course, when I say "computers" here, I mean computers programmed in simple, straightforward ways. The whole aim of cognitive science is to induce cleverness and subtlety into the behavior of computers, not by inventing new hardware, but by writing programs that incorporate the heuristic devices that people use.

The history of chess playing programs for computers illustrates nicely the uneasy alliance between artificial intelligence methods and the methods of cognitive simulation. The earliest chess playing programs largely relied on computer speed (such as it was in the middle 1950s) to explore, to a modest depth, a large number of possibilities in the game tree.[1] In 1958, Newell, Shaw, and I demonstrated a much more humanoid chess program that searched much less (a few hundred or thousand branches), but thought much more. It did not play very good chess (neither do most people), but it was in the same ballpark as the speed-oriented programs contemporary with it.

Today, we have some very good chess playing programs, comparable to human experts in playing strength (hence ranking with the top few hundred players in the United States). These programs resemble neither kinds of the earlier systems. They do a great deal of search, using the power of the computer, and of course they search many times faster, hence more extensively, than the programs of the 1950s. On the other hand, they incorporate a large amount of chess knowledge, so that their searches are in fact conducted quite selectively, and hence to considerable depth in the game tree. So the progress in computer chess has depended upon a blending of the computer's brute force with humanoid intelligence, and the field is likely to retain this hybrid form for some time to come. There is no sign that, in the near future, a strong computer chess program will be able to dispense with either machine speed or heuristic selectivity.

However closely history may have entwined their fates, it should be clear that artificial intelligence and cognitive science have quite distinct goals. Artificial intelligence is a normative discipline. Like OR, its goal is to find powerful problem-solving algorithms, and no holds are barred. In fact, there is no real boundary between these two disciplines, and today the theory of heuristic search is being pursued vigorously by both. Cognitive science, on the other hand, is a positive discipline, a branch of psychology, whose goal is to discover how humans perform complex cognitive tasks. Artificial intelligence (together with OR and parts of statistical decision theory) is a normative science of procedural

---

[1] For a history of these efforts up to 1958, see Newell, Shaw, and Simon (1958).

rationality; cognitive science is a positive science of procedural rationality. Whether the two fields will continue to be associated as closely as they have been in the past will depend on the relative rates of progress of computer speed and power, on the one hand, and our understanding of human heuristics, on the other. For once, I shall be cautious and not offer a prediction. (Al Newell and I are just now celebrating the twentieth anniversary of our 1957 ORSA predictions of the future of artificial intelligence (Simon and Newell, 1958).)

4. The state of the art in artificial intelligence

■ There are a number of areas other than chess where artificial intelligence programs have reached respectable human levels of performance. Some examples go back nearly twenty years—I refer to Tonge's assembly-line balancing program, Clarkson's simulation of an investment trust officer, and some of the heuristic programs for finding good solutions to scheduling problems.[2] Other examples are more recent: the DENDRAL program (Buchanan and Lederberg, 1971), which identifies molecules by analysis of mass spectrogram data; MYCIN (Davis, Buchanan, and Shortliffe, 1977) and INTERNIST (Pople, 1977), which make medical diagnoses in particular areas of disease; the programs for automatic chemical synthesis developed by Corey (Corey and Wipke, 1969), Powers (1972) and Gelernter and his associates (Gelernter *et al.*, 1977). All of these programs share with the chess programs a hybrid reliance on machine power and human selectivity, but with a considerably stronger admixture of the latter than is characteristic of the best existing chess programs.

When I speak of these programs as reaching "respectable human levels of performance," I mean levels like those we expect of professionals in the field. (The progress of chess programs was long obscured by unwisely setting world championship as the goal.) Evaluating such programs is beset with the same difficulties as evaluating OR algorithms. There are few opportunities for proving optimality, or even theoretical dominance of one program over another. For the most part, evaluation is carried out by examining performance in comparison with the performance of other programs and human professionals on standard tasks. In chess, standardized evaluation has been facilitated by the generosity of human chess players in admitting chess playing programs into their tournaments, and by the organization of computer chess tournaments in recent years. In areas like medical diagnosis, no such standardized evaluation procedures are available, and *ad hoc* methods have had to be devised.

A few islands of theory have begun to appear, however, that enable more general statements to be made about the power of algorithms. The first of these relates to the alpha-beta heuristic, a scheme for pruning search trees in two-person games, and thereby reducing the total search effort. The alpha-beta heuristic belongs to the general class of branch-and-bound methods, which are widely used in integer programming.

The central idea of branch-and-bound is this. Suppose that I can place an accurate upper bound and an accurate lower bound on the payoff obtainable from any of the branches of a search tree that are descendants of a particular

---

[2] A number of the pioneering heuristic search programs are described in Feigenbaum and Feldman (1963).

branch, say A. Suppose that the upper bound on a different branch, B, is lower than the lower bound on A. Then it is obviously futile to search any of the descendants of B, and the tree can be pruned accordingly. The alpha-beta scheme is essentially the same, complicated only by the fact that the optimizing criterion is the minimax criterion, to take account of the opposed goals of the two opponents and their alternating choice of move.

Theoretical analysis of the alpha-beta heuristic shows that, in a tree with B branches at each node, it may be expected to reduce the search to a subtree with a branching factor of about $B^{1/2}$. Results of this kind can be proved either by making simplifying assumptions about the tree (e.g., that payoffs are distributed randomly on the terminal branches), or by carrying out worse-case analysis—that is, computing the amount of search that would be required if the payoffs were arranged in the tree in the most perverse order possible.[3] Results of these kinds belong to a new branch of computer science called computational complexity. I shall have more to say about it in the next section.

A number of theories about optimal algorithms have been constructed in addition to branch-and-bound theory. There are two separate pieces of theory, each applicable to a particular class of problems. In one class of problems of heuristic search, we are interested in finding *shortest paths* to the goal. An example is the traveling salesperson problem: we are seeking the shortest path that will take a person from a starting city, through all of a set of cities, and back to the starting point. Finding *a* path that encompasses all the cities is trivial; finding the shortest path is a difficult combinatorial problem, which we can try to solve with an algorithm for *best-first* search. Suppose that, for any partial path, we have some way of estimating accurately the length of the shortest path for accomplishing the complete goal. Then the optimal path is one for which, at all points along it, the sum of the distance already traveled and the minimum distance yet to be traveled is equal to the shortest total path. With an accurate distance measure, the optimal path can be detected without any extraneous search whatsoever.

In general, of course, we possess no such infallible distance measure, but only some more or less approximate estimate of the distance that remains to be traversed. We can still use this estimate to conduct a best-first search, as follows. First calculate the measure for all cities reached directly from the starting city. Then pick the lowest of these and continue the set of possible paths one city further, evaluating again. Repeat this process, always continuing from the city whose total path estimate is "best so far" until a path is found along which all the cities have been visited, and the goal city reached again.

Now we call a distance estimator *admissible* if it is guaranteed, when used to evaluate best-first search, to find the shortest path. Any distance estimator that never overestimates the distance is clearly admissible, while distance estimators that sometimes overestimate may not be. An admissible estimator need not, of course, be efficient. If the estimator grossly overestimates distances on the true path relative to other paths, until it comes close to the very end, it will make many unnecessary explorations before it finds the solution.[4]

---

[3] These statements are very approximate. A general discussion of the alpha-beta heuristic will be found in Nilsson (1971), pp. 140–149, and analyses of the efficiency of the heuristic in Knuth and Moore (1975) and Newborn (1977).

[4] Further discussion of these matters will be found in Nilsson (1971), pp. 54–71, and in Gaschnig (1977).

In a great many practical problem situations, finding the shortest path that leads to a solution is quite a different matter from finding a path with a minimal expenditure of computing effort. Of course, in the case of the traveling salesperson problem we want the shortest path—that is part of the definition of the solution. On the other hand, when we are searching for the proof of a mathematical theorem, and if we want to find the proof in a reasonable time, we shall not insist upon finding the shortest proof. In most cases where the length of the solution path is irrelevant, the cost of search can be reduced enormously by ignoring path length as a criterion.

A different approach is needed to evaluate search procedures when the task is to find any solution from the one required when the task is to find the shortest-path solution. Suppose we are searching for oil. Oil can only be discovered in certain locations if we have previously explored other locations—that is, we can only drill to the 2,000-foot stratum by first drilling through the 1,000-foot stratum. If there may be oil in either stratum, both possibilities must be taken into account in determining the profitability of drilling to either layer. Such precedence relations complicate greatly the task of devising a good evaluation function for a best-first search in problem domains of this kind. Roughly speaking, we should do our next bit of drilling at the point where the expected return is greatest, taking into account the various outputs that might be obtained from a well penetrating through that point, and the depths at which they may be found. Characterizations of optimal algorithms for a large class of such problems have been obtained by Garey (1973), Simon and Kadane (1975), and Sidney (1975).

5. Computational complexity

■ I have several times referred to computational complexity as a new theoretical approach to the study of problem solving. Some of the results I have just described in the discussion of artificial intelligence—those relating to the optimality of certain search algorithms—could be regarded as part of the theory of computational complexity. However, they lie a little aside from the central focus of that theory.

Classical numerical analysis has long been concerned with goodness of approximation and speed of convergence of computational algorithms. In the case of algorithms that are, sooner or later, guaranteed to find exact solutions, the corresponding concern is with the amount of computing required to find them.

Early in its history, computer science became concerned with the decidability problem: with the question of whether one could be sure that a problem solving program operating in some problem domain would always reach a positive or negative answer to a question from that domain in a finite time. The celebrated theorem of Gödel (1931) showed that no such guarantee could be provided for any domain that was sufficiently rich (which meant a good many domains, many of them deceptively simple in appearance).

It gradually dawned on computer scientists, however, that the decidability question was not usually the right question to ask about an algorithm or a problem domain. (So great was the fascination of automata theory and the prestige of the Gödel theorem that the dawning took several decades.) It really did not matter very much whether the answer to a problem would never be forthcoming,

or whether it would be produced only after a hundred years. The important questions for computing were the probabilities that answers would be produced in a reasonable computation time (an hour or a day, depending on the importance of the problem), and what the average computing time would be for those problems from a domain that could be solved at all with reasonable amounts of computation. These are the questions addressed by the theory of computational complexity (Aho, Hopcroft, and Ullman, 1974).

Questions of computational complexity can be approached with either mathematical or computational tools. We can try to prove theorems about the complexity of a certain class of problems, or we can actually compute solution times for samples of problems drawn from the class. Most mathematical results that have been obtained in the theory provide information only about the worst case: they give upper bounds on solution times for all problems in a class. In general, these upper bounds are expressed as functions of problem size. Thus, the statement that a certain class of problems is "exponential" means that as we increase the number of problem elements or components, the maximum time required for solution will rise exponentially. Here "time" really means the number of elementary computational steps that must be executed to solve the problem.

Notice that these are not theorems about the efficiency of particular computational algorithms. They are limits that apply to *any* algorithms used to solve problems of the domain in question. The theorems warn us that we must not aspire to construct an algorithm that will improve upon the worst-case limit. Nor are the theorems usually constructive: they do not tell us how to build an algorithm that will actually reach the limit—the latter is a lower bound.

When it has been proved that a particular class of problems is exponential, we know that we shall sometimes fail to solve problems in this class unless the problems are quite small. If the class of problems is only polynomial—and especially if it is only linear—then we can aspire to solve problems up to very much larger size limits. Several very important classes of problems, including a number of standard OR problem classes, have been shown to be of equivalent complexity—NP-complete. Since it is not yet known whether NP-complete problems are polynomial or exponentially complex (or something in between), we may still be optimistic about finding powerful algorithms to solve them. However, there is a suspicion abroad among specialists in computational complexity that these problems are actually exponentially complex.

Before we become too despondent about these results and prospective results, we should recall that they apply only to the worst case. We might be willing to put up with failing, occasionally, to find the solution to a problem in a reasonable length of time if we could be assured that we *usually* would succeed. Unfortunately, theorems about expected computing times for problems belonging to some class, and, even more relevant, theorems about the fraction of problems that could be solved within some specified time, are hard to come by. Here, we generally have to be satisfied with empirical tests on randomly selected samples of problems from the domain.

In the last couple of years, Rabin has obtained some interesting potentially important results to the effect that we can reduce certain domains from exponential to polynomial complexity if we are willing to settle for approximate solutions. He has also shown that in some theorem-proving domains, complexity is greatly reduced if we permit a small fraction (it need only be an

epsilon fraction) of erroneous proofs. Those of us who are faster than we are accurate can take comfort from that.

I have long had a favorite example to show how computational complexity can be greatly reduced if we are willing to accept approximations: it has to do with finding needles in haystacks. If needles are distributed randomly in a haystack of size, H, with an average density of distribution, d, then to find the sharpest needle in the stack, we have to search the entire stack, and the search time will vary with H. Search time will be linear with size, which does not seem too bad until we remember that the haystack of life is essentially infinite.

If we are satisfied with any needle, however (after all, they are all sharp enough to sew with), then the amount of search to find one will vary with d—that is, will be independent of the size of the stack. Complexity independent of the size of the problem domain is a property we badly need in algorithms designed to face the problems of the real world. The theory of computational complexity has not yet really addressed issues of this kind. Its main value at present to persons concerned with the decisionmaking process is to underscore the great complexity even of problems with relatively clean structures (e.g., integer programming problems), and to warn us against abstracting from the costs of computation and the needs for approximation in our theories of problem solving and decisionmaking.

6. Cognitive simulation

■ Thus far we have been discussing normative theories of procedural rationality that have been developed in operations research, artificial intelligence, and computational complexity. There has been a parallel development in the past two decades of a positive theory of procedural rationality in the discipline of cognitive psychology, based primarily on work that uses the computer to simulate human thought processes.

I have already suggested the basic reason that we might expect the positive theory to differ in some respects from the normative theory. Human beings do not have the arithmetic capabilities of computers. There is a striking difference in speed, a difference that increases by an order of magnitude each three to five years. There is also a very important difference in the capacities of short-term memory (STM). Information that is being processed by the human central nervous system has to be held in STM, a memory of notoriously small capacity. STM is the memory we use when we look up a number in the phone book, and retain it long enough to dial it. Most people find that a phone number can be held comfortably in STM, but adding the area code may fill it past overflowing.

Research in recent years has shown that human performance on cognitive tasks (especially, but not exclusively, when they are carried out without paper and pencil) is dramatically sensitive to the limits of STM. For example, in concept attainment tasks, when a generalization is to be derived from a sequence of instances, only a few of the most recent instances (sometimes only one or two) can be held in memory, with the result that hypotheses are often entertained that contradict evidence that was available only a few minutes earlier. The "event matching" behavior that is frequently observed in human attempts to predict time series is largely attributable to this excessive attention to events in the recent past and ignoring of earlier ones. In fact, large anomalies

are observed in the ways in which people combine earlier and later evidence in making judgments, so that their inferences are generally inconsistent with Bayesian models of information accumulation (Kahneman and Tversky, 1973). These anomalies are probably attributable in large measure to limits on memory.

The relative slowness of human thought processes (measurable in tens or hundreds of milliseconds, while the primitive processes of contemporary computers are at the microsecond or nanosecond levels) leads people to avoid problem solving methods that require searches of large spaces. The human chess grandmaster probably does not search more than a hundred (just possibly as many as a thousand) possibilities in the course of his consideration of a difficult position. His computer competitor (who is only an expert, at best) will examine several hundred thousand possibilities, or even several million, in the same situation. Clearly the use of selective heuristics, a characteristic of many artificial intelligence programs, is even more highly developed in human thinking.

The study of skilled chess players shows that a body of knowledge stored in long-term memory (LTM) compensates in large measure for the slowness of search. The human expert does not so much *search out* the correct move as *recognize* it (see Chase and Simon, 1973). I do not mean anything mysterious by this. There is now good empirical evidence that a skilled human chessplayer holds in memory perhaps 50,000 different patterns of pieces (i.e., patterns of clusters of three of four pieces) that he will recognize instantly when they are present in a chess position, and that will evoke for him information stored in LTM about possible moves that are relevant when those particular patterns are present. This is the reason why a grandmaster can play fifty or more opponents simultaneously, spending only a few seconds at each move. He simply waits for his weaker opponents to make a mistake (i.e., to create a recognizable feature on the board that denotes a weakness in the position), and then exploits it with the help of the knowledge he has stored in LTM.

This evidence suggests that, for humans, accumulated experience is indeed a very large component of high-level skill (no one, not even Bobby Fischer, reaches grandmaster status in less than about ten years of intense study and effort). This accumulation of experience may allow people to behave in ways that are very nearly optimal in situations to which their experience is pertinent, but will be of little help when genuinely novel situations are presented. That conclusion is consistent with our general belief that the limits of human rationality become particularly important in explaining behavior under uncertainty—where we translate "uncertainty" here to mean any kind of significant novelty.

These, then, are some of the considerations that must be incorporated in any positive theory of procedural rationality that purports to explain how human beings make decisions in complex task domains. The theory must take account of the fact that the human information processor operates serially, being capable of dealing with only one or a few things at a time. The basic processes are slow (by comparison with modern computers), so that the system is incapable of undertaking extensive searches of problem spaces, particularly when it must reach a decision in a matter of minutes or hours. Judging from the chess evidence, consideration of a hundred closely related alternatives could easily occupy a quarter hour's concentrated thinking. The flexibility of search (e.g., the use of best-first search techniques) is further limited by the very small size of human short-term memory. It is not possible to "save" in STM

recollections of significant numbers of unexplored branches in the search tree, so that when search along a particular line is unsuccessful, there is very limited capability for backup to more promising lines that were passed up earlier.

On the positive side, the human information processing system is capable of storing large amounts of information in long-term memory, and of retrieving them upon recognition of familiar patterns in stimuli. Direct retrieval of possible courses of action as a result of recognizing familiar features of the problem situation provides a major (one might almost say *the* major) basis for professional performance in complex problem situations. We would predict of a system having this characteristic a very much more sophisticated level of performance in familiar situations, where the recognition mechanism could operate effectively, than in situations possessing any considerable element of novelty.

We would not expect a system like this to perform in a history-free manner, but would expect, instead, to see many evidences of learning in its behavior over any considerable period of time. Thus, we would not necessarily expect a positive economic theory that fit nineteenth century data to fit twentieth century data. For example, there is considerable evidence that, as a result of the availability of computers, methods of determining desired inventory levels changed substantially in large American business firms in the 1950s and 1960s. These changes in decision methods showed up at the level of the economy as changes in the parameters of the inventory cycle. As Franco Modigliani was fond of saying, "If businessmen are not now maximizers, after enough of them have graduated from business school, they will be." So we might even expect that a positive theory of economic behavior will have to include as a subtheory the way in which business schools produce and diffuse decisionmaking techniques.

I do not want to exaggerate, however, the speed of such learning processes. Not only is social diffusion of new techniques a lengthy matter, but the rate at which the invention of new techniques is expanding human cognitive capabilities is modest. For a long time to come, man, even in association with his most powerful computing devices, will be navigating a very small ship in a vast sea of alternatives and their possible consequences.

7. Applications to economics

■ Many questions of economics cannot be answered simply by determining what would be the substantively rational action, but require an understanding of the procedures used to reach rational decisions. Procedural rationality takes on importance for economics in those situations where the "real world" out there cannot be equated with the world as perceived and calculated by the economic agent. Procedural rationality is the rationality of a person for whom computation is the scarce resource—whose ability to adapt successfully to the situations in which he finds himself is determined by the efficiency of his decisionmaking and problem solving processes.

The domains where a theory of computation, normative or descriptive, is likely to prove useful are the domains that are too complex, too full of uncertainty, or too rapidly changing to permit the objectively optimal actions to be discovered and implemented. Nor can we rely on evolutionary arguments to conclude that natural selection will find the optimum where calculation cannot. All we can conclude from natural selection is that the fitter will survive in competition with the less fit. There is no theorem that proves that the process will

converge, in historical time, to limit survival to the absolutely fittest—those who have found the objective optimum. It is much more likely, in a world with rapidly advancing human knowledge and technology, with an unpredictably shifting political situation, with recurrent and unforeseen (if not always unforeseeable) impacts of demographic, environmental, and other changes, that the location of the objective optimum has little relevance for the decisionmakers or the situations that their decisions create.[5]

Let us consider, more concretely, some of the specific domains in economics to which a theory of calculation and procedural rationality has something to offer. Normative microeconomics is one, for that is the area where operations research and management science have already made large contributions. American business firms make their inventory decisions, their cash-holding decisions, and their investment decisions in a significantly differently way from that which they did thirty years ago, as the result of the availability of the new algorithms and the new computers. In some cases, the new methods are sufficiently powerful to permit them to achieve at least suboptimization in limited problem domains. In other cases, the new methods simply provide them with more powerful heuristics than they had before for reaching "good enough" decisions. They allow a higher level of procedural rationality to be reached.

What we have learned of the procedural complexities of normative microeconomics underscores the need for introducing procedural considerations—as has already been done to a modest extent—into positive microeconomics. Entering wedges have already been made in labor theory, with models of search processes in the labor market. Similarly, the search and decision processes underlying consumer brand and product decisions are now receiving some attention.

The theory of the business cycle is another important candidate area, for a procedural theory of the forming of expectations and the making of decisions. Our capacity to predict the ups and downs of the economy within the framework of neoclassical theory, even with the help of "rational expectations" (more accurately described as "consistent expectations"), is far from adequate to the needs of policy. One direction of progress is to erect theories that postulate, more explicitly and accurately than the current ones, exactly how expectations for the future are in fact formed by economic actors, and how those expectations enter into the calculations of actions. A realistic procedural theory would almost certainly have to include learning mechanisms, thus leading to historical irreversibilities in behavior (see, for example, Cyert and DeGroot (1971)).

Another candidate area of great importance is the Schumpeterian domain of long-term dynamics. The search for new products or new marketing strategies surely resembles the search for a good chess move more than it resembles the search for a hilltop. Rough stochastic models have been offered to describe the diffusion of innovations among firms, but little has been done to provide these phenomenological models with explanatory theories of the underlying search and decision processes. The computing capabilities and search strategies of

---

[5] For exactly the same reasons—the frequency of major disturbances to equilibrium—many ecologists believe that the plant communities actually observed in nature are quite as often transitional, nonequilibrium states as they are equilibrium climax communities in which only the very fittest have survived. A theory of these communities, then, requires a theory of the dynamic processes of adaptation, as well as a theory of the static optimum. See Connell (1978).

firm managers and engineers are central to any theory of firm growth or of inter-firm competition for markets.

These are some of the areas where the more vigorous introduction and exploitation of procedural considerations appear promising. I am sure they are not the only areas. I am not even sure they are the *most* promising, for I doubt whether we are better optimizers in this domain of speculation about the prospects of economic theory than we are in making economic decisions.

There exist, of course, some standard techniques for avoiding a separate theory of procedural rationality—in particular, the proposal that we simply fold in the costs of computation with all of the other costs in the general optimization problem. While that solution to the problem of scarce computational resources may suffice for certain simple questions in economics, it leaves unanswered all of the fascinating and important questions of what constitutes an efficient decisionmaking procedure, of how the structures of such procedures are related to the structures of particular kinds of decisionmaking environments, of the evolution of computational capabilities in the face of competition, and of the shape of an economic system in which effectiveness in computation is one of the most important weapons of survival.

8. Conclusion

■ In my remarks here, I have tried to survey some of the pieces of a theory of procedural rationality that have been emerging from research in the disciplines of OR, artificial intelligence, computational complexity, and cognitive simulation. All of these disciplines have been making great strides toward identifying powerful problem-solving algorithms, and at the same time, toward identifying the limits of human beings and computers in their efforts to explore large, complex problem spaces.

I have said only a little about the potential application of these results to the standard concerns of economics. Elsewhere, I have argued that there is an urgent need to expand the established body of economic analysis, which is largely concerned with substantive rationality, to encompass the procedural aspects of decisionmaking. As that need is increasingly recognized by the discipline, economists will find that there is available to them a substantial body of relevant work on these topics that has been produced by economics' sister disciplines.

References

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *The Design and Analysis of Computer Algorithms*. Reading, Mass.: 1974.

BUCHANAN, B. G. AND LEDERBERG, J. "The Heuristic DENDRAL Program for Explaining Empirical Data." Proceedings of the IFIP Congress 71. Ljubljana, Yugoslavia, 1971.

CHASE, W. G. AND SIMON, H. A. "Skill in Chess." *American Scientist*, Vol. 61 (July/August 1973), pp. 394–403.

CLAUSEWITZ, C. VON. *Vom Kriege*. Berlin: F. Dümmler, 1832.

CONNELL, J. H. "Diversity in Tropical Rain Forests and Coral Reefs." *Science*, Vol. 199 (March 24, 1978), pp. 1302–1310.

COREY, E. J. AND WIPKE, W. T. "Computer-Assisted Design of Complex Organic Synthesis." *Science*, Vol. 166 (October 10, 1969), pp. 178–192.

CYERT, R. M. AND DEGROOT, M. H. "Interfirm Learning and the Kinked Demand Curve." *Journal of Economic Theory*, Vol. 3 (1971), pp. 272–287.

DAVIS, R., BUCHANAN, B., AND SHORTLIFFE, E. H. "Production Rules as a Representation for a

Knowledge-Based Consultation System." *Artificial Intelligence*, Vol. 8 (February 1977), pp. 15–45.

FEIGENBAUM, E. A. AND FELDMAN, J., EDS. *Computers and Thought*. New York: 1963.

GAREY, M. R. "Optimal Task Sequencing with Precedence Constraints." *Discrete Mathematics*, Vol. 4 (1973), pp. 37–56.

GASCHNIG, J. "Exactly How Good Are Heuristics." Proceedings of the 5th IJCAI, Vol. 1 (1977), pp. 434–441.

GELERNTER, H. L., *et al.* "Empirical Explorations of SYNCHEM." *Science*, Vol. 197 (September 1977), pp. 1041–1049.

GÖDEL, K. "Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme." *Monatshefte für Mathematik und Physik*, Vol. 38 (1931).

KAHNEMAN, D. AND TVERSKY, A. "On the Psychology of Prediction." *Psychological Review*, Vol. 80 (July 1973), pp. 237–251.

KNUTH, D. E. AND MOORE, R. W. "An Analysis of Alpha-Beta Pruning." *Artificial Intelligence*, Vol. 6 (Winter 1975), pp. 293–326.

MARSCHAK, J. AND RADNER, R. *Economic Theory of Teams*. New Haven: 1972.

NEWBORN, M. M. "The Efficiency of the Alpha-Beta Search on Trees with Branch-Dependent Terminal Node Scores." *Artificial Intelligence*, Vol. 8 (April 1977), pp. 137–154.

NEWELL, A., SHAW, J. C., AND SIMON, H. A. "Chess-Playing Programs and the Problem of Complexity." *IBM Journal of Research and Development*, Vol. 2 (October 1958), pp. 320–335.

NILSSON, N. *Problem-Solving Methods in Artificial Intelligence*. New York: 1971.

POPLE, H. E. "The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning." Proceedings of the 5th IJCAI, Vol. 2 (1977), pp. 1030–1037.

POWERS, D. J. "Heuristic Synthesis in Process Development." *Chemical Engineering Progress*, Vol. 68 (1972).

SIDNEY, J. B. "Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Referral Costs." *Operations Research*, Vol. 23 (March/April 1975), pp. 283–298.

SIMON, H. A. "A Behavioral Model of Rational Choice." *Quarterly Journal of Economics*, Vol. 69 (February 1955), pp. 99–118.

———. "From Substantive to Procedural Rationality" in S. J. Latsis, ed., *Method and Appraisal in Economics*. Cambridge: 1976.

———. "Rationality As Process and As Product of Thought." *The American Economic Review* (1978).

——— AND KADANE, J. B. "Optimal Problem-Solving Search: All-or-None Solutions." *Artificial Intelligence*, Vol. 6 (Fall 1975), pp. 235–248.

SIMON, H. A. AND NEWELL, A. "Heuristic Problem Solving: The Next Advance in Operations Research." *Operations Research*, Vol. 6 (January/February 1958), pp. 1–10.

STIGLER, G. J. "The Economics of Information." *Journal of Political Economy*, Vol. 69 (June 1961), pp. 213–215.