Krohn-Rhodes Complexity on Decision Rules∗

Hans W. GOTTINGER
STRATEC Munich Germany
stratec_c@yahoo.com

Abstract: The notion of 'global rationality' underlying the concept of 'economic man' as generally been accepted in normative economics has come increasingly under attack by those who strive for more realistic behavioral assumptions in economic reasoning. The critique applies particularly to various optimization programs that have been implemented in economics but that have been found only of limited use in realistic, complex situations. Herbert Simon deserves credit for having observed the limitation of global rationality and suggesting a modification of this program by introducing his concept of 'bounded rationality'. To a great extent these ideas were advanced through studies of actual human thought processes. This paper relates the discussion on bounded rational decision rules to algebraic complexity of dynamic systems in the context of Krohn-Rhodes.

Keywords. Computability, Algebraic Complexity, Decision Analysis, Bounded Rationality, Problem Solving, Automata

1. Introduction

In this paper we show how algebraic measures of complexity derived from automata theory can be meaningfully interpreted in the context of 'bounded rationality' regarding individual or social choices. We show more specifically how algebraic complexity impacts individual and social decision rules. The complexity measure appears to be a natural consequence of looking at a decision rule as a finite-state automaton that computes preferences bounded by computational constraints. By factoring the decision process into component processes very much alike decomposition into component machines, we will be able to show how searching for improvement depends on 'structural' and 'computational' limitations. We argue that the three notions 'complexity', 'bounded rationality' and 'problem-solving' are intrinsically related. That is, *complexity* appears to be a structural property of any observable system, decision-making mechanism, organization or bureaucracy that imposes constraints upon computation, information, control mechanisms or decision-making powers. These limits to putative optimal functioning constrain

∗Dedicated to Herb Simon's 100th birthday (1916-2016).

rationality. Moreover, structural constraints such as complexity modify the handling, manipulation and controllability of information within the systEffective practical solutions in this environment require heuristics, search, step-by-step procedures, *ad hoc* remedies which constitute problem-solving in a task domain (e.g., problem-solving in contrast to implementation of optimal programs). In the upcoming sections we seek to establish links in a meaningful way.

2. Complexity of Decision Rules

The decision-maker identifies alternatives within a choice space and expresses preferences between two or more alternatives by acts of computation, finds alternatives 'undecidable' or incomparable' that cannot be computed. Preference statements are therefore translated into computing devices (indifference statements are kept out because of possible vagueness). The decision-maker can be represented as a simple finite state machine, decomposed according to these distinct tasks. In the first case the job to be done, e.g. computing preferences, is achieved by a simple group machine. In the second case the activity consists of a combinatorial machine, acting as a 'flip-flop' which does not compute anything. A realized decision rule therefore associates with a specific decomposition of the decision process. This, in turn, associates with a decomposition of machines into component machines that 'hooked' together (via the wreath product) realize the overall machine. Of course, the complexity of decision rules may vary; a sophisticated' decision-maker may activate more simple groups, less flip-flops, or groups that compute faster, more accurately and more reliably. Thus, the sophisticated decision-maker embodies more structural complexity in the sense given previously. Again, alternatives may be 'undecidable' or 'incomparable'. People perforce perform finite computations but the properties of their calculations — taken individually or collectively — constitute a rich system for which certain existence theorems, e.g. respecting social rationality or 'strategy-proofness' cannot be proved at the level of generality usually attempted in the literature.

A (social) decision rule is a sequential decision rule and as such is considered to be a finite state machine (associated with a finite semigroup), and according to complexity theory it has a finite decomposition. In this regard the results of Krohn-Rhodes complexity theory apply. The idea involved here is to factor a social choice process into parts (components) where the global process is modeled as a transformation semigroup associated with a social decision rule, and the local parts are represented by transformation sub-semigroups. The new tools originate from decomposition results in automata theory.

Consider a finite choice set of many alternatives X = {a, b,..., x, y, z) and let $D_i = 1$ iff i prefers x to y, $D_i = 0$ iff i is 'undecided' about x and y, $D_i = -1$ iff i prefers y to x. Let D be a nonempty set of decision rules D_i; $\mathcal{X}$ a nonempty collection of subsets of X, a social decision function (SDF) then is a function $F: ZxD \rightarrow P(X)$, P(X) being the power set. A SDF for individual i is given by $F(\{x,y\}, D_i)$, $x,y \in X$. Social decision functions are in fact decision machines in the sense that they decide propositions about accepting or rejecting social states through discrimination

(preference, non-preference calculations). By doing this, they generate as outputs decision rules and induce next states representing changes in preference profiles or configurations. In order to construct such a decision machine as a decision maker (DM) let us state the following Problem: Let $X^n = X_1 \times .. x X_n$ be the social choice set when the DM is confronted with a sequence of finitely many social alternatives. Let $A_0 \subseteq A_1 \subseteq \ldots \subseteq A_n$ be those sets of alternatives in which the DM can actually find comparisons (in the sense of preferred computable alternatives in these sets). Let A be a nonempty collection of all A_0, A_1, ...A_n. Then the DM constructs selection functions ρ_0, $\rho_{1,\ldots,}$ ρ_n, ρ_i: $X^n \rightarrow A$ such that for all $x_i \in X_i$ $\rho(x_i) \in A_i$). In a way, ρ_I constitutes a mechanism that reduces all possible alternatives to those which can be computed as actual choices. It is said that the DM accepts the decision rule $D_i(x_0, \ldots, x_i)$ if $\rho(x_0, \ldots, x_i) \in A_i$, more explicitly, accept $D_0(x_0)$ if $\rho(x_0) \in A_0$, accept $D_1(x_0, x_1)$ if $\rho(x_0, x_1) \in A_1$ etc.

Now this reduction mechanism induces the choice space to be partitioned into at least two parts, one part which is 'computable', generated by computable preference statements, the other part is 'non-computable', imposed by indecisiveness in choosing among alternatives.

Therefore, the actual choice space generated by the selection functions is derived from the following equivalence: computable choice space equals given choice space modulo non-computable choice subspace.

There is an upper bound, representing the complexity limits of search for the DM. This upper bound restricts the DM to select decision rules which are 'within calculating capacity'.

Therefore, let k(D) be the largest integer satisfying the bound such that $A_{K(D)-1} \leq A_{k(D)}$. HOW is the bound to be determined? In the context of game theory, as interactive decision-making, to model a player as a finite state automaton expresses the constraint on his computational ability by the number of states of the automaton that mimicks him (see Neyman(1985), Abreu and Rubinstein (1988), Kalai and Stanford (1988), Rubinstein (1986,1987), Gottinger(1990)) as well as Rubinstein (1998), though other related models for 'computing agents' beyond finite state automata have been used (Mount and Reiter, 2002). The underlying theory originates from work by Krohn-Rhodes on algebraic complexity through semigroups, automata models, and complexity of finite state machines (Rhodes, 2010).

3. Design vs. Control Complexity of Decision Rules

Regarding the complexity of (dynamic) finite-state systems, we distinguish between *design* and *control* complexity. To recall, under *design* complexity we assign that complexity (number) which associates with the transformation semigroup in which full use of the system potential is made. Under *control* complexity we assign that specific complexity (number) that associates with computations which keep the

entire system or at least part of it under complete control. A *qualitatively* stable decision rule would be a rule for which design and control complexity coincide. However, in most practical cases design complexity will exceed control complexity. Since one cannot assume that the control complexity of an average (unsophisticated) DM can be increased by exhortations to behave in a rational manner one should seek designs of decision rules for which there is reasonable understanding and control. Another way of looking at it utilizes H. Simon's (1973) distinction between a well-structured and an ill-structured problem. A *stable decision rule* is equivalent to a well-structured problem. An unstable decision rule results from the possible 'computational gap' which may occur in the problem-solving process. As Simon (1973, p. 186) puts it: " ... definiteness of problem structure is largely an illusion when we systematically confound the idealized problem that is presented to an idealized (and unlimitedly powerful) problem-solver with the actual problem that is to be attacked by a problem-solver with limited (even if large) computational capacities". So, in a way, if the problem-solver's control complexity is below the design complexity of the decision rule, he himself encounters an ill-structured problem, or equivalently, his decision rule is unstable. Then, it is desirable to redesign the decision rule in such a way that his ill-structured problem becomes well-structured to the extent that the new design coincides with the computational power of the problem-solver.

In chess the number of possible strategies which can be generated by a general chess-playing program for particular endgame configurations correspond to design complexity. The number of actual strategies engaged by a particular player or program corresponds to control complexity. Suppose two chess players are initially endowed with the same knowledge of the rules of the game, e.g., identical design complexity, then if in a sufficiently long sequence of repetitive plays one does better than the other, the player with a superior understanding of the game must be attributed to a higher control complexity. To a degree, design complexity and control complexity associate with 'programs of optimization' and 'programs of satisficing or bounded rationality', respectively. That is to say, design complexity pertains to computable resources available for obtaining the best possible result (e.g., involving a general optimization principle), whereas control complexity involves the best result possible given limited computational resources. The Krohn-Rhodes complexity theory offers an axiomatics that allows to determine the complexity level on the computability restrictions as outlined in the next section.

4. **Bounded Rationality**

In traditional decision theory it is generally acknowledged that at least two definitions of rationality are conceivable, depending on whether the approach is abstract (normative), based on non-contradictory reasoning, or pragmatic (descriptive), based on experience. We hold that these two concepts are not necessarily mutually exclusive, if we add one important aspect to the description of rationality, e.g. *computability.* Rationality in the normative sense is too restrictive by granting the decision-maker *unlimited computational resources* which obviously

fail to hold in view of complex (ill-structures) situations. On the other hand, rationality in the descriptive sense is too elusive and diffuse to be of any analytical or even predictive value since it violates unique links to consistency and coherence standards of normative postulates. The concept of 'bounded rationality' is offered for the middle ground where computability is a desideratum (required by most real-world problems). In practice 'bounded rationality' is an appropriate description of a decision environment or a strategy of decision when one or more of the following conditions are present: (1) *limited computational resources* of the decision maker (2) *thresholds of complexity* beyond which individuals are unable to discriminate, choose and reveal cognitive limits, (3) *ill-structured problems.* Let us take a moment to discuss the last point.

4.1 Ill-Structured Problems

Many choice processes in the real world engage in *ill-structured* problems for which solutions are not readily available or involve excessive computational resources. In contrast, a problem is considered to be *well-structured* if it satisfies a number of criteria, the most important of which relate to the existence of at least *one* problem space that provides for solvability with the help of a practicable (reasonable) amount of computation or search. Apparently well-defined problems such as theorem-proving and chess-playing in artificial intelligence turn out in many instances to be *ill-structured,* given the problem-solving power of contemporary problem-solving methods. There seems to be an intrinsic relationship between well or ill-structuredness of a problem and the threshold of complexity (in von Neumann's sense) below which a system shows regular, stable and predictable behavior but beyond which often quite different, sometimes counterintuitive modes of behavior can occur. A problem can be well-structured in the small, but ill-structured in the large. According to H. Simon (1973) "the difficulty stems from the immense gap between computability in *principle* and *practicable* computability in problem spaces as large as those of games like chess". This comment applies generally to choice processes in the social sciences.

An ill-structured problem (ISP) fails to satisfy at least one (or, more likely, several) of the listed *computability restrictions*:

DSC (Definite Single Criterion): there is a definite single criterion for testing any proposed solution,

RPS (Representation in Problem Space): there is at least one problem space in which can be represented the initial problem state, the goal state, and all other states that may be reached.

TPS (Transformation in Problem Space): attainable state changes (legal moves) can be represented via transitions in a problem space.

APPS (Accurate Prediction in Problem Space), if the actual problems involve acting upon the external world (environment), then changes of the state by

applying operators can be predicted, controlled and directed toward the goal state with any desirable degree of accuracy, conditional on the knowledge of the environmental states,

PAC (Practical Amount of Computation): all basic processes underlying the step-by-step procedure of problem-solving search involve only a 'practicable' amount of computation' so that only a practicable amount of search is needed for terminating the problem solving process.

Starting with problems regarding characteristic DSC we note that ISPs usually involve a representation of *multiple criteria*, requiring *complex trade-off statements* which in practice also increase the number of computational steps and engage PAC. In the set-up of a decision problem the trade-offs may pertain to any of the following different situations: (a) Two or more values are affected by the decision, but they are known to the decision-maker, (b) At least one of the outcomes is subject to uncertainty, e.g., involves a lottery that has to be traded against a sure prospect, (c) The power to make a decision is dispersed over a number of individual actors of organizational units representing different values or goals. These trade-off problems have been treated, one way or another, in now classical contributions to decision theory (J. Marschak and R. Radner (1972), R.L. Keeney and H. Raiffa (1976). These attempts are exclusively confined to static problems and hardly exhaust the range of ISPs. Conditions RPS and TPS refer to the dynamic nature of the problem space and require that the problem to be solved is well-defined and well-structured per se so that the goal structure is clearly determined a priori, while condition APPS alludes to the possible stochastic nature of the problem to pure uncertainty, and/or to the random character of environmental states.In our view, PAC is the crucial condition and the structure of computations is the dominating concern. To establish PAC one may search for effective heuristic procedures that to some extent substitute for computational burdens that go far beyond the information processing capability of human decision-makers. As H.Simon argues, 'practicable amounts of computation' are only defined relatively to computational power and there is a continuum of degrees of definiteness between the well-structured and ill-structured ends of the problem spectrum.

4.2 Ill-Structuredness and Bounded Rationality in Chess-Playing Games

A good paradigm of bounded rationality is provided by designing chess-playing programs. There are various reasons for studying outcomes and strategies in games in connection with the problem of complexity and problem-solving programs.

(1) First, people are involved in complex games and attempt to find good strategies. Does there exist a computer program that matches the best human play? Furthermore, if it exists, is there anything in the structure of the program that would be beneficial to be learnt by the human problem-solver? According to Newell, Shaw and Simon (1963b): "We do assert that complexity of behavior is essential to an intelligent performance - that the complexity of a successful chess program will approach the complexity of the thought processes of a successful human chess player".

(2) In the early phase computer programs as applied to a general class of problems did rather poorly, as compared to humans, but from the eighties there have been some fascinating improvements as evidenced by chess playing programs such as 'Chess 5.0' all the way to 'Deep Blue'. More than 50 years ago, in a then state-of-the-art survey, Newell, Shaw and Simon (1963b) have pointed out that there are just too many alternatives for a computer to examine each move, so an adequate chess-playing program must contain heuristics which restrict it to the examination of reasonable moves, also to win a game you need not select the best moves, just the satisfactory ones. This is still true today though the sheer size of computational power has tilted the balance toward 'brute force' search procedures (computational complexity) first at the expense and then in combination of 'self-learning' expert heuristics (structural complexity) .

(3) Studying game playing sheds a crucial light on the concept of learning in games which is not well understood. To teach an intelligent person the rules of chess, by itself, does not make him an expert player. One must have experience. If we could build effective playing programs which profit from experience we have at least some clue how to practice problem-solving in real life situations that require strategic planning.

(4) How a computer program should acquire chess knowledge is an interesting and difficult point. One way, of course, is for certain records to be built into the original program. To an extent this is done. Most recent chess-playing programs contain the sequence of moves and counter-moves for standard defenses. The situation at mid-game is more difficult, since so many positions might arise.

How was it possible that good chess-players still outperformed computer programs of chess which were much more powerful in computing strategies? The answer is that they evidently activate powerful heuristics that more than offset their lack of computational power up to a point when massive super computing power took over. (Apparently, this was evident in the latest chess contest between Kasparov and IBM's 'Deep Blue', but also the latter activates powerful heuristics !)

4.3 Heuristics

We claim that activating successful heuristics is intrinsically connected to the notion of structural complexity in dynamic algebraic systems. Chess belongs to the class of two person games with complete information and no chance moves. It is known that there exists for each board position (or more generally for each state of the game) one (or several) optimal moves. A tabulation of the optimal moves is a tremendous task. Chess has on the average over 10^{120} board positions, hence the table would have to have the same number of entries. Such a complete search for the optimal move is so enormous that it transcends the capabilities of any physical computer, in other words, 'brute force computing' was not likely to be the ultimate solution.

By designing the first chess playing program Shannon (1950) proposed two principles on which an algorithm for playing chess could be formulated:

(1) Scan all the possibilities (moves) and construct a search tree with branches of equal length. Hence, all the variants of the moves to be searched for are computed to the same depth. At the end of each variation (at the end of the branch) the position is evaluated by means of a numerical evaluation function. By comparing the numerical values, one can choose the best move in any given starting position, simply by a minimaxing procedure, i.e. averaging strategies by the evaluation function.

(2) Not all possibilities are scanned, some are excluded from consideration by a special rule, special search or pre-selective criteria (J.Pearl,1984,Chap.8). In this method, with the same computational resources, the depth of computation can be greater.

In the first case, information of high value will be treated equally with information of low value, or collecting information is uniformly assigned equal cost to each node. A substantial part of the work will be useless, i.e. not leading to a desirable goal (checkmate). This is a modified *breadth first* search with a numerical evaluation function and minimaxing procedure.

Under option (2) it appears that highly selective search, the drastic pruning of the tree or *in depth* search is likely to be more successful, to treat highly complex decision problems. For this purpose one needs a heuristic, as a rule of thumb, strategy or trick which drastically limits search for solutions, they even do not guarantee any solution at all, but a useful heuristic offers solutions which are good enough most of the time ('satisficing').

The pay-off in using heuristics is greatly reduced search and, therefore, involves a 'practicable amount of computation'.

In summarizing the experience of various chess-playing programs, we observe that some programs have put more emphasis on computing power along tree search in the direction of option (1), whereas others have traded off computing speed against sophistication or selectivity as sources of improvement in complex programs. Selectivity is a very powerful device and speed a very weak device for improving the performance of complex programs. By comparing two major chess-playing programs, the Los Alamos and the Bernstein program, we see that they achieve roughly the same quality of performance by pursuing different routes the computational vs. the heuristic approach: the first by using no selectivity and being very fast, the second by using a large amount of selectivity but not relying on computational speed. So, in a way, Bernstein's program introduces more sophistication to the chess program. Most of the major game-playing programs are based upon (local) look ahead and minimax techniques. As might be expected such programs have been most successful in games that have challenged the memory ability of human players, but not in games that require experience, thinking creativity, sophistication, such as chess. This apparently has been accomplished with 'Big Blue' with experience type learning on big data.

5. Problem-Solving

Let us start with a definition of a problem according to Newell, Shaw and Simon (1963a): ‘ A problem exists whenever a problem-solver desires some outcome or state of affairs that he does not immediately know how to attain’. To generate all kinds of task-related information that pertains to ‘problem-solving’ is to involve heuristics that reflect practical knowledge, experience, but also logical consistency, smartness, sophistication.

5.1 Theory of Problem-Solving

A theory of problem-solving is concerned with discovering and understanding systems of heuristics. A particular, interesting method is provided by GPS, consisting of means-end analysis and planning.
Problem-solving has developed into a challenging subdiscipline of artificial intelligence, but the methods and techniques used are of sufficient general interest for dealing with decision-making situations of politicians, bureaucrats or managers. It is likely that these decision-makers could improve their decisions if they make use of a formal theory of problem-solving. The state-space approach is a very appropriate problem-solving representation, since it has a natural association to dynamic algebraic systems and complexity.

Assume the existence of a finite or countable set Z of states, and a set J of operators consisting of semigroups S acting upon Z. The problem-solver is seen as moving through space defined by the states in an attempt to reach one of a desired set of goal states. . A problem is solved when a sequence of semigroup operators $S = S_1,S_2,...S_n$ could be found for some decomposition of the state-space such that a nested relationship holds for some initial state z_0 to generate the goal state

$$Z=S_n(S_{n-1},(...S_2(S_1(z_0))...)).$$

One could establish a one-to-one correspondence between the problem of finding S and the problem of finding a path through a graph. Let Z be defining the nodes of a graph, with arcs between nodes i and j if and only if there is an operation S_{ij} connecting z_i with z_j. The graphic representation of state-space problem solving has three advantages. It is intuitively easy to grasp, it leads to a natural extension in which we associate a cost with the application of each operation S_i. Finally, in many cases the next step to be explored can be made a function of a comparison between a goal state and a final state.

How does a theory of problem-solving relate to decision theory?

The ingredients of the conventional decision problem under uncertainty consist of

(i) a set of actions available to the decision-maker and subject to control by himself,
(ii) a set of mutually exclusive states of nature, one and only one of them can occur,
(iii) a set of consequences that obtain if the decision-maker chooses particular actions and a certain state of nature turns out to be true.

If the decision-maker is rational and satisfies certain consistency criteria on the choice of actions, he will attempt at maximizing expected utility or expected pay-off.
In this problem it appears that uncertainty about which event obtains is his most severe restriction in following an optimal course of actions. On the other hand, apparently, the decision-maker need not cope with computational constraints, either there are no physical or psychological limits on his ability to handle an immense amount of data, facilitating his choice problem, or else costs of computation are virtually known, so that the decision-maker need only determine his net pay-off making allowance of the computational costs.

A problem-solving situation, requiring decision-making in contrast reveals special features that could be circumscribed by *degree of difficulty, limited decision-making capabilities* or resources, intrinsic complexity in finding *acceptable or satisfying strategies* (solutions). These characteristics require adequate methods such as complexity-bounded search, heuristics etc..

Example:
Consider the description of a genuine problem in the framework of artificial intelligence.. In the 'missionaries and cannibals' problem, three missionaries and three cannibals wish to cross a river from the left bank to the right. They have available a boat which can take only two people on a trip. All can row. The problem is to get all six safely to the right bank subject to the constraint that at no time the number of missionaries on either side of the river may be exceeded by the number of cannibals on that side. To translate the 'puzzle' into a formal problem, let a state be defined by the number of missionaries and cannibals on the left bank and the position of the boat. The starting position is (3,3,L) and the goal (terminal) state (3,3,R). The permissible moves of the boat define the operators. The problem is solved in a number of steps, whereby the minimal number, if it exists, constitutes the optimal solution.
Problem-solving is certainly linked to 'survivability', given a chess position, change it into a position in which the opponent's king is checkmated. En route to this position, avoid any position in which your own king is checkmated or in which a stalemate occurs. The board positions define the states, and the piece moves the operator.
In this example the terminal state need not be fixed, but in the process problem-solving may be subsequently redefined and modified subject only to the restriction

that at no point 'survivability' is endangered (endogeneous value generation).

Among the tools for a comprehensive assessment of complex public and private decision problems, decision analysis appears to be the most comprehensive one. But comprehensive methods of decision analysis, as proposed by H. Raiffa (1968), for instance, are restricted in several ways:

(1) they are basically *off-line* procedures, i.e. limit choices to the 'givens' once stated,
(2) they limit complexity to the *determination of uncertainty* via probability,
(3) they address only to *'well-structured'* decision problems, where the whole set of alternatives is laid out before the decision-maker and where he knows how to achieve a particular course of action,
(4) they apply only to situations where the *goal structure has been fixed in advance* or no change of goals is anticipated in the process of taking a course of actions,
(5) they pertain to the computational part of decision-making using expected utility as the unique performance index, but making no use whatsoever of the strength of heuristics, sophistication, creativity, innovation etc., that is the unique feature of complex decision processes.

There have been recent criticisms on the major defects of contemporary decision analysis.
They can be loosely summarized as follows:
(a) Complexity is an outcome of physical constraints on information processing and therefore a matter of design.
(b) Complexity is a matter of economic constraints imposed by costs of making decisions.

(a) and (b) could be considered of being independent significance. The first point has been emphasized here from the view of systems complexity, as has been pointed out by H. Simon(1969) in his Architecture of Complexity. The second point, not less important, has been more related to costs of economic decision-making. As Th.S. Ferguson (1974) remarks, "one of the drawbacks of decision theory in general and of the Bayesian approach in particular, is the difficulty of putting the cost of the computation into the model." There are no doubt examples in which quick and easy rules are preferable to optimal rules for a Bayesian simply because it costs less to perform the computations. This has been taken up earlier by modeling the cost of computation as a finite-state machine (Gottinger, 1991).

An example of a physical constraint of a problem-solving mechanism, as in chessplaying programs, is given by the well-known *traveling salesman problem.*

Example:
A salesman wants to visit all cities $C_1,C_2,...,C_n$, pass through each city exactly once (starting from and returning to his home base city C) while minimizing his total mileage. The set of objects in the travelling salesman problem is the set of all acyclic permutations of the cities, i.e. the set of feasible tours. The number of these turn out to be bounded by (n-1)!/2 which is an extremely large number for moderate n. By Stirling's formula $n! = (n/e)^n$, hence n! increases very rapidly. For instance, for n = 10 the number is about 180,000 and for n= 11 it is nearly 2 million. Several exact mathematical solutions of this problem have been proposed, but they amount to sensible complete enumeration of the alternatives, that is, enumeration of the more likely cities. Such methods seem to work up to about n = 20 and then break down because of excessive demand upon computer time.

The traveling salesman problem is closely related to many other problems that are considered to be NP-hard (M.R. Garey and D.S. Johnson (1979)), the letters NP standing for 'nondeterministic polynomial'. These problems have received even more prominence by the invention of Khachiyan type ellipsoid algorithm (B. Aspvall and R.E. Stone (1979); C.H. Papadimitriou and K. Steiglitz (1998, Chap. 15)) Any problem for which an algorithm can be devised can be solved but no practical general solution may be feasible, because the solution requires an impractical amount of computational time and effort. This can be referred to as 'intractability". It is now generally agreed by computer scientists that algorithms whose computational time increases exponentially as a function of the size of input are 'intractable' or simply inefficient: The only algorithms to be considered efficient are 'polynomial time' algorithms. Relating program size n to algorithmic complexity, f(n), we see that for any f and for 'large' n the exponential f is always larger than the polynomial f. For the traveling salesman problem *exhaustive search* would constitute an exponential f, hence, be inefficient, whereas some shortcut heuristic procedure would let computational time increase as a polynomial function, constituting an efficient algorithm. Now, even among efficient algorithms some are faster than others, thus, it is important only to distinguish polynomial-time algorithms as a class from exponential-time algorithms. The speed of the algorithm is almost machine independent, thus for large n, a polynomial-time algorithm will find a faster solution on a slow machine than an exponential-time algorithm on a powerful machine. This is another hard fact to support the thesis of complexity tradeoffs between computational and structural complexity.

5.2. Algorithmic Complexity and Problem-Solving
A way to identify and classify *algorithmic complexity* i.e., the complexity assigned to the most efficient algorithm required to solve a problem provided it exists, is to count the number of arithmetic operations (additions, subtractions, multiplications, divisions) that are required in order to carry out an algorithm. A major purpose of the theory of algorithmic complexity is to characterize the intrinsic complexity of

specific computational problems. This is typically done by fixing some idealized model of computation and establishing upper and lower bounds on the amount of time or space, or other resources needed to solve a giving problem. An upper bound on complexity is obtained by analyzing the resource requirements of known algorithms which solve the problem. Establishing lower bounds on complexity is usually a much more difficult process, since it involves demonstrating that no possible algorithm can surpass a given level of efficiency.

A frequent choice for the model of computation is a Turing machine, and the problems studied are often posed in the form of recognition problems. By a 'recognition problem' we mean any problem in which some string is initially placed on the tape of the Turing machine, and the Turing machine must decide whether or not the input string belongs to some predefined set of strings. For example, the input string may be a formula in some formal logical theory, the truth of which is to be decided.

It is convenient to classify recognition problems into complexity classes defined in terms of worst-case and space bounds, which are expressed as a function of the length of the input string, for example, the class of problems as polynomial space consists of all problems which can be decided by a Turing machine using an amount of tape that never exceeds some poly nominal function of the input length. Similarly, the class 'polynomial time' consists of all problems which are decided with the polynomial bounded number of machine steps.

The complexity theory of specific problems, along the lines just described, have emerged in the past decade as an active and fruitful area of research. At higher levels of complexity (that is, where the time and space bounds are very rapidly-growing functions of the input length), this work shares much with the classical theory of undecidability. The objects of study are often formal logical theories, and the work has served to extend the negative results of undecidability theory by showing that many problems that are formally decidable are nevertheless not decidable in any practical sense because they have a very large (e.g. super exponential) time complexity.

At lower levels of complexity, however, the problems studied often concern combinatorial objects such as graphs and other finite structures, rather than logical theories. At this level, polynomial space contains many problems of great practical interest such as the traveling salesman problem, combinational assignment problems and network problems etc. which seem to be computationally infeasible, in that all known algorithms have a time requirement which increases exponentially with the size of the input. Although the question of whether this exponential growth is unavoidable is one of utmost practical interest, the present state of the theory gives little aid for resolving this question.

In the absence of provable lower bounds on the complexity of these problems, relative measures of complexity have come to play an important role. It is in many cases possible to reduce one problem to another, that is, to find a way of mapping one problem into another, so that if an efficient solution were available for the latter problem, then this would also give a way of solving the formal problem efficiently. Thus, the latter problem can be thought of as being at least as hard as the former. Various definitions of reducibility have been given which formalized this intuitive notion of reducing one problem to another. Each of these reducibility relations is a partical ordering of the class of all recognition problems.

A key notion which arises in studying the problem of reducibility is that of a 'complete' problem. A problem is *complete in a complexity class if it lies in the class and every problem in the class is reducible to it.* Such a problem can be thought of as a hardest problem for the class, relative to the particular reducibility relation which is being used.

Complete problems are of interest for two main reasons. First, depending on the complexity class, completeness may constitute evidence of computational difficulty. For example, if a problem is complete in polynomial space or complete in NP then it seems likely that the polynomial-time algorithm for the problem does not exist. This has practical value in that the decision-maker or researcher, knowing the problem is complete, can avoid wasting time looking for an efficient general algorithm and instead may look for some good approximation algorithm.

Second, complete problems provide reference measures of complexity, in relation to which the complexity of other problems and even the relations between complexity classes can be explored. For example, a problem that is complete in polynomial space serves as a kind of test case for the open question of whether polynomial time equals polynomial space. This question has an affirmative answer if and only if there exists a polynomial-time algorithm for the problem. Complete problems have been found in a number of different complexity classes, both within and outside polynomial space.

The completeness results which have attracted the greatest attention are those involving complete problems in non-deterministic polynomial-time (or NP). These are usually called NP-complete problems. NP-complete problems aroused particular interest because they included some very practical problems, for which efficient algorithms had been intensely (sought). As a result of linking these problems to each other and to the open question of whether 'polynomial' equals 'non-deterministic polynomial', the hope for finding efficient algorithms for these problems was greatly diminished. Since then hundreds of new NP-complete problems have appeared. A rather comprehensive list is found in Garey and Johnson (1979) and NP-completeness has come to be regarded as virtually certain evidence

that a problem does not have an efficient solution.

Combinatorial games and decision problems arising in assignment and allocation problems of great public interest provide an obvious target for complexity theory (Karp, 1987), because it is easy to give examples of game positions which can be described very definitely, but would seem to require a vast amount of computation to analyze.

6. **Conclusions**

Decision and choice processes could be factored into component subprocesses and these are associated with properties of transformation semigroups. A social choice process could be understood as a sequential game — an interaction between individual choice processes such that the interaction generates a SDR that is compatible with all individual choice processes. To achieve this, we use the tool of 'bounded rationality', to derive automata representing a system of social decision-making. Complexity is a crucial factor in the choice of decision rules and is related to the natural limitations of human decision-makers when it comes to recalling, memorizing and computing only relatively few items among which consistent choices can be made. In contrast to conventional social choice theory we only consider preference profiles that are in a certain sense 'computable', thus restricting the choice process to reasonable rules. Krohn-Rhodes complexity is a sort of super structure that binds elements of computational and structural complexity in time-dependent decision processes as it also embraces design, evolutionary and control complexity applicable to dynamic finite-state systems. This eems to be missing in modern guidelines on Complexity (Mitchell, 2009).

An obvious extension would consist of using complexity of decision rules as a primitive notion for an axiomatization of economic behavior that introduces special behavioral assumptions related to limited computablility. (A set of structural constraints for such an axiomatization could be linked to assumptions DSC-PAC.)Structural complexity is a measure of an algebraic structure ('module') that pertains to a class of heuristics and cuts drastically on the computational dimension of the problem-solving process. We have argued that in large-scale decision problems there is necessarily a complexity-trade-off between structural and computational complexity. The complexity theory of the algebraic theory of machines points to the fact that any non-purely-routine type operating system carries 'modules' of a simple problem-solving power as well as computational steps that can be identified with routine-type operations. This seems to explain the major strengths and weaknesses of human and computer problem-solving capabilities. The human decision-maker is comparatively strong in activating heuristic principles pertaining to structural complexity, but being restricted to depth-tree search, whereas the computer is comparatively strong in searching for many different types

of solution in a breadth- type search, emphasizing computational routines by computational power and speed. The construction of useful heuristics built into computer programs, aimed at solving major tasks of a problem-solving variety, becomes a tremendous challenge to artificial intelligence, amounting to substituting computational complexity by structural complexity.

Useful heuristics with high structural complexity must include:

(i) *long-run 'look-ahead' rules,* fixing the planning horizon,
(ii) *reasoning by analogy,* e.g. evaluating subtle patterns of change,
(iii) *depth-tree search,* e.g. exploiting more relevant information affecting the goal or payoff-structure in the search process.
(iv) *experience* entering problem recognition.
(v) *endogeneous value generation,* striking a delicate balance between local an strategic behavior.

A successful heuristic, revealing high structural complexity, should adapt these components repeatedly to the changing problem structure.

The tradeoff balance between structural and computational complexity can hardly be determined in advance, but in the history of chess-playing programs there are indications that such balance exists. By comparing two differently designed chess playing programs, the Los Alamos Program (1956) and the Bernstein Program (1958), Newell, Shaw and Simon (1963b) definitely make a statement on the complexity tradeoffs in terms of overall global performance of the two programs:

'To a rough approximation, then, we have two programs that achieve the same quality of performance with the same total effort by two different routes: the Los Alamos program by using no selectivity and being very fast, and the Bernstein program by using a large amount of selectivity and taking much more effort per position examined in order to make the selection. For instance, suppose both the Los Alamos and the Bernstein programs were to explore three moves deep instead of two as they now do. Then the Los Alamos program would take about 1000 times (30^2) as long as now to make a move, whereas Bernstein's program would take about 50 times as long (7^2), the latter gaining a factor of 20 in the total computing effort required per move'.

From this we may conclude that as the depth of the moves increases it becomes correspondingly more difficult, at some point even practically impossible, to trade off computing speed and power, as represented by computational complexity, for sophisticated heuristic search procedures given by structural complexity.

References

Abreu, Dilip and Ariel Rubinstein (1988), The Structure of Nash Equilibrium in Repeated Games with Finite Automata, *Econometrica* 56 pp. 1259-1288.

Aspvall, C.B. and Stone, R.E., (1979) 'Khachiyan's Linear Programming Algorithm', Stan-CS-79-776, Dept, of Computer Science, Stanford Univ., Nov. 1979.

Ferguson, Th.S., (1974) 'Prior Distributions on Spaces of Probability Measures', *Ann. Statist.* 2, 615-629.
Garey, M.R. and Johnson , D.S., (1979) *Computers and Intractability,* W.H. Freeman San Francisco.
Gottinger, H.W.(1990), 'Complexity of Games and Bounded Rationality', *Optimization* 21, 991 -1003
Gottinger,H.W.(1991), 'Computational Costs and Bounded Rationality', in Stegmüller, W.,Balzer,W. and W. Spohn, eds., *Philosophy of Economics*, Springer: Berlin, 223-238
Kalai E. and Stanford W. (1988), 'Finite Rationality and Interpersonal Complexity in Repeated Games', *Econometrica,* 56, pp. 397-410.
Karp, R.M., (1972) Lecture Notes, Dept, of Computer Science, Univ. of California, Berkeley, Ca.
Karp, R.M., (1987) 'Combinatorics, Complexity and Randomness'. *ACM Turing Award Lectures*, Addison-Wesley :Reading, Ma., 433-455.
Keeney, R. and Raiffa, H., (1976) Decision Analysis with Multiple Conflicting Objectives, Wiley: New York.
Marschak, J. and Radner, R., (1972) *Economic Theory of Teams,* Yale Univ. Press: New Haven.
Mitchell,M.(2009), *Complexity -A Guided Tour , Oxford Univ. Press: New York*
Mount, K.R., and S. Reiter (2002), *Computation and Complexity in Economic Behavior and Organization,* Cambridge Univ. Press: Cambridge
Neyman, A. (1985), 'Bounded Complexity justifies cooperation in the finitely Repeated Prisoners Dilemma', *Economic Letters* 19, 227-229
Newell, H., Shaw, P. and Simon, H., (1963a) 'General Report on GPS', in R.D. Luce et al. (eds.). *Readings in Mathematical Psychology* II, Wiley: New York.
Newell, H., Shaw, P. and Simon, H., (1963b) 'Chess-Playing and the Problem of Complexity', in Feigenbaum (ed.) *Computers and Thought,* McGraw Hill: New York.
Papadimitriou,C.H. and K. Steiglitz(1998) *Combinatorial Optimization,* Dover: New York
Pearl,J.(1984), *Heuristics: Intelligent Search Strategies for Computer Problem Solving,* Addison Wesley: Reading,Ma.
Raiffa, H., (1968) *Decision Analysis,* Addison-Wesley: New York.
Rubinstein, A. (1986) 'Finite Automata play the Repeated Prisoners Dilemma', *Jour, of Economic Theory* 39, 83-96.
Rubinstein, A. (1987), 'Complexity of Strategies and the Resolution of Conflict', London School of Economics, Suntory Toyota Intern. Centre for Economics, Disc. Paper 87/150.
Rubinstein, A. (1998), *Modeling Bounded Rationality,* MIT Press: Cambridge, Ma.
Shannon, C.E., (1950), 'Programming a digital computer for playing chess' *Philosophy Magazine* 41, 346-375.
Simon, H. (1972) Theories of Bounded Rationality', in C.B. McGuire and R. Radner, eds., *Decision and Organization,* North Holland: Amsterdam.

Simon,H. (1969), 'The Architecture of Complexity' in H. Simon, *The Sciences of the Artificial* , Cambridge,Ma.; MIT Press
Simon, H., (1973) 'The Structure of Ill-Structured Problems', *Artificial Intelligence* 4, 181-201.