Algorithmic Foundations for Business Strategy

Mihnea Moldoveanu

Working Paper 17-036

Algorithmic Foundations for Business Strategy

Mihnea Moldoveanu
Visiting Professor
Harvard University Graduate School of Business Administration
Professor of Business Economics
Desautels Professor of Integrative Thinking
Director, Desautels Centre for Integrative Thinking
Vice-Dean, Learning, Innovation and Executive Programs
Rotman School of Management, University of Toronto

Working Paper 17-036

Abstract

I introduce algorithmic and meta-algorithmic models for the study of strategic problem solving, aimed at illuminating the processes and procedures by which strategic managers and firms deal with complex problems. These models allow us to explore the relationship between the complexity of an environment, the sophistication of the problem solving processes and procedures used to optimally map problem statements into strategic actions, and the organizational structures that are best suited to the implementation of solutions. This approach allows us to distinguish among levels of sophistication in the strategic management of complex predicaments, specifically among rational, irrational, quasi-rational and super-rational problem solving processes and responses of strategic managers and organizations. It highlights a set of dynamic search and adaptation capabilities that can be studied via the algorithmic and computational properties of the problems they are meant to solve and the efficiency and reliability by which they search a solution space. It points to several new components of competitive advantage that are linked to the complexity adaptation of a firm: 'offline problem solving' and 'simulation advantage' emerge as key strategic differentiators for firms facing complex problems.

1. *Introduction: Strategy and Algorithmics*

All life is problem solving.

Karl R. Popper

Using tools and models from computational complexity theory and the algorithmics of hard problems that are new to the strategy field, this paper addresses the question of how strategic process and structure adapt to the complexity of the strategic scenario or predicament. Relevant literature has focused on the effects of environmental complexity on firm level decision processes and strategic outcomes (Levinthal, 1997; McKelvey, 1999) and on the barriers and obstacles to adaptation, performance and imitation that complexity raises at the level of optimization and decision processes [Rivkin, 2000; Denrell and March, 2001] and organizational structures [Siggelkow and Rivkin, 2005; Davis, Eisenhardt and Bingham, 2008]. An algorithmic perspective on the strategic problems faced by the firm opens up the opportunity to systematically explore optimal adaptations to complexity at the level of *both* problem solving processes and *organizational architectures*, and to distinguish between different levels of sophistication in the ways in which strategic processes and structures deal with complexity. The current paper pursues this opportunity by contributing an algorithmic and computational model of strategic problem solving that allows researchers to distinguish between different levels and kinds of adaptations to complexity; and to explore the fit between the canonical strategy problems a firm faces, its stock of problem solving procedures, and its architectural and procedural adaptations to complexity.

The question of strategic adaptation to complexity has received significant attention in the strategy literature [McKelvey, 1999; Rivkin, 2000; Siggelkow and Levinthal, 2003, 2005; Moldoveanu and Bauer, 2004; Siggelkow and Rivkin, 2005 Moldoveanu, 2009], which has built on the 'organizations-as-problem-solving-entities imagery and associated distinctions of the Carnegie School [March and Simon, 1958; Cyert and March, 1963] to showcase the ways in which strategic choices more or less successfully map changes in organizational structure and function to payoffs as a function of internally and externally generated complexity. This literature has used models strategic environments adapted from complex adaptive systems research [Holland, 1962; Kauffman, 1969, 1993, 1995] to study the interaction between strategic choices and consequence under complexity-induced constraints, and pointed to a range of structural, cognitive and procedural adaptations to complexity that have broadened both the scope of strategy research and the strategist's toolkit. I sharpen the focus of the study of 'strategy and complexity' by introducing an algorithmic and computational perspective on strategy making, which allows us to draw new and useful distinctions when gauging the effectiveness of strategic processes and the capabilities of strategic managers faced with complexity. It uses the lexicon of computational complexity theory and 'algorithmics' to model the ways in which strategic managers 'set strategy' by solving 'hard problems' – problems that are likely to confront them in complex environments - and shows how problem solving procedures, capabilities and the organizational structures they engender can differentiate between more or less effective strategic processes and procedures.

I build on recent work that links structural and functional complexity to the computational complexity of solving an adaptation problem in a complex environment, but expand the range of models of the firm-environment nexus that we can apply complexity analysis to. Following [Porter, 1996], Rivkin [2000] models firms as coupled activity systems, which in turn are modeled, following [Kaufmann 1969; 1993], as Boolean networks of *N* nodes with *K* edges per node, wherein the state of each node evolves as a function of the previous state of that node and the states of all of the nodes linked to it by edges. If we model strategic choices as periodic decisions over alternative coupled activity sets, and activity sets as evolving *NK* networks, we can understand the problem that the strategic manager solves as that of predicting the evolution of various *NK* networks that model the value linked activities comprising the fabric of the business. Rivkin [2000] points out that this problem is technically intractable [Weinberger, 1996] – in the sense that the number of operations required to solve it grows exponentially with *N* for $K>2$ - and infers that successful 'complex' strategies (high *N*, high *K*) are hard to imitate in the sense that successful imitation is statistically very rare. But a careful, algorithmically informed analysis of the computational complexity of optimizing NK fitness functions [Wright, Thompson and Zhang, 2000] shows that there exists a tractable *approximate* solution algorithm for optimizing *NK* fitness functions with arbitrary *N* and *K* , suggesting that complexity-related barriers to imitation may be weaker if the fitness landscape is not sharply peaked and that the imitator is in the possession of the right search procedure (the approximation algorithm whose performance they demonstrate). Such findings highlight the importance of understanding the *precise computational structure of the optimization problem that the strategist is trying to solve*. They suggest that a computational perspective should be applied to other canonical strategic problems that are known to be intractable - such as finding the Nash Equilibrium in a competitive game that guarantees the firm a set minimum payoff [Gilboa and Zemel, 1989], or finding the maximum-value configuration of a product's costly features and attributes subject to a maximum cost constraint [Moldoveanu, 2009]. Modeling strategic choice and optimization problems in terms of the algorithms that most efficiently solve these problems, and complexity in terms of the computational complexity of these algorithms extends the range of models of strategy making that we can study from a complexity perspective.

I extend previous research that models the effects of computational complexity on strategic choice and optimization by introducing a *range* of algorithmic and architectural adaptations to complexity that differ in terms of effectiveness, efficiency and reliability. They provide measures of the capability of a firm or a strategic manager to search or explore the solution space [Newell and Simon, 1972] of a strategic problem. This represents an advance on models of adaptations to complexity which posit that strategic managers systematically *avoid* certain classes of problems on the basis of their degree of difficulty. Moldoveanu [2009] uses the time complexity hierarchy (*P-NP, or,* tractable-intractable) devised for the study of hard problems to study the ways in which strategic managers solve problems, and marshals empirical and experimental evidence to argue that strategic managers systematically choose tractable over intractable problems as ways of representing their raw predicaments. He argues strategic managers exhibit lexicographic preferences over problem types (*P*>*NP*) and register or collect information solely on the variables needed to solve the *tractable* problems which they have chosen. He predicts that one of the key signatures of computational sloth - the systematic and non-adaptive unwillingness to think in terms of *NP*-hard problems on the part of managers – is systematic ignorance of a set of variables, such as network position or competitors' reaction function, that would only matter *if* managers were actually engaged in solving *NP* hard problems such as computing an optimal network strategy or finding the Nash equilibrium in their buyer-seller market). I argue, however, that brute force truncation of the solution process and lexicographic problem selection are but two of several strategies that strategic managers and firms can use to solve problems for which exhaustive searches of the solution space are computationally expensive. I show how meta-algorithms and heuristics [Hromkovic, 2003; Michalewicz and Fogel, 2004] can be both used to structure the process of solution search so that polynomial time searches will yield acceptable answers to *NP*-hard problems, and how the deployment of such algorithms can be understood to lie at the foundation of the firm's adaptive strategic advantage. We are, as a result, able to texture our understanding of the rationality of strategic managers in ways that go beyond the distinctions between 'full' and 'bounded' rationality and 'optimizing' versus 'satisficing' [Simon, 1978; 1991] and include forms of (hyper)rationality that take into account the costs and benefits of the processes by which optimization is carried out. Key to such hyper-rational patterns of strategic response is the set of solution space search procedures (algorithms) that an organization can implement. The algorithmic language introduced here gives us a set of models for measuring the *fit* between the structure of a strategic problem and the algorithmic suite of procedures for solving it.

Attempts to understand organizational responses to turbulent environments [Rivkin and Siggelkow, 2003, 2007; Siggelkow and Rivkin, 2005, among others] have also focused on examining changes in organizational structures or architectures (more/less centralized decision making authority; more-fewer couplings among outcomes of individual level decisions) to different environmental regimes (more/less uncertainty; longer/shorter required organizational response times). However, structural modifications to an organization are costly, and the time required for their implementation is sometimes shorter than that needed for a survivable response. Virtual search for optimal solutions - like 'fictitious play' in the theory of games [Brown, 1951]– can provide a low(er) cost alternative to solving a strategic adaptation problem by making *actual* changes to a firm's allocation of tasks, incentives and decision rights. An algorithmic analysis of a firm's strategic problems will be shown to give us tools for gauging a firm's strategic 'simulation advantage', which arises from its ability to explore, 'off-line', the solution spaces of difficult strategic problems more reliably and efficiently than its competitors. 'Off-line' strategic problem solving can be a useful complement to the 'on-line' exploration of fitness landscapes involved in actual experimentation, provided that the problem statement guiding the offline, virtual search closely tracks the causal structure of the problem statement that guides an online search. But, unlike approaches to simplification in strategic problem solving [Gavetti and Levinthal, 2000; Gavetti, Levinthal and Rivkin, 2005; Gavetti and Rivkin, 2006] that rest on representational simplifications – such as metaphors and analogies - to model the ways in which strategic managers think their way through complexity, the current work shows how *computational* short cuts and fast solution algorithms can also function as effective simplification devices for strategic problems.

Structural strategic adaptations to high complexity regimes are also be informed by the algorithmic models of problem solving introduced here. Some of the technically 'hard' problems of business strategy are more susceptible than others to decentralized and parallel search processes, and some problems will be more likely to yield to fully decentralized (or, random) search processes than others. The kind and degree of structural changes (such as thickening, patching, coasting and trimming [Siggelkow, 2002]) that will produce beneficial effects in the face of environmental change will depend on the algorithmic structure of the firm's problems and its solution space search procedures - in the same way in which understanding the class of problems a RISC processor is designed to solve and the class of algorithms it is designed to implement can inform optimal architectural enhancements to it.

Some work in economics [Rubinstein, 1993] has for some time now posited that agents may differ not only with respect to their preferences and beliefs, but also with respect to their logical and computational prowess. Bounds to computational prowess are also bounds to the sophistication of search procedures [Levinthal, 1997], and make the difference between 'luck' and 'skill' based explanations of strategic outcome [Denrell, Fang and Winter, 2003]. This work extends attempts in microeconomics [Rubinstein,1986; 1993 ; Gilboa and Zemel, 1989] and the theory of computation [Daskalakis, Goldberg and

Papadimitriou, 2006] to model the behavior of agents solving complicated decision problems in the face of bounds to the depth of the logical computation they can perform . It broadens the reach of the approaches of these prior papers by (1) showing how large classes of business strategy problems can be understood in terms of their computational structure and complexity, (2) how the time complexity of the associated solution algorithms can be measured by inclusion of the canonical problems in one of the classes of algorithmic problems known to be either tractable or intractable, (3) how the science of designing algorithms for hard problems ('algorithmics') [Hromkovic, 2003; Michalewicz and Fogel, 2004] can be used to understand both a firm's adaptation to complexity (by making intractable problems locally tractable) and superior performance (by developing a capability for adaptively solving hard problems under resource constraints). It extends these analyses by applying a complexity hierarchy not only to understanding how strategic managers solve the problems of their business (cost reduction, strategic pricing, tactical pre-commitment, profitable product re-positioning or industry diversification) but also to understanding the basic ways in which managers think about data, principles, rules, solution spaces, and about search itself in terms of solving problems of known complexity and structure.

Outline. I show how strategies can be understood as the outcome of computational processes and as algorithmic solutions to strategic problems, and demonstrate the generality of the model by showing how many core problems of strategy can be reduced to canonical algorithmic processes of measurable time complexity. I show that intractability in itself need not pose an insurmountable barrier to strategy: There are families of meta algorithms and heuristics for solving intractable problems 'accurately enough, enough of the time' in acceptable amounts of time and under viable resource constraints; and many strategic formulation processes can be understood as the application of such meta-algorithms to strategic problems. I put forth an adaptive capacity to solve hard problems (often corresponding to 'complex' problem solving environments) as a source of firm-level advantage, and create a three-dimensional measure of competitive advantage that incorporates the *quality* (precision, accuracy) of the solution produced, the probability of reaching that solution under time and resource constraints (its *reliability*), and the speed with which a solution of minimum acceptable quality is reached (its *efficiency*). I derive the implications of the algorithmic perspective for the empirical study of strategic procedure, process, outcome and performance in terms of a three-fold dependence: of problem solving procedures on the computational complexity and structure of the firm's strategic problems, of the optimal structure of the organization on the choice of problem solving procedure, and of the metrics and performance measures for problem solving processes on the specific procedure used to solve a strategic problem.

2. Three Motivational Examples. Consider the following as examples of the ways in which the complexity of a problem matters to the strategic and technological choices of a business, and the way in which *prior insight* into the complexity of the problem matters to a firm's strategic choices.

Example 1. Google, Inc.'s co-founders [Bryn and Page, 1998] solved the problem of producing a search engine for billions of *WWW* pages without either avoiding the problem [as Moldoveanu, 2009 would predict] or 'just stumbling', via haphazard search [Cyert and March, 1963; Newell and Simon, 1972] onto the solution. The problem that the search engine solves –that of providing a ranking of web sites in terms of relative popularity, influence or impact – can be represented as the problem of searching all of the paths through a potentially not-fully connected graph (the network of all 1 Billion *www* pages existing circa 1999) to find those containing the site of interest and tabulating the number of paths in the network that pass through each node – which would require *enumerating* all such paths, whose number is an exponential function of the number of nodes (1Bn). The insight on which the Google search engine, circa 2000, ('PageRank') is based captures the basic intuition of ranking pages that have more links pointing to them more highly, but restricts itself to measuring the relative number of citations among pairs of web sites (i.e. how many times does page *A* cite page *B*, normalized by the total number of citations *A* makes). As a result, it provides an approximate measure of importance that tracks – but does not always equal – the exact solution based on the exhaustive enumeration of all paths through the network.

Example 2. Cisco Systems, Inc.'s makes (designs, builds, sells, services) network appliances that link end users, through network hubs (switches and routers) to one another and to central transmission points. It is constantly solving the problem of providing mutually interoperable hardware, software, firmware and netware 'stacks' (protocols, encoding, decoding and signal formatting algorithms running on custom and general purpose silicon) aimed at matching customer demands with the lumpy characteristics of existing platforms, the hard-wired specifications of a large number of *IEEE* and *ETSI* technical standards and a set of gross margin and operational cash flow constraints. Rather than executing 'brute force' searches of all possible solutions to this very large scale optimization problem, Cisco evolved into an 'acquisition machine' of technology firms (and proceeded to acquire some 180 firms as of 2011, an average of 6.7 firms acquired per *year*) that independently and asynchronously searched the solution search space, under the influence of high powered incentives provided by venture capital investments and the prospect of a high-valuation 'exit' (in the form of an acquisition by Cisco). This strategy seems to heed the well-known insight [Garey and Johnson, 1983] that the problem of *evaluating* the set of solutions to an intractable problem for satisfaction of a performance target is a *tractable* problem. The difficulty of intractably large optimization problems consists in the exhaustive *enumeration* of the solutions in the search space, which must precede the independent evaluation of each candidate; uncoupling *enumeration* from *evaluation* results in a massive decrease in the computational complexity of the task. By contrast, what used to be Nortel, Inc. became famous for a 'not-invented-here' syndrome that saw the company attempt to develop and re-develop, from scratch, all of the hardware and software stacks for its network appliances and base stations, ranging from DMS switches to WiMax base stations. Cisco today has a market capitalization of over $100 *Bn*, whereas Nortel filed for bankruptcy in 2010.

Example 3. The securitization of cash flows arising from mortgage debt repayments is widely believed to have contributed to the worldwide financial crisis of 2007 and 2008, but the connection between the nature of the crisis of confidence and trust in inter-bank lending and acquisition of mortgage derivatives has only recently been signaled [Arora, Barak, Brunnermeier and Ge, 2010]. The complexity connection revolves around the well known 'market for lemons' model of Akerlof [Akerlof, 1970]: if buyers know that 20% of the used cars in the market are 'lemons', the price of a used car is $10,000.00, and they cannot easily tell lemons apart from reliable cars, then they will not be willing to pay more than $8000.00 for a used car, which would cause those who do sell reliable cars to pull out of the market (because they do not want to sell at a 20% discount), leaving the market to grind to a halt. It is possible for Bank of Limonia, say, to create a set of iN derivatives (CDO's) whose payoff depends on the probability of default of up to *M* different mortagegs (or, tranches of mortgages) coming from *C* different risk classes in such a way that a buyer who wants to figure out whether higher-probability-of-default-mortgages ('lemons') are reflected in any pool of securities would need to calculate whether there is any class of mortgage that is over-represented in the design of the derivatives, which requires considering all of the links in the bi-partite graph linking asset classes and derivatives, which is a computationally intractable problem. [Arora et al, ibid] show that a computationally bounded buyer would not be able to figure out whether or not the lower payoff is due to change or to the special design of a lemon by the seller. *Knowing* this, buyers should rationally choose to stay away from the market altogether.

These examples highlight the importance of complexity – as a measure of the difficulty of solving certain problems – to both the specific design of a strategy and to strategic outcomes. Moreover, they highlight ways in which *insight* into the complexity of a large scale optimization or decision problem can be advantageously incorporated in the strategy making process.

3. Managerial and Organizational Algorithmics: The Study of the Computational Structure of Business Problems and Problem Solving Processes. The computational (or, 'time') complexity of a problem has become an established measure that indexes the worst- case number of operations that the most efficient known procedure – or, algorithm – for solving that problem will take up [Cook, 1971 is the seminal paper; Papadimitriou, 1994 provides a thorough review; Rivkin, 2000, Moldoveanu and Bauer, 2004, Moldoveanu 2009 apply the measure to quantifying the difficulty of solving business problems]. Computational complexity is expressed as a function, *C(N)*, of the number of independent variables *N* of a problem statement. These variables may be the set of arguments and constraints of an optimization problem (which produces a vector or scalar corresponding to the optimum as a solution) or a decision problem (which produces a *0* ('no') or a *1* ('yes') as the answer. The functional form of *C(N)* lets us distinguish among different types of strategy problems. Some problems can only be solved by algorithms with time complexity that grows very quickly as a function of the size of the input, whereas others can be solved by algorithms whose time complexity grows rather more slowly.

'Very quickly' and 'more slowly' need tightening: the *polynomial-time* property was introduced to provide it. Polynomial-time problems (P) are those problems that can be solved by deterministic solution algorithms whose (worst case) complexity is *at most* a polynomial function of the number of input variables, i.e. $C(N) = P^k(N)$ is the k*th* degree polynomial of the argument, where $0 \leq k < \infty$. By contrast, Nondeterministic Polynomial Time (*NP*) problems are problems that can be solved *non-deterministically* by polynomial time algorithms or *deterministically* by algorithms with super-polynomial (eg exponential, factorial) time complexity - i.e. $C(N) > P^k(N)$ for any k, and, typically, $C(N) \geq e^{kN}$. The relevance of the exponential term in rendering the exact solution of problems with even small *N* (30-100 variables) impractical for state of the art computational devices is well covered in standard references [eg Garey and Johnson, 1983; Papadimitriou, 1994]. *NP*-hardness (for optimization problems; *NP*-completeness for decision problems) is the key signature of *intractability* of a problem, as a problem solver will predictably run out of time when trying to solve it via a deterministic algorithm that exhaustively searches its solution space. Because of this, intractability is sometimes held to entail *practical* unsolvability [Rivkin, 2000]. But, as we shall see, intractability need not entail unsolvability: many intractable problems will yield to approximation and randomization-based search procedures which can provide 'good enough' solutions which, moreover, can be monotonically improved through additional computation.

The algorithmic study of the ways in which strategic managers solve problems entails (Figure 1) the articulation of well-defined, well-structured (*WDWS*) problems that represent the predicaments faced by the firm's managers and the encoding of these problems via a set of canonical problem statements (decision or optimization problems) whose complexity can be measured. We can thereby classify them as easy, hard or intractable and study of the ways in which strategic managers and firms go about solving them by comparing the managers' problem solving processes with 'optimal' or 'quasi-optimal' algorithms that have been designed to solve canonical problems in various complexity regimes.

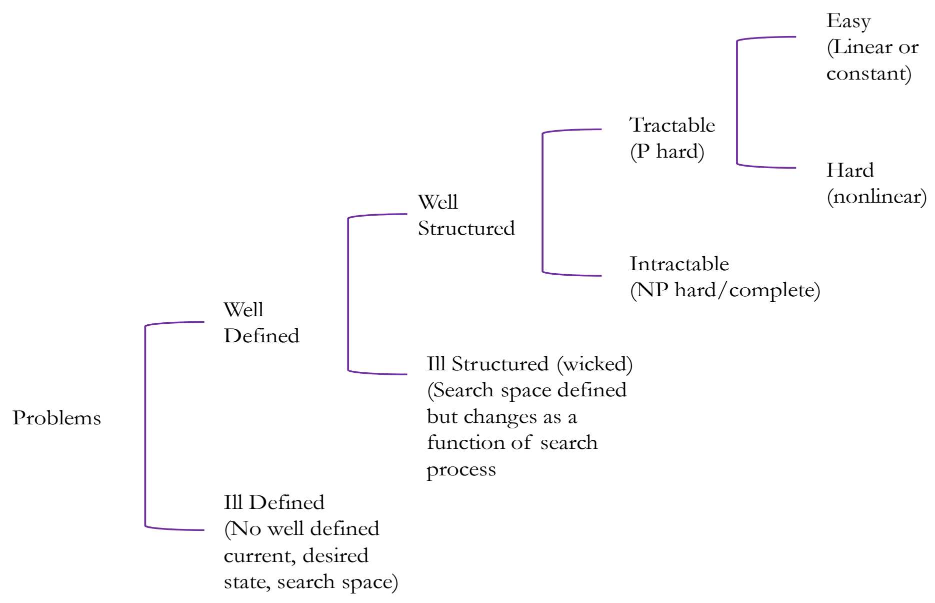

Problems
Well Defined
Ill Defined (No well defined current, desired state, search space)
Well Structured
Ill Structured (wicked) (Search space defined but changes as a function of search process
Tractable (P hard)
Intractable (NP hard/complete)
Easy (Linear or constant)
Hard (nonlinear)

Figure 1: Business Problems: A Map

Figure 2 represents the modeling operation by which problems of strategic managers are encoded as canonical problems whose time complexity can be estimated. *P* hard problems encode both 'easy' and 'hard' problems that are nonetheless tractable as a canonical set of problem statements such as linear optimization, searching, sorting and ranking of options, pattern matching. Appendix I presents a canonical 'library' of *P*-hard problems useful for the study of strategy because their solution algorithms can be used to encode a large number of production tasks that a strategic manager or management team undertakes.

NP hard (complete) problems encode intractable problems, whose complexity grows exponentially or super-exponentially as a function of the number of variables in the problem statement. They include problems of finding the *equilibria* of competitive games, designing strategic products or solutions with lumpy constraints and optimizing the network flow of information, trust, matter and money. Appendix II presents a library of canonical *NP*-hard/*NP*-complete problems that encode another large class of problems of strategic management. The Appendices are summarized in Table 1.

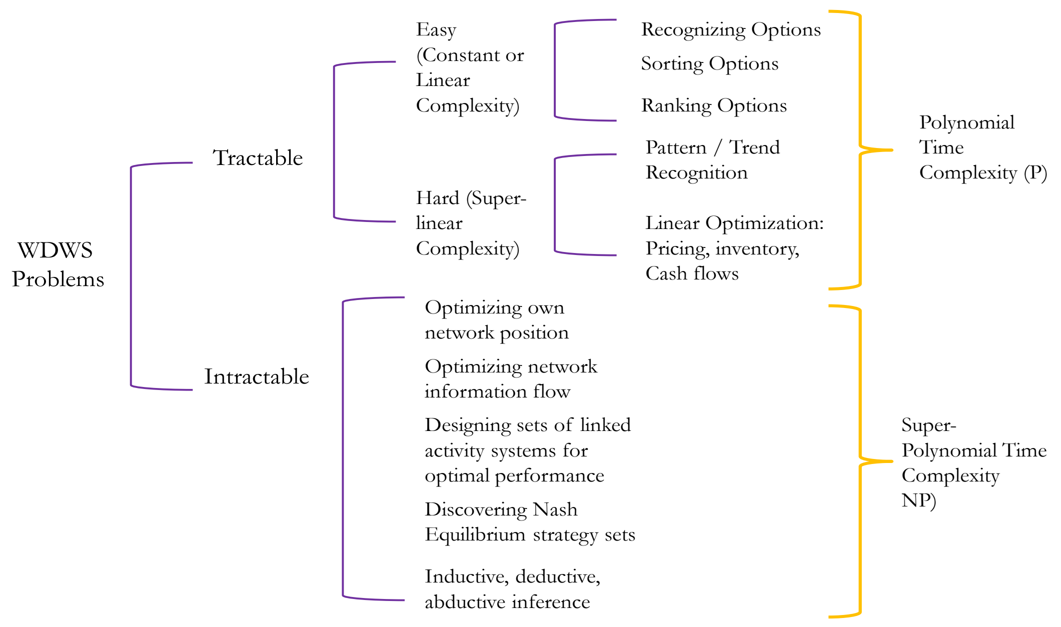

WDWS Problems
Tractable
Intractable
Easy (Constant or Linear Complexity)
Hard (Super-linear Complexity)
Recognizing Options
Sorting Options
Ranking Options
Pattern / Trend Recognition
Linear Optimization: Pricing, inventory, Cash flows
Polynomial Time Complexity (P)
Optimizing own network position
Optimizing network information flow
Designing sets of linked activity systems for optimal performance
Discovering Nash Equilibrium strategy sets
Inductive, deductive, abductive inference
Super-Polynomial Time Complexity NP)

Figure 2: Using Algorithmic Complexity Classes to Map the Problems of Business

Canonical Problem	Problem Statement	Complexity Class	Examples of Managerial Problems It Encodes
Correlation	Calculate correlation between two vectors (e.g. time series)	P	Reasoning by analogy: Find situation/object with characteristics most similar to current one
SORT	Order a random list according to a criteria	P	Rank a set of alternatives in terms of payoffs, desirability/utility
K-SAT	Given a set of elementary propositions, determine whether or not they satisfy a set of clauses	NP	Predict evolution of densely connected value-linked activity chain, modify a contractual agreement in a self consistent fashion.

Canonical Problem	Problem Statement	Complexity Class	Examples of Managerial Problems It Encodes
TSP (Travelling Salesman Problem)	Find minimum distance path connecting N locations	NP	Optimize workflow on assembly line, optimize information flow in network
KSP (Knapsack Problem)	Find optimal subset of objects of known value and volume that fit into a finite volume bag	NP	Find optimal set of product features of fixed cost that will maximize profit
CLIQUE	Determine whether or not a graph G has a clique of size K	NP	Find strategic equilibrium of a complexity game that has a given minimum payoffs; find specific cliques in a organizational or inter-organizational network.
COVER	Find minimal set of subsets of S whose union covers S	NP	Find the optimal set of predictors for the values of a variable {Y}
MIN VERTEX COVER	Find minimal set of vertices in a graph that span all of its edges	NP	Network search problems: find maximally connected set of organizations or individuals in a network.

Table 1: Canonical Problems of Known Time Complexity (Columns 1,2) and the Problems of Strategic Management That They Can Be Used to Encode (Column 4).

Canonical problem statements provide a toolkit for analyzing not only the substantive problems of business strategy (selection of the optimal activity set, large scale optimization of cost structure via operational optimization, strategic product and platform design choice of partners in a strategic network in order to maximize network advantage, the strategic design of contractual arrangements by the construction of a set of clauses that map states of the world and actions of the contractants onto payoffs via jointly agreeable mapping functions) , but also the fundamental *modes of thinking* of strategic managers as they tackle the problems of their business [Moldoveanu, 2011]. These include 'one-off, frugal' heuristics such as the recognition heuristic and one-reason decision making (Figure 3a), as well as more complicated and systematic patterns of thought such as regression-based learning (EXACT COVER), inductive (COVER), deductive (K-SAT), and abductive (KSAT, COVER) (Figure 3b). The resulting map is as much a canonical map of the problems of strategy as it is a map*ping* of strategic managers' modes of *thinking through the problems of strategy* (Figure 3b). One of the advantages of the canonical approach to describing strategic thinking *qua* computation is the availability of a common language for describing *both the products and the processes* of managerial thinking.

Implications for Strategic Managers' Cognitive Simplification of Problems. The important point is that the basic patterns of inference that managers *would* need to use to make optimal inferences (the best explanation from among a number of explanations all supported to some extent by the available data set (abduction); the best inference to a set of rules or regularities that most parsimoniously explain a data set (induction); self-consistently adding an explanatory axiom or assumption to a set of axioms in a model (deduction)) can itself be understood as solving an intractable (*NP* hard) problem. *Informational* restrictions on problem statements [Denrell, 2007; Denrell and March, 2001] and cognitive frames and metaphors meant to provide simplifications of such inference problems can be understood as simplifications of the *computational* task of solving the full inference problem (deductive, inductive, abductive) which is unsolvable for high numbers of independent variables, especially when the problem is in the NP (intractable) complexity class). The implication of this mapping for strategic problem solving research is that the use by strategic managers of simplifications and informational restrictions is doubly contingent: it will vary *both* with the number of variables in the problem statement (*N)* and with the computational complexity of the problem (*P, NP*).

Heuristic or Algorithm	Upper-bound on Computational Complexity	Complexity Class	Relative Frequency of Utilization (predicted and/or estimated)
Availability heuristic	MN^2	P	Presumed High
Representativeness Heuristic	MN^2	P	Presumed High
Recognition Heuristic	M	P	High
Minimalist Heuristic	M(N+1)	P	Medium
Dawes' Rule	N(2M-1)+M	P	Low→Medium
Franklin's Rule	N(2M-1)+M	P	Low→Medium

Figure 3a: The Computational Complexity and Complexity Class Partitioning of Commonly Used Heuristics for Making Inferences About M Entities, Using N Available Cues That Offer Discrimination Value Between Any Two of the M Entities.

Mode of Thinking	Algorithmic Model	Time Complexity / Class
Analogical	Correlation	Super Quadratic (P)
Deductive	Satisfiability	Exponential (NP)
Linear Regression	Set Cover	Exponential (NP)
Inductive	Set Cover	Exponential (NP)
Abductive	3 Sat, Cover, Exact Cover	Exponential (NP)

Figure 3b: Encoding of Systematic Modes of Managerial Thinking in Canonical Forms.

4. Algorithmics for Hard Business Problems: Dynamic Search Capability as Strategic Ingenuity and Degrees of Rationality. The decomposition of the 'hard problems of strategy' that emerges via encoding strategy problems by intractable (*NP* hard optimization problems and *NP* complete decision problems) canonical problems suggests that many – and perhaps most – problems of business strategy are intractable. One can make sense of this result in several ways. One approach [Rivkin, 2000] is to postulate that strategic problems that can be accurately modeled by intractable canonical problems are *only very rarely solved in real time* by managers or teams of managers. The *NP*-hardness of the problem of predicting the evolution of a *NK* network that models value-linked activity chains, for instance, is held to entail that complexity is hard to either imitate or replicate, and that it can be a source of local competitive advantage because of the difficulty of replicating it. Another approach [Moldoveanu, 2009] is to posit that *NP*-hard problems are *systematically avoided* by strategic managers in practice by a mechanism of 'lexicographic preferences over complexity classes' of problems: Strategic managers prefer to solve *P* hard problems rather than *NP* hard problems, *even* if the cost of solving the two problems is equal, which can occur for low N (for instance: for $N=4$, $2^N<N^3$) and will, accordingly, restrict their attention span to variables that they need to solve *P* hard and not *NP* hard problems (Figure 4). They will, accordingly, *encode* unstructured situations and difficulties ('predicaments') via problem statements drawn from the *P* class, and avoid using *NP* class problem statements to turn their predicaments into *WDWS* problems.

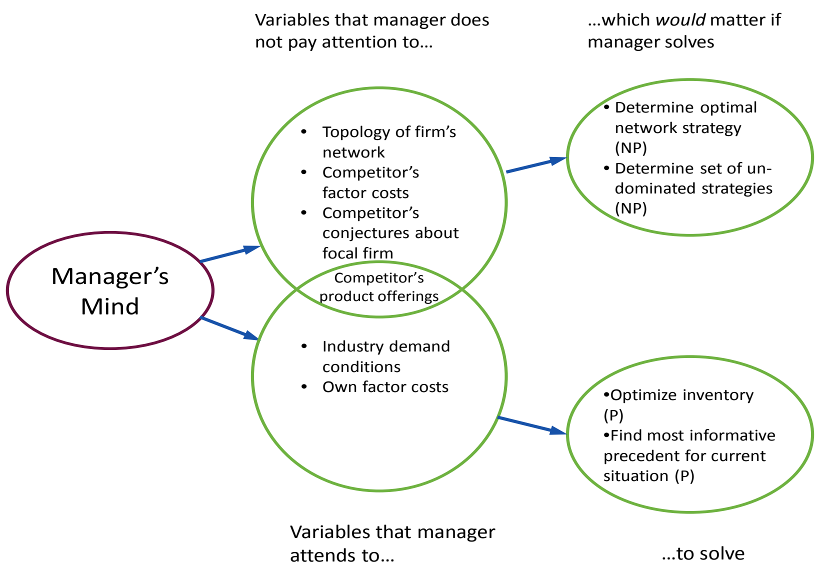

Variables that manager does not pay attention to...
...which *would* matter if manager solves
Manager's Mind
• Topology of firm's network
• Competitor's factor costs
• Competitor's conjectures about focal firm
Competitor's product offerings
• Industry demand conditions
• Own factor costs
• Determine optimal network strategy (NP)
• Determine set of un-dominated strategies (NP)
•Optimize inventory (P)
•Find most informative precedent for current situation (P)
Variables that manager attends to...
...to solve

Figure 4: Synthesis of "Inattentional Blindness" in Strategic Managers Induced by Lexicographic Choices of Problem.

However, it is also possible that strategic managers and firms formulate and sometimes successfully solve *NP*- hard optimization and *NP*-complete decision problems by *adaptive* approximations. Strategic managers tackle hard design problems (finding the optimal set of features of a product, subject to *lumpy* cost and multiple compatibility constraints), craft complex contractual arrangements (articulating the optimal set of internally consistent clauses compatible with a set of external constraints and expectations), perform root cause analyses of challenges and difficulties (finding the minimal network of propositions or hypotheses that best explain a set of data) and attempt to find specific subsets of *Nash Equilibria* of pricing and pre-commitment games their firms engage in. They need not do so by engaging in brute force searches that exhaustively plough through solution spaces. 'Randomization', 'satisficing' and 'muddling through' have for a long time been recognized as hallmarks of organizational problem solving [March and Simon, 1958; Simon, 1991]. But, this classical picture of satisficing and randomizing *can and should be substantively refined in view of the various adaptations to complexity that computational complexity theory contributes*. I show that less-than-exhaustive approaches to solving intractable problems are characterized by varying levels of *ingenuity*, which I distinguish from organizational *intelligence* as follows: whereas organizational intelligence relates to the sheer level of computational work (as in the intensity of search) that an organization can coherently organize and marshal towards solving a given problem, the ingenuity of the organization relates to the stock of alternative search procedures (algorithms and meta algorithms) that can radically enhance the speed of the search process over that achieved by exhaustive search *and* the desirability of the solution it produces relative to that produced by blind randomization or other non-adaptive satisficing manoeuvres.

The Motivational Examples, Revisited. Let us re-examine the examples introduced earlier through the lens of the proactive management of and adaptation to computational complexity.

Example 1: Google, Inc. The problem of computing a measure of the relevance or salience of every one of up to 1 x 10^9 www pages is computationally hard on any of the measures of 'centrality' (in-degree betweenness, in-degree Bonacic) that could function as plausible proxies, and which would have involved enumerating all or most of the permutations and combinations of web pages. Moreover, this problem would need to be solved *at least once a day* by any search engine claiming to provide an 'ordering' of the web pages that provides a timely input to an interested user. Page [2001] created a special centrality measure ('Google' centrality) that ranked the number of times a web page was cited by any of up to *k* web pages *relative* to the number of times these pages cited *other* pages and *weighted* by the centrality measure of the pages that cited the original page, and thereby created a highly sparse $10^9 x 10^9$ matrix that could be inverted (an order of 10^{18}) operations, executable within a day by a set of linked, parallel computational devices – a patent worth over \$300MM in options on Google shares to Stanford University after Google's IPO in 2004. The key to the success of the search engine is the special adaptation to complexity that it affords to one who wishes to compute centrality measures for the nodes of a very large network in a short period of time.

Example 2. Cisco Systems, Inc.'s 'production function' can be represented by a massive collection of *COVER* problems (see Appendix 2) representing the set of logical consistency checks needed to maintain inter-operability and logical consistency among many different networking and communication protocols. *COVER* is *NP* hard, but the solution procedure can be broken up into an *enumeration* step – whereby all possible combinations of the basic elements (standard specs, for instance) are laid out, and an *evaluation* step, whereby each of the solutions is evaluated for consistency. Cisco's targeted growth-through-acquisition strategy capitalized on the fact that both steps can be parallelized, which means that the generation and partial verification of solutions is provided by the market (fuelled by the high powered incentives provided by venture-backed firms). More generally, in the context of an industry with high network externalities in which generation of solutions that 'fit' is computationally intractable, a growth through acquisition strategy is coherent with the insight that the problem of *verifying* the solution to an *NP*-hard problem is only *P*-hard [Papadimitriou, 1994].

Example 3. The problem of detecting tampering in a large number of derivatives that have been constructed from a large collection of mortgages that fall into many different risk classes is computationally equivalent to that of identifying dense sub-graphs of a large bi-partite network [Arora et al, 2010], which is *NP*-hard. To a buyer that either does not have the computational resources to find the solution to the detection problem but understands that the problem exists, the problem will be unsolvable, and the buyer will rationally choose to stay out of the market, as Akerlof 'market for lemons' model predicts. If the prospective buyer does not understand the computational structure of the detection problem, the problem is non-existent. However, a buyer that is armed with a *P*-hard approximation algorithm [as can be found in Charikar, 2000] for the detection of the dense subgraphs in a network that gives a performance guarantee (it will not overestimate or underestimate by more than a factor of *2*, or of *1.x*) there arises the possibility of investigating a pool of derivatives with respect to the degree to which they have been tampered with, and thereby rejecting the 'lemons' on the basis of a due diligence process that does not overwhelm computational resources.

These cases highlight the importance of a structural (Cisco)or functional (Google) adaptation to computational complexity. They point to the extent to which ingenuity – or, the complexity-adaptive deployment of intelligent computation – can make a very large difference both to strategic choice and ensuing dynamics. An algorithmic and computationally informed study of strategic problem solving offers additional 'resolving power' above and beyond the classical 'satisficing-optimizing' distinctions inherited from the Carnegie School (Figure 5). Strategic managers faced with computationally intractable problems have traditionally been deemed boundedly rational to greater or lesser degrees depending on whether their solution search procedures are more characteristic of *satisficing* approaches (random guessing, one reason or one criterion decision making) or optimizing (commonly assumed to consist of exhaustive searches among possible solutions – including multiple *optima* in the case of optimization problems and choosable options in the case of decision problems). However, as the examples above highlight, strategic managers can also function as 'meta-optimizers' who seek *procedures* for solving computationally hard problems that are *optimal given the structure and the complexity of the problem and the time and resource constraints available for solving it* –i.e 'adaptively optima;'.

Is there a Silver Bullet Among Intractable Problem Solving Procedures? It is tempting to ask: is there a problem solving procedure that 'beats out' all others on *any* given problem? A powerful result from the theory of search [Wolpert, 2001] suggests that the adaptation of problem solving procedures (algorithms) to the structure of the problem faced is a required pre-condition for successful adaptation: Wolpert proves that no single algorithm or family of algorithm can statistically or deterministically dominate *any* other algorithm against *any* hard problem. Thus, 'meta-rationality' – as an adaptation to complexity – is grounded in the insight of the strategic manager into the computational complexity and structure of the problem under consideration and the *range of problem solving procedures* that strategist and organization can devise and implement.

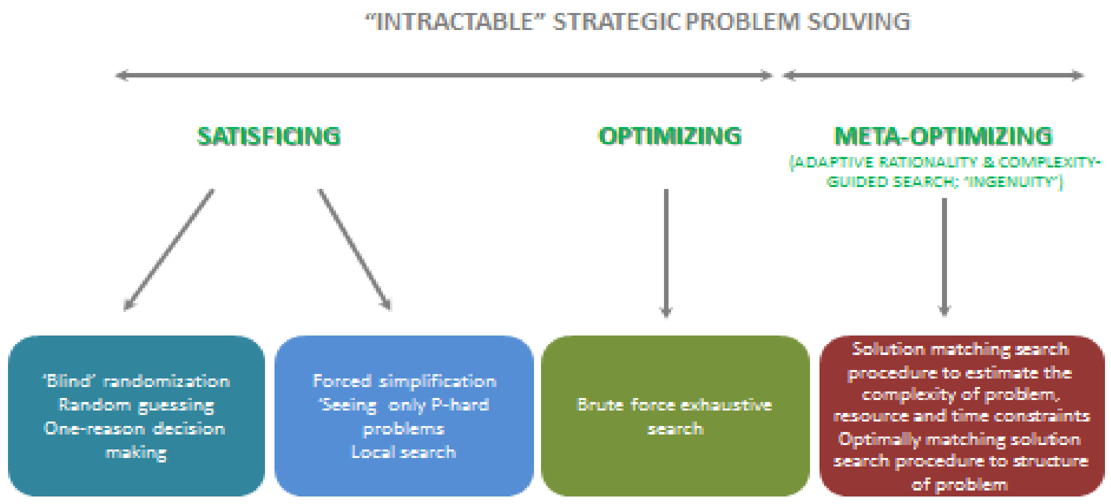

"INTRACTABLE" STRATEGIC PROBLEM SOLVING
SATISFICING
OPTIMIZING
META-OPTIMIZING
(ADAPTIVE RATIONALITY & COMPLEXITY-GUIDED SEARCH; 'INGENUITY')
'Blind' randomization Random guessing One-reason decision making
Forced simplification 'Seeing only P-hard problems Local search
Brute force exhaustive search
Solution matching search procedure to estimate the complexity of problem, resource and time constraints Optimally matching solution search procedure to structure of problem

Figure 5: Illustrating Degrees of Computational Intelligence and Exploratory Ingenuity in Strategic Problem Solving.

Strategic Ingenuity: Intelligent Adaptations to Complexity. The study of strategic problem solving stands to learn a lot from feats of *NP*-hard problem solving (Figure 6) such as that involved in solving a 4663 city TSP - wherein brute force would require *C(4663-TSP)~ 5 x* 10^{1403} calculations, which would take 1.6 x 10^{1383} years on a state of the art machine - in about 6.9 minutes on a non-state-of-the-art Pentium-powered *PC* via a local neighborhood search algorithm devised by Lin and Kernighan [1973]: The degree to which a problem is ‘solvable’ within the available resource limitation of the problem solver depends on (a) the structure of the problem and (b) the nature of the solution search procedure. The Lin-Kernighan ‘local neighborhood search’ heuristic was used to produce a reduction of 10^{1384} in the time required to find the shortest path linking the 4663 cities.

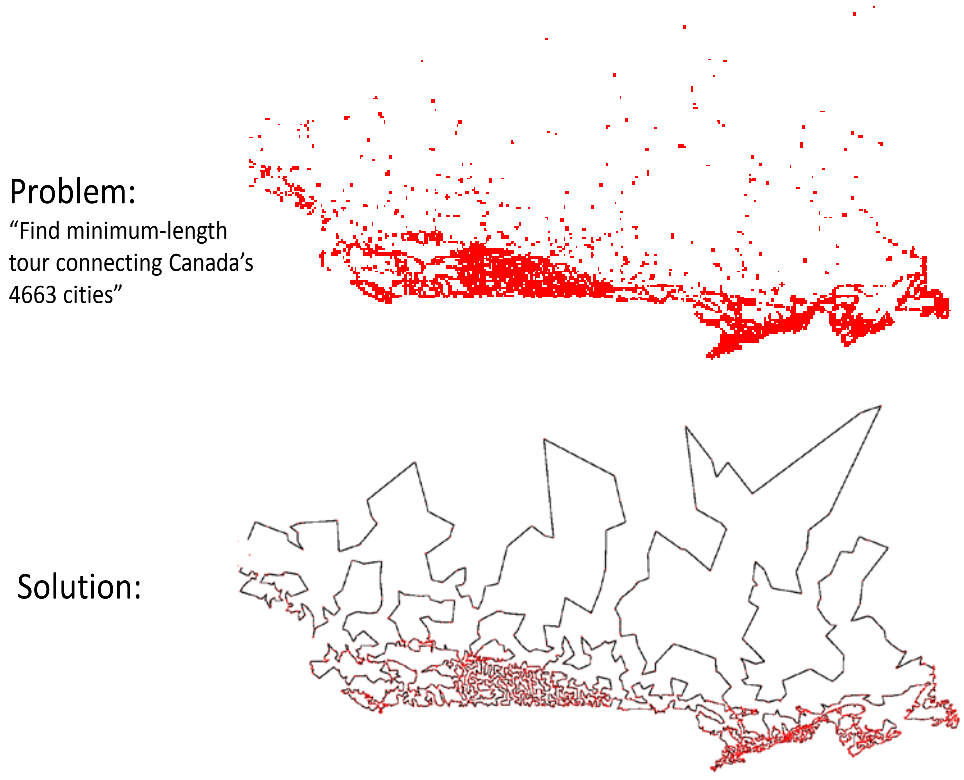


Figure 6. A TSP Problem Search Space for Canada’s 4663 Cities and Solution to the Problem Using Lin Kernighan Local Search Heuristic

The examples above suggest that one should look carefully for a systematic way to study 'good enough' solution procedures for hard problems ('ingenuity') – as well as to precisely characterize 'good-enough-ness' on a problem-specific basis - and that is what the field of 'algorithmics for hard problems' [Hromkovic, 2003; Michalewicz and Fogel, 2004] provides. I examine families of algorithms and algorithm design techniques ('meta-algorithms') for solving intractable problems, and show how the judicious deployment of such techniques can form the basis of both 'adaptive strategy-making' and of a new measure of adaptive advantage.

Table 1 provides a road map to this project: Faced with a difficult (intractable) problem, strategic managers can engage in a full, exhaustive search of the space of solutions (which may exceed the computational and operational limits of the strategist and the firm); they can engage the kind of blind, random guessing at a solution that characterizes 'muddling through'; or they can use complexity-adaptive search methods which involve intelligent approximation and randomization to arrive at a good enough solution most of the time. In order for a model to add value to a research enterprise, this approach should allow researchers to make new and more precise distinctions that can be linked to measurable properties of the process and outcome of strategic search. Table 2 summarizes the ways in which intelligent approximation and randomization procedures for solving intractable problems can be distinguished at the levels of process and outcome from blind randomization - 'just guessing' and 'muddling through', as elaborated below.

Algorithmic or Meta-Algorithmic Procedure for Solving Computationally Hard Problem	**How it Differs from Blind Search and Random Guessing at the Level of Process and Outcome Pattern**
Branch and Bound	***(Strategic Solution) Process***: rapid, tree-structured generation and evaluation of trial strategic solutions and elimination of under-performing families of solutions; ***Outcome (performance) Pattern***: lower number of trial solutions, lower probability of competency traps (local optima), faster convergence to global optimum.
Divide and Conquer	***(Strategic Solution) Process***: early partitioning of strategic solution search space into more easily searchable sub-spaces, parallelization of the search effort; ***Outcome (performance) Pattern***: lower number of trial solutions (local optima), ower probability of getting caught in local optima, faster convergence to global optima.

Algorithmic or Meta-Algorithmic Procedure for Solving Computationally Hard Problem	**How it Differs from Blind Search and Random Guessing at the Level of Process and Outcome Pattern**
Local Neighborhood Search	***(Strategic Solution) Process***: early iterative modification of a promising trial strategic solution and evaluation of modified variants vis a vis first guess and subsequent modifications, continued iterative search around local optima; ***Outcome (performance) Pattern***: clustering of intermediate solutions around a local subset of the solution search space; lower *spread* of the local optima to which solution procedure converges; lower probability of settling into local optimum; faster rate of convergence to global optimum.
Stochastic Hill Climbing	***(Strategic Solution) Process***: early generation of multiple starting points and trial solutions for the strategic search process, frequent evaluation of solutions vis a vis one another and vis a vis solution criteria; ***Outcome (Performance) Pattern***: lower probability of getting caught in local optima, higher probability of escaping local optima and finding global optimum, faster convergence to global optima.
Genetic/ **Evolutionary Search**	***(Strategic Solution) Process***: early and parallel generation of *components* of global solution, frequent alteration and recombination of candidate components into multiple candidate solutions, frequent evaluation of overall solutions vis a vis one another; ***Outcome (Performance) Pattern***: fast generation and elimination of many local optima, lower probability of settling on any one local optimum, slow(er) convergence to a global optimum.
Relaxation to Linear Optimization	***(Strategic Solution) Process***: smoothing of integer or lumpy constraints, followed by global search of smoothed solution search space; ***Outcome (Performance) Pattern***: very low probability of finding or settling in local optima, very fast convergence to (sometimes sub-optimal) 'almost-global' optimum.

Table 2: Illustrating How Strategic Solution Search Procedures Adaptive to Difficult Problems

Branch and Bound (BB) techniques rely on a partitioning of the solution search space via a tree whose nodes represent decisions among different elements of a solution, calculating bounds on the performance of a solution that arises from different branches of the tree, and deleting from the search space branches likely to result in a sub-optimal solution. The key feature of a good tree structure for *BB* methods is that it is 'prunable': estimates of performance bounds for different branches are calculated in advance, to lower the chances that an optimum be 'missed' by the resulting search. A search tree for solutions to a *4*-variable *MAX SAT* problem is shown in Figure 7a. If the MAX SAT problem is the same as that in the Appendix II, wherein $F=(X_1\hat{}\sim X_2\hat{}X_4)\&(X_1\hat{}\sim X_2\hat{}\sim X_3)$, then, the *BB* meta-algorithm will classify the various paths through the search tree and save by searching only part of the entire search space (Figure *7b*).

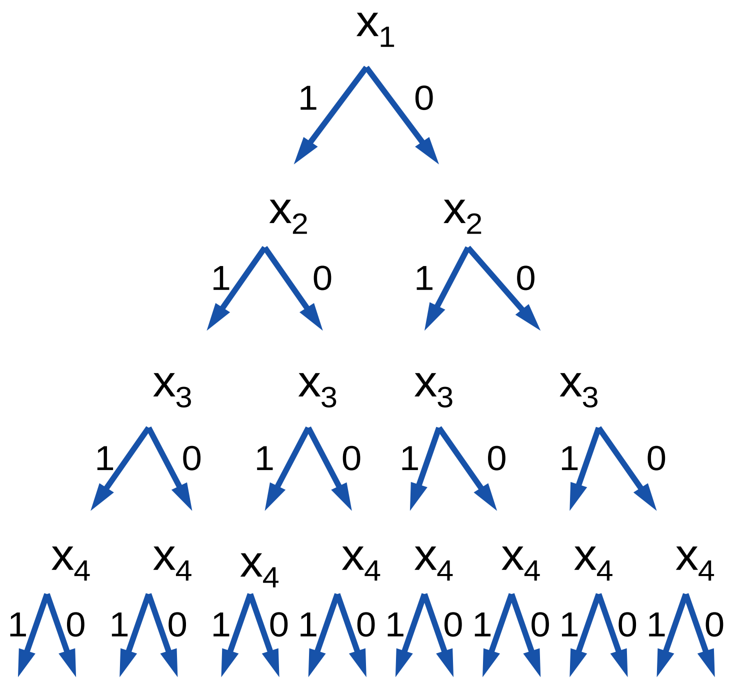
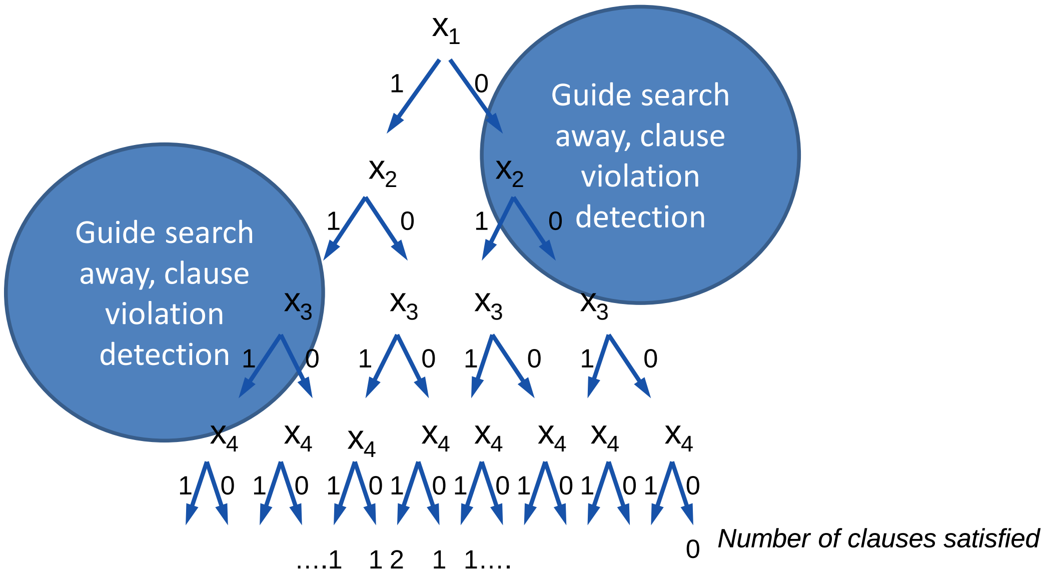


Figure 7a: Branch and Bound-Ready Decomposition of Four Variable Search Space $\{x_1, x_2, x_3, x_4\}$ for SAT Problem.



x_1
1
0
x_2
x_2
Guide search away, clause violation detection
Guide search away, clause violation detection
x_3
x_3
x_3
x_3
x_4
Number of clauses satisfied
....1 1 2 1 1....
0

Figure 7b: Branch and Bound Variable Depth Search of Search Space $\{x_1, x_2, x_3, x_4\}$ for SAT Problem (X1^~X2^X3)&(X1^~X2^~X4).

For the six-city TSP problem of Figure 14, a *BB*-suitable tree search can be built on the basis of *whether or not a path contains a particular segment connecting two cities*. The first node of the tree creates two 'buckets' of possible routes: one containing routes containing *AB* and one containing routes that do not. Subsequent nodes of the tree (there will be *N(N-1)/2* nodes for an *N* city tree in total) provide finer-grained partitioning of the space of possible paths. The key to reducing the time complexity of the search is a tight characterization of the best case performance that one can expect from *any given sub-tree*: Each bifurcation of the tree cuts the number of total search operations required by 50 per cent. Therefore, there is a premium on making estimates that trade off optimally between tightness and precocity of choice.

BB methods can be used to quickly narrow the search space in problems of strategy choice that have runaway computational complexity. In a simultaneous move oligopolistic competition game with 4 competitors, each of whom has 6 strategies at her disposal, the search space for combinations of strategies has 1296 distinct outcomes (6^4). A BB method can quickly narrow this space by 50% by eliminating combinations of strategies that include a 'low cost' product or service offering on the basis of examining the worst case scenario (a price war) that is likely to be triggered by this choice. Each step of eliminating combinations of strategies that contain an undesired component will trigger a similar contraction of the search space. BB is thus a form of 'elimination by aspects' of undesirable alternatives [Tversky, 1972] which relies on a structured partitioning of the search space, and quick calculation of the extremal values of the payoff landscape along different branches (i.e. based on a 'quick look-forward' by the problem solver).

How it differs from blind random search. A complexity-aware strategic management team using *BB* to sequentially narrow the space of possible solutions - 'options', 'strategies' - will produce periodic estimates of the merits of the various branches of the tree and iteratively eliminate the dominated branches. Its processes will differ from those used by a team that guesses blindly by (1) the structuration of the search space as a tree of mutually exclusive, collectively exhaustive (MECE) components of the possible solutions, and (b) the quick evaluation of families of solutions corresponding to certain roots of the tree, and (c) the quick elimination of dominated families of solutions. The performance of *BB* methods against many intractable problems suggests that *BB* approaches to strategic search should correlate with results that dominate blind, random guessing by producing a lower number of local optima ('competency traps') and a higher rate of convergence to the globally optimal solution.

Divide and Conquer (*DC*) methods relate to (a) partitioning the problem search space into smaller search spaces ('sub-problems') that can be more easily searched, and (b) piecing together the separate solutions to the smaller problems to form (possibly sub-optimal, but still superior) solutions to the larger, intractable, problem. For instance: he set of all possible subsets of a set of features of a product can be divided up into subsets-of-subsets of features that can be searched independently by several different individuals or teams working in parallel. *DC* methods offer no guarantee that the concatenation of solutions to smaller sub-problems will in general be an optimal or even feasible solution to the bigger problem: the concatenation of the minimal paths connecting two sets of *N/2* cities will not be the minimal path connecting the full set of *N* cities. Organizations tackling problems that have the algorithmic structure of *TSP* will generally not benefit from parallelization and decentralization of the search effort to the same degree that organizations which tackle problems which can be so decomposed - such as subset-search problems like COVER.

DC can be used to achieve both parallelization (*DC-P*) and/or randomization (*DC-R*) of the search process. *DC-P* can be used by a top management team to assign different sub-problems to different individuals or groups, based on their expertise, marginal incentives, such that individual team members or sub-groups can function as individual problem solvers in a way that enhances the functioning of the team as a whole as a group problem solver. *DC-R* can be used as an intelligent random search procedure, wherein the time complexity saving in the overall problem will be commensurate with the tightness of the estimate of the optimal performance solution that can be expected from each sub-group, but will also allow for the possibility that some bounds are either not tight or not correct.

How it differs from blind random search. Unlike a strategic team that guesses blindly, a complexity-aware strategic management team using *DC* search methods will engage in *early* partitioning of the solution search space and the parallelization of the solution search effort. It will also maintain tight coordination (and synchronization) between the parallel sub-space search efforts so as to minimize the probability of unnecessary search.

The outcomes of a DC-based search process will likely outperform blind guessing-based search by decreasing the probability of adopting local optima, because only *entire* solutions are considered as viable, and entire solutions can only be the result of concatenating all or most of the partial solutions produced by the parallel search groups. The decreased probability of adopting a locally optimal solution therefore arises from the fact that *DC* methods generate *fewer* local optimum 'trial solutions' than do alternative solution search methods.

Local Neighborhood Search (LNS). The dramatic reduction in the time complexity of TSP highlighted in Figure 12 above was accomplished by a procedure for searching the *N!*-size search space of the TSP using a local search meta-algorithm named after its inventors, Lin and Kernighan [Lin and Kernighan, 1973]. The procedure consists of selecting an 'almost-complete' tour of the cities, or a 'delta path', which includes all of the cities exactly once, except for the last one (e.g.: *1-3-2-5-6-3* for the six city *TSP* of Figure 6, for instance: 'almost complete' because the last link (return to *1*) is missing, and is replaced by a return to *3)*, measuring the total distance of the delta-path that had been generated, selectively making switches among the edges included in the delta path and edges that are 'available' but not included, comparing the total distance of the modified circuit with its last version, and maintaining the more efficient path. One key to the (exponential) speed-up achieved by this heuristic is the strategy by which edges are exchanged, which is (usually) *2* (specifically: *1-3&3-2 replaced with 1-2&2-3*) at a time (entailing a local search space of $N(N-3)/2$. The algorithm allows for the exclusion 'by inspection' of many possible combinations of routes: for instance, in the 4663 city TSP (Figure 12), one can exclude combinations such as (*Toronto (Central Canada)-Kelowna (Western Canada)-London (Central Canada*) without 'evaluation'. The speedup produced by *LS* is related to the replacement of the *N!* search space by a tightly coordinated sequence of moves along a trajectory of $O(N^2)$ search problems, and is typical of both the architecture and the performance of well-designed local search strategies generally.

How LNS differs from blind random search. One would expect strategic managers engaging in *LNS* to (a) use intuition and experience to guide the starting point(s) of the search to a plausible 'complete' solution (a full contingent plan found in 'scenario planning'), (b) generate small deviations from the initial solution by altering individual components thereof, and (c) tightly coordinate the process by which they evaluate the performance of the set of 'perturbed' solutions to keep track of iterative improvements over the initial solution. Other than the exclusion of dominated solutions, search moves within the local neighborhood of a local solution are 'random'. A strategic management team, for instance, may 'search' in the neighborhood of a capacity expansion strategy in an undifferentiated oligopoly by looking for combinations of product features and marketing and distribution tactics that will sustain margins in the eventuality of a possible p[rice war its expansion will trigger. LNS can provide an algorithmically sound explanation for states of 'frenetic local search' within a strategic management team, which, although seemingly 'unintelligent' and 'prematurely anchored' on an initial guess can in fact produce dramatic reductions in the costs of producing solutions to *NP* hard problems. *LNS* methods will likely generate local optima that are more tightly clustered around a point on the firm's fitness landscape corresponding to the starting point of the search, and a higher rate and probability of convergence to the global optimum of the strategic solution search space.

Randomization Methods (RM): Stochastic Hill Climbing. Randomized algorithms [Hromkovic, 2003] have achieved great levels of sophistication and success in the field of algorithmics, with good reason: *NP* stands for Non-*Deterministic* Polynomial Time (rather than *Non*-Polynomial Time). It points to the fact that intractable problems *are* solvable by non-deterministic algorithms (or, 'Turing Machines') in polynomial time. It is not, then, surprising that intelligent randomization yields significant improvements in complex problem solving performance. A *randomized algorithm* (as opposed to a process of 'blindly guessing') is a *structured and informed guessing* strategy, aimed at maximizing the probability of arriving at a good enough solution in a given number of steps. The difficulty of searching the solution spaces of most 'hard problems' is the vast number of 'local minima' that incremental procedures ('local hill climbing', or incremental improvement strategies) can get trapped into [Rivkin, 2000; Levinthal and Ghemawat, 1999]. Stochastic hill climbing methods (*SHC*) [Michalewicz and Fogel, 2004] ease limited search processes from the constraints of local optima by probabilistically causing the searcher to 'jump' to different regions of the search space and thus to perform large numbers of bounded local searches. *Simulated annealing algorithms* generalize stochastic hill climbing methods by specifying various *temperature level gradients* of the search process: temperature denotes 'mean kinetic energy' of a state of particles assembled in a macro-state (e.g. liquid), and the *temperature of a search process* increases with the probability that the process will jump away from the local search it is currently performing within a certain time window. Strategic managers using stochastic hill climbing-type solution procedures can 'heat' or 'cool' their adaptive search process depending on the problem solver's estimate of the number of local optima entailed by the problem statement or, adaptively, as a function of the results of the ongoing set of local searches. Adaptive search procedures of this sort will exhibit jumps from one region of the search space to another whose frequency and probability of occurrence depends on the results that immediately precedent sub-space searches have produced.

Intelligent randomization provides one explanation of how organizations solve problems that map into technically intractable canonical problem statements by arriving at good enough solutions in a number of steps that does not overwhelm the ability of the organization to calculate and execute. A well known model of organizational structure, dynamics and performance represents the organization as a set of N elementary activities that are in either 'on' (1) or 'off' (0) states. The performance of the organization (its 'fitness function') is a real valued function of all of the possible sets of activities that the organization jointly engages in. Subsets of activities are more or less tightly coupled, and the degree of inter-dependence of these activities is represented by a natural number k, representing the number of activities to which each activity is coupled. The resulting *NK* model has been used repeatedly [Levinthal and Warglien, 1997; McKelvey, 1999; Rivkin, 2000; Lenox, Rockart and Lewin, 2006] to examine the dependence of a firm's performance on the structure and topology of its activity sets. To a firm that aims to strategically choose the set of activities that it pursues, the problem of choosing the set of activities that optimizes its performance (the real valued fitness function) has been shown [Weinberger, 1991] to be computationally equivalent to the well-known intractable (*NP*-complete) *kSAT* problem for $k>1$. That problem takes as an input a set of N variables and a set of M Boolean expressions containing up to k variables AND, OR, and NOT, and asks for an assignment of the N variables to the M expressions that will make these expressions true (i.e. will 'satisfy' them, hence the name of the problem. The *Nk* strategic decision problem ('Is there a fitness function of the N activities, each mutually coupled to k others with value greater than V?') maps into the *kSAT* problem ('Is there a set of variables whose aggregate satisfiability score is at least N when plugged into a set of M k-variable formulas?') trivially for $M=N$, and with padding of the search space for $M>N$ and $M<N$ [Weinberger, 1996]. Based on this result, Rivkin [2000] argued that the intractability of the *Nk* decision problem (derived from the intractability of the *kSAT* problem for $k>1$) can make complex strategies (characterized by the patterning of many, densely linked activities) difficult to imitate.

However, the complexity of solving the *kSAT* problem yields to searches of the solution space that are based on randomly permuting both the sets of initial variables and the assignment of truth values to variables within a formula [Schoening, 2002; Brueggermann and Kern, 2004; Ghosh and Misra, 2009]. These approaches achieve a worst-case complexity of solving the 3SAT problem of $(1.d)^N$ (where d is a natural number following the decimal point*)* instead of *2^N*, which, even for very large values of *N* can produce a massive decrease in the complexity of the *3SAT* problem. See Table 3: a factor of 10^{20} reduction in the complexity of solving the 3SAT problem for *N=100* is achieved by a random walk based algorithm, transforming a problem that it unsolvable on any computational device or combination thereof to one that is trivially solvable on most computational devices today, and suggesting that a realistic Nk problem (with *N=100, k=3*) can be solved by an organization (or a strategic manager) that pursues a structured randomization strategy.

N	**Exhaustive Search Complexity 2^N, Total Number of Operations.**	**Random Walk Based Search Complexity 1.334^N, Total Number of Operations.**
1	*2*	*1.3334*
2	*4*	*1.7796*
3	*8*	*2.3707*
4	*16*	*3.1611*
10	*1,048*	*17.7666*
20	*1,048,576*	*315.6523*
100	*1.27×10^{30}*	*3.16×10^9*

Table 3: Comparison of Computational Complexity of Solving 3SAT Problem Using Deterministic Exhaustive Search (Column 2) Versus a Set of Rapid Random Walks (Column 3) As a Function of the Number of Variables (Column 1).

How SHC differs from blind random search. SHC-guided search for strategic solutions embodies one of the key advantages that intelligent, complexity-aware randomization has over blind randomization: it forces the solver to abandon local optima as part of the search protocol. Processes guided by *SHC* will likely exhibit the quick and possibly parallel exploration of multiple candidate solutions, and rapid jumps of the problem solving process from one candidate solution to another. The search protocol is less likely to converge to a dominated local optimum than an alternative based on blind guessing because it embodies stopping rules for searching around local optima; and will exhibit higher probabilities of converging to a global optimum, because of the lesser likelihood of getting trapped in local optima, and the greater coverage of the full solution search space.

Genetic (or, evolutionary) algorithms (GA) combine the insight that randomization can induce (probabilistically speaking) speedup of search with a structured approach to the solution generation process inspired from evolutionary biology [Baeck, 1996]. Primitive, random candidate solutions or 'partial solutions' (which could, for instance, be delta paths in a Lin Kernighan representation of TSP) are perturbed ('mutation') and combined ('sexual reproduction') to produce new candidate solutions that are then selected on the basis of the quality of the solution that they encode [Fogel, 1995]. Mutation rate, selection pressure and recombination intensity are all parameters under the control of the problem solver. Exponential speedup of convergence to the shortest route has been reported for the TSP [Wang, Zhang and Li, 2007], which arises both from the parallelization of the randomization operator across members of a population of candidate solutions ('mutation', 'recombination') and from the iterative application of the selection operator which operates at the level of the entire population.

How GA differs from Blind Random Search. Strategic solution processes patterned by *GA* type procedures will differ from blind random search with regard to the process by which candidate solutions are generated: fast generation of many bits and pieces of a solution ('chromosomes') followed by recombination of the candidate solutions and the concatenation of the randomly generated and recombined solution segments into candidate solutions will be followed by the rapid elimination of clearly inferior solutions. At the level of outcomes, *GA* methods provide fast coverage of the solution search space, and, accordingly, lower probability of choosing a dominated solution (a local optimum); along with faster convergence to the overall global optimum. Unlike the case of *SHC,* however, where local optima are filtered away because the search process is more likely to *escape* from them, or the case of *LNS*, where local optima are filtered out because the search process is less likely to generate them, in the case of *GA*, the reduction of incidences of falling into local optima is due to the fact that local optima are less likely to be chosen as candidate solution, because of the inherent competition among local optima that the approach generates.

Relaxation to Linear Optimization. (*RLO*) Informed guessing (and updating of the solution search space accordingly) is not the only approach that organizations can take to solving the problems arising from an *N-k* model of their own activity sets. They can also *re-frame* the decision problem into an optimization problem that is approximable by algorithms that have well behaved complexity bounds. In other words, given a problem they cannot solve, they can re-formulate as a closely related problem that they *can* solve. A strategic manager can re-frame the *Nk* decision problem from that of deciding whether or not the configuration of activities the firm is or could be pursuing is maximal, to that of maximizing the total 'fitness score' of the set of activities it is or could be pursuing. The resulting problem maps into the well known *MAX SAT* optimization problem: 'given a set of k-variable clauses and a set of *N* variables that figure into these clauses, maximize the number of satisfies clauses over all of the possible ways of assigning variables to clauses.' Asano and Williamson [2002] show that an approximation algorithm which takes as an input a linear-programming equivalent of the integer programming problem *MAX 3SAT* can achieve performance guarantees of close to 0.85 of the global maximum in polynomial-time (i.e. in the 'tractable' regime). Mapping back to our *N-k* optimization problem: if the organization (a) re-frames the *N-k* decision problem into an *N-k* optimization problem (finding the optimal configuration of activities, as opposed to determining whether or not there exists an activity set that will generate a pre-determined utopia point – which may in any case be more realistic), (b) approximates the problem statement by an associated linear programming problem (allowing for performance figures of clauses to take on values between 0 and 1, and not just ether 0 or 1), (c) solves the resulting (tractable) problem and then (d) adjusts the resulting solution to reflect the constraints imposed by its real activity sets, then it can achieve – by only solving a tractable problem – performance results that are guaranteed to be within 15 per cent (1-0.85) of the global optimum. Randomization and approximation can render intractable problems into practically solvable ones, provided that (a) randomization is pursued intelligently and in a way that informed by the structure of the problem, and (b) that the organization can adjust its goals from achieving the global maximum with no margin for error to achieving a global maximum within some tolerance. This is not an argument that complexity *cannot* inhibit imitation or function as a source of competitive advantage, but rather that it *need not* do so, and a computationally savvy imitator can break through the intractability barrier of optimizing its activity sets to match or exceed the performance of a competitor. It also suggests that there may exist a special set of 'imitative capabilities' – grounded in randomization and approximation procedures – that allow perennial 'second movers' to become competent imitators.

The Knapsack Problem (*KSP*) [Karp, 1972] (P: "Find the set of utensils of known value and volume that can be packed into a knapsack of fixed volume such that the total use value of the utensils in the knapsack is maximized") can be used to model a large number of problems of product, solution or service design (P: "find the maximum value set of features of known value and cost that are embodied in a product or service such that the total cost does not exceed C") or the problem of the optimal design of a value-linked activity set (P="find the maximum net value combination of value-linked activities of known value and cost"). The difficulty of *KSP* is known to lie in the fact that it is an integer programming *(IP)* problem, which arises from the *lumpiness* of the objects in the problem statement: no utensil can be sub-divided for inclusion in the knapsack, and any utensil must either be taken or left behind. Were utensils (product features) sub-divisible or only probabilistically inclusible in the knapsack, then the *IP* problem would 'relax' to a linear programming (*LP*) problem of maximization of *N* variables (under *1* constraint: volume), a problem of polynomial time complexity. A simplification procedure for *KSP*, accordingly, comprises (a) *relaxing* the constraint that all numbers of utensils might be integers, (b) solving the associated *LP* problem, (c) *rounding* the resulting solution to the nearest set of integers, (d) verifying that the solution is feasible and (e) repeating (c) (with a different rounding criterion) and (d) until a solution is found. *RL* is applicable to a large number of intractable problems, and Hromkovic [2003] gives *LP* relaxations of other *NP* hard problems. Strategic managers and strategic management teams can implement *RL* by *cutting corners* on an associated *IP* problem: By loosening the 'lumpy' constraints that are 'written into' the problem statement via the "'0 or 1' rule" by either *negotiating* or *cheating* strategies. Negotiating (with large clients, for instance) the *precise definition* of technical terms such as '*99.999* per cent reliability' (a common requirement in telecommunications firms) can turn a hard requirement into a soft one (one that can be satisfied with a system that is "*99.999* per cent reliable in *these* conditions). Alternatively, one can engage in a form of *cheating* (by 'marke-tecting' rather than 'architecting' a product, service or platform) in a way that *promises* satisfaction of all requirements and constraints but does not deliver on promises that are *ex ante* believed to be difficult to audit by the market. *Distortive* maneuvers undertaken by firms faced with *IP* (*NP* hard) problems can, in this case, be seen as adaptive relaxations of the *NP* hard problem of product, platform or activity set design to a *P*-hard *LP* problem of constrained linear optimization with variables ranging over the real numbers.

How RLP differs from blind random search. RLP spells out a set of approximation rather than randomization search methods. As such, the problem solving process patterned by *RLP*-type algorithms will likely exhibit a significant focus on the precise definition of the problem, aimed at iteratively and adaptively *relaxing* the constraints that render the problem intractable (integer constraints, for instance). The quality of the performance outcome will depend on the degree to which the global optima to the relaxed problem statement track those of the intractable problem. 'Better' relaxations of the constraints of integer programming problems will provide tighter approximations of the global optima of the original problem statement. Because the resulting optimization problem is tractable, the probability of getting trapped into local optima is lower, and the rate of convergence to the global optimum will be significantly higher than those achieved by alternative methods.

Implications for the Definition and Measurement of Strategic Problem Solving Capabilities. The range of intractable problem solving techniques introduced allow us to refine our understanding of strategic problem solving by incorporating the dependence of 'complexity' and 'decomposability' on the structure of a problem and the properties of the solution search method:

Separability and Decomposability Are Contingent on Both Problem Statements and Solution Procedures. The use of Branch and Bound methods to solve the TSP suggests that *separability* (or, decomposability [Simon, 1996]) of a problem depends *both* on structural elements of the problem itself (in this case the TSP) *and* on the approximation method used (in this case, Branch and Bound). While the TSP is *not* decomposable using 'divide and conquer' methods (the shortest path connecting N cities will in general *not* be the concatenation of the shortest paths connecting k disjoint subsets of N/k cities), it *is* decomposable via Branch and Bound (or, for that matter, via genetic algorithms that operate on orthogonal sets of cities).

There Are Many Ways to Decompose a Problem Into Sub-Problems. Solving large scale optimization problems by fractionation into sub-problems bring up the well-known *credit assignment* problem [Minsky, 1961; Denrell, Fang and Levinthal, 2004] which requires the problem solver to assign 'partial credit' to the outcomes of the individual steps (or, sub-problem solutions) of a many-step sequence of goal-directed actions after learning the outcome of the entire process. Denrell et al [2007] posit that the *local mental models* of the problem solver will influence credit assignment by constraining the local optimization problem that each agent solves at each step of the process. Through the algorithmic perspective, a *global model* for an intractable problem will comprise *both* the problem statement and the *procedure* for breaking up the problem into sub-problems, which is supplied by the search heuristic used (Divide and Conquer, Branch and Bound, Stochastic Hill Climbing, Local Neighborhood Search, Genetic Algorithms), which will then supply different *local* models for the sub-problems to be solved. An *LNS* based search of the TSP search space will produce a very different set of sub-problems (and local mental models) than will a BB-based partitioning of the TSP search space. In the first case, the *initial guess* supplies a very large number of hard constraints on the local search. In the latter case, the initial partitioning supplies a low number of constraints of (a much simpler) local search.

5. Implications for the Definition of Adaptive Advantage and Dynamic search Capabilities.

The family of algorithmic and quasi-algorithmic procedures for strategic problem solving introduced here allow us to characterize the advantage relating to the firm's ability to adapt in real time to changes in the landscape of its strategic payoffs. It is a form of exploratory advantage [March, 1991] that accrues to a firm in virtue of its dynamic search capabilities – its ability to deploy its problem solving techniques, routines and procedures to solve the problem of optimally mapping organizational actions to predicted payoffs. These can be understood as a specific set of dynamic capabilities [Teece and Pisano, 1994; Eisenhardt and Martin, 2000] that allow the firm to search the solution spaces of its strategic problems more effectively, efficiently and reliably. The algorithmic framework allows us to operationalize dynamic search capabilities as the strategic ingenuity of a firm by considering the net benefit of using a problem solving procedure (an algorithm A, which may be deterministic or probabilistic) to a problem (tractable or intractable) to generate a solution S_A. The time complexity of the algorithm gives us an estimate of the total cost of using it to generate S_A, i.e. $C(A)= c(m)\ K(S_A)$, where $c(m)$ is the marginal cost of an operation and $K(S_A)$ is the time complexity (number of operations) of producing the solution using A. The value of the solution S_A generated by algorithm A, $V(S_A)$, will be different for different kinds of algorithms, as follows:

Deterministic algorithms, exact solutions. $V(S_A)=V(S)$ if the algorithm has converged (i.e. if $m=K(S_A)$), and 0 otherwise. Since the solution generated by the algorithm is the solution to the problem, the value of the solution generated by the algorithm at any step m is the value of having solved the problem exactly. Examples include small competitive games (small numbers of players and strategies) and low-dimensional induction, abduction or deduction tasks;

Deterministic algorithms, approximate solutions. $V(S_A)=F_m(||S-S_A||)$, where $dF/d()<0$: the value of the solution generated by the algorithm at any step m will be a decreasing function of the distance between the solution produced by the algorithm (the approximation) on the mth iteration and the exact solution to the problem. Examples include approximate solutions to IP problems using LP relaxation methods;

Randomized algorithms, exact solutions. $V(S_A)=G(Pr_m(S=S_A))$: the value of S_A is proportional to the probability that the solution generated on the mth operation of the algorithm is the solution to the problem. Examples include stochastic hill climbing algorithms and genetic algorithms;

Randomized algorithms, approximate solutions. $V(S_A)=G(Pr_m(||S=S_A||<€)$, where $dG/d()>0$: the value of the solution produced by the algorithm at step m will be an increasing function of the probability that the solution produced at step m is within a small enough distance of the solution to the problem. Examples include stochastic hill climbing, genetic algorithms as well as randomized versions of Branch and Bound and Divide and Conquer meta-algorithms.

We can now articulate a measure of the *adaptive advantage* of a firm over another that is general enough to encompass all firm and industry level predicaments of interest (via the canonical representation of 'problems'); specific enough to make predictions of relative performance in individual cases; and adaptive to the time constraints that firms constantly face when they solve problems. If firms taken in their entirety produce solutions to business problems and if their strategy making processes can be understood as the application of the collection of problem solving tools and methodologies at their disposal, then their strategic search advantage should track the quality of the solutions they provide.

To define the quality of a solution-generating process, we focus on *three* dimensions (the accuracy, reliability and efficiency of the problem solving process), and accordingly on the following trade-offs: between the accuracy of a solution and the speed with which it can be attained (which will depend on the time complexity of the problem solving procedure and the marginal cost of individual operations), between the accuracy of a solution and the probability of attaining it in a given time window (which measures the reliability of the algorithm or meta-algorithm used), and between the reliability of a solution and the speed with which it is generated (Figure 8). These frontiers can be pieced together in a three-dimensional measure of time-bounded competitive advantage (Figure 9), which induces a simple 'ordering' among firms relative to a particular problem (or, collection of problems): "Firm *A* has a strategic search (adaptive) advantage over Firm *B* at solving problem *PROB iff* it can produce a more accurate solution to *P* more quickly and more reliably than can firm *B*."

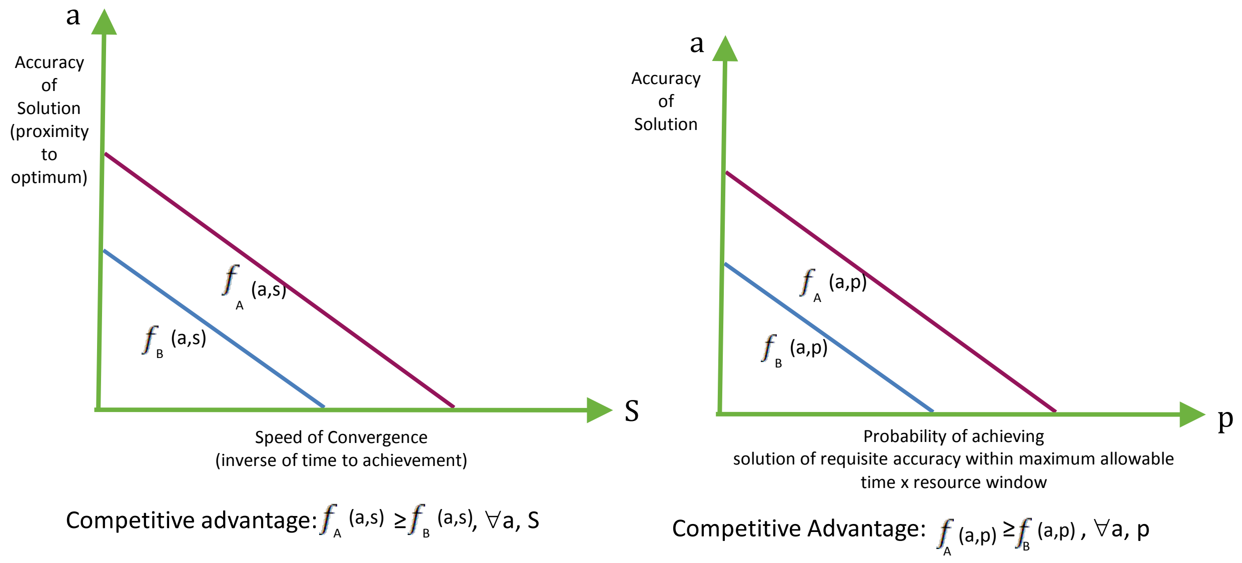
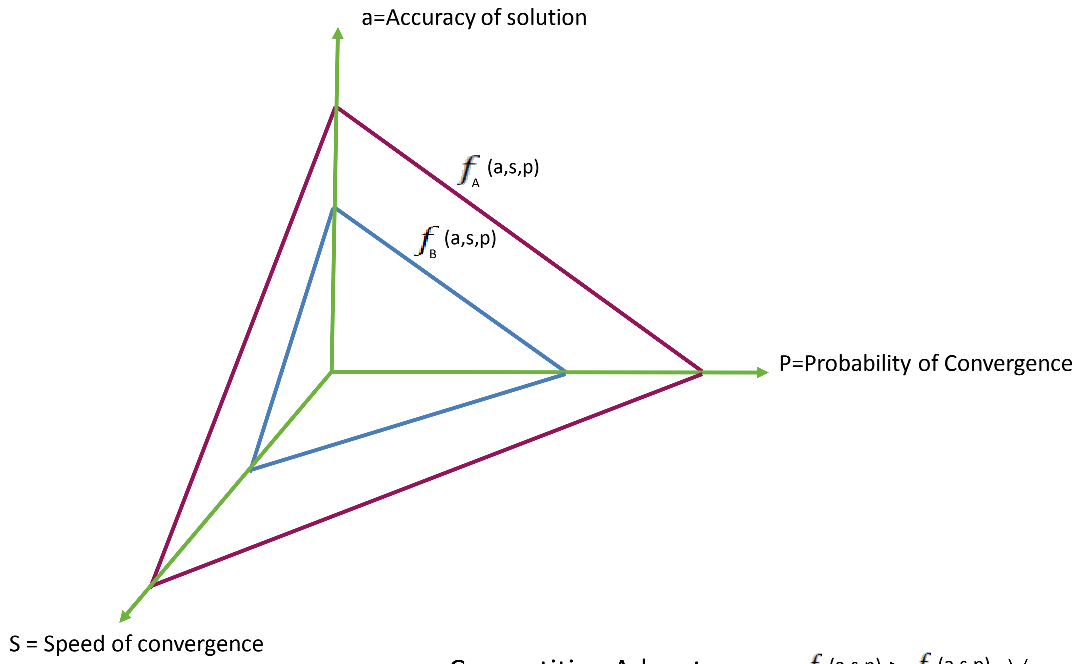

a
Accuracy of Solution (proximity to optimum)
f_A (a,s)
f_B (a,s)
S
Speed of Convergence (inverse of time to achievement)
Competitive advantage: f_A (a,s) ≥ f_B (a,s), ∀a, S
a
Accuracy of Solution
f_A (a,p)
f_B (a,p)
p
Probability of achieving solution of requisite accuracy within maximum allowable time x resource window
Competitive Advantage: f_A (a,p) ≥ f_B (a,p) , ∀a, p

Figure 8: Illustrating the Concept of "Adaptive Advantage of Firm A over Firm B" via Trade-off Frontier Between Accuracy (Validity, Goodness of Fit) and Convergence (Speed of Solutions Offered by the Two Firms), and the Concept of "Competitive Advantage of Firm A over Firm B" via Trade-off Frontier Between Accuracy and Reliability of Solutions.



Competitive Advantage: $f_A(a,s,p) \geq f_B(a,s,p), \forall a,s,p$

Figure 9: Three-Dimensional Measure of 'Adaptive Advantage'.

Functional and Structural Adaptive Advantage Defined. The adaptive search advantage of a firm can take both functional (the procedures it uses) and structural (the organizational architecture it uses) forms. Strategic managers may choose both among different solution procedures for a complex problem, and among different organizational structures for implementing the search process [Rivkin and Siggelkow, 2003]. The computational approach to strategic problem solving offers a way of parsing the types of strategic search advantage that a firm enjoys in distinct categories, relating to both structural and functional forms of adaptation:

Off-line versus on-line problem solving. Strategic managers can attempt to solve strategic problems 'off-line' – in their minds, or in sessions of strategic deliberation and planning that are uncoupled from the operations of the firm. But, a firm as a whole can be modeled as searching the solution space of the problem it is trying to solve, thus engaging in 'on line' problem solving, and this embodied search process does not require the conscious representation of a problem, its search space, or the search of optimal procedures for solving it. A large logistics and distribution business, for instance, is, as a whole, 'looking for' the optimum (minimal cost) configuration of flows of goods from one location to another over a network of possible (cost-weighted) paths, which can be mapped into an optimal network flow problem that is *NP*-hard [Hromkovic, 2003]. The owner of a small shoe shop may have no conception of linear programming (*LP)*, and still less of using *LP* methods to optimize her inventory subject to estimated local demand for shoes, but the shop may nevertheless *produce* inventory numbers that *correspond* to the outputs of an *LP* procedure. Strategic managers in a mining firm may have no clue about randomized search procedures for very large spaces, but the mining firm as a whole may embody a set of randomized search procedures for the optimal drilling plans in a region suspected of harboring a deep gold vein [Moldoveanu and Martin, 2009]. That intelligence does not require representation [Brooks, 1991] is a point made in the artificial intelligence literature more than 20 years ago, and is relevant here as well. The intelligent search for solutions and design of search procedures may be something that individuals in firms engaged in solving intractable problems do without knowing (in the sense of being able to articulate) *that* they do and *how* they do it.

Structural versus functional adaptations. The literature examining the links between complexity and the range of complexity-adaptive strategic actions [Siggelkow and Levinthal, 2005; Rivkin and Siggelkow, 2003] usually take the topology of the firm's value-linked activity system and the assignment of tasks to individuals as fixed or to a large extent determined by the production function of the firm. However, as we have seen in the previous discussion of the design of search procedures for the solution to complex problems, different algorithms and heuristics can make use of parallelization, randomization and brute force methods, each of which entail different assignments of tasks, decision rights and incentives to groups and individuals. The computational framework allows us to distinguish between *structural* adaptations – relating to the assignment of incentives and decision rights to individuals and teams, and *functional* adaptations – relating to the design of the specific tasks carried out by different individuals' minds and bodies. Structural and functional adaptations can shape both online and off-line problem solving, as follows:

Offline, functional. This is the classical image of strategic problem solving taking place in the minds of strategic managers, or within the top management team: problem statements are formulated, and optimal solution search procedures are designed and simulated. Adaptive advantage accrues to the strategic managers and teams that design procedures that yield the most accurate and reliable results over the shortest period of time.

Online, structural. This is a less familiar image of the organization as a whole engaging in a problem solving process and searching the solution space through a large sequence of adjustments in decision rights, expertise/talent and incentives. Like a colony of ants working together seamlessly to build a raft that transports the colonies' offspring across a rivulet after a flood, there is no complete representation of the problem or the solution search space in the mind of any individual within the organization. Strategic search advantage derives from the speed with which the organization as a whole can explore a rugged payoff landscape and avoid traps arising from local optima and catastrophes arising from coordination failure, which in turn arises from the fit between the solution space search procedure the organization is executing and the type and complexity of problem it is solving;

Offline, structural. These are adaptations of the 'offline' processes of planning, deliberation and strategic problem solving of a strategic management team to the nature of the problem being solved. These processes remain offline because they are not yet implemented by the organization, but the problem-solving culture of the management team – the set of communication, coordination and search heuristics that the team uses, and the fit between these procedures and the complexity and type of problems being solved – all shape the speed, reliability and accuracy with which solutions or optimal solution search procedures are actualized;

Online, functional. These are adaptations of the tasks of individuals and groups within the firm to the specific search problem the organization is solving. Different solution search procedures for canonically hard problems – like *DC, BB, SHC, LNS* – entail different sets of tasks (narrow exploration around local optima or in the neighborhood of a plausible best first guess, wide jumps among exploratory regimes, etc) which can be implemented by the same individuals or teams. Mapping tasks to teams and groups is constrained but not determined by the expertise, decision rights and incentives that have been provisioned throughout the organization. Therefore, a further degree of freedom for the design of optimal strategic search processes relates to the adaptive assignment and reassignment of tasks to people (or, machines) in order to maximize the firm's ability to synthesize more reliable and more accurate solutions in a shorter period of time.

6. Implications of the algorithmic approach for the empirical study of strategic process, outcome and performance. The computational approach to strategic problems has significant implications for the empirical study of the 'strategic management of complexity'. Some researchers have focused on the fit between firm architecture (centralized/decentralized; sparsely/densely connected) and environmental complexity (uncertainty, ambiguity, rate of change of relevant variables) [Siggelkow and Rivkin, 2005; Davis, Eisenhardt and Bingham, 2008]. In parallel, others have focused on the 'dynamic capabilities' [Teece and Pisano, 1994; Arend and Bromiley, 2011] that may explain why some firms succeed in quickly-changing environments when others fail. The computational view of strategic problems and their on line and off line solution procedures clarifies the interplay between problem structure and complexity, organizational heuristics and procedures and outcome and the importance of the problems firms solve and solution search procedures that they search with in shaping the fit between the architecture of the firm and the structure of the environment.

Firms develop core capabilities for solving particular kinds of problems and claiming value from the resulting solutions. A large systems integrator like Cisco or Alcatel Lucent will develop specialized algorithms and meta-algorithms for solving large scale logical compatibility problems like *COVER,* that a logistics company like CEVA or Kuehne and Nagel will evolve specialized routines and sub-routines for quickly solving *TSP*-type problems; and that a software solutions company like SAP, Inc. will evolve better, faster, more reliable heuristics and meta-algorithms for solving multi-feature product design problems captured by *kSAT* and *COVER*. The structure and complexity of these problems, along with the solution procedures and organizational architectures used to implement them and the metrics used to gauge the quality of a solution, together determine firm level outcomes. The computational approach offers a *contingent* mapping among different strategic problems, solution procedures and the organizational structures by which solutions are generated, selected and implemented. With such a mapping, we can examine the relationship between changes in environmental conditions, changes in the complexity class and structure of strategic problems, and changes in problem solving procedures and organizational outcomes. We can also use the approach to figure out whether Kuehne and Nagel's logistical optimization procedures are optimized to its core problems, or, rather, there is significant room for improvement - potentially by an ingenious innovator. The computational and algorithmic study of strategic problem solving outlined here implies a *three-fold contingency* in the factors that influence outcomes and performance, which can be summarized in terms of fit (Figure 10):

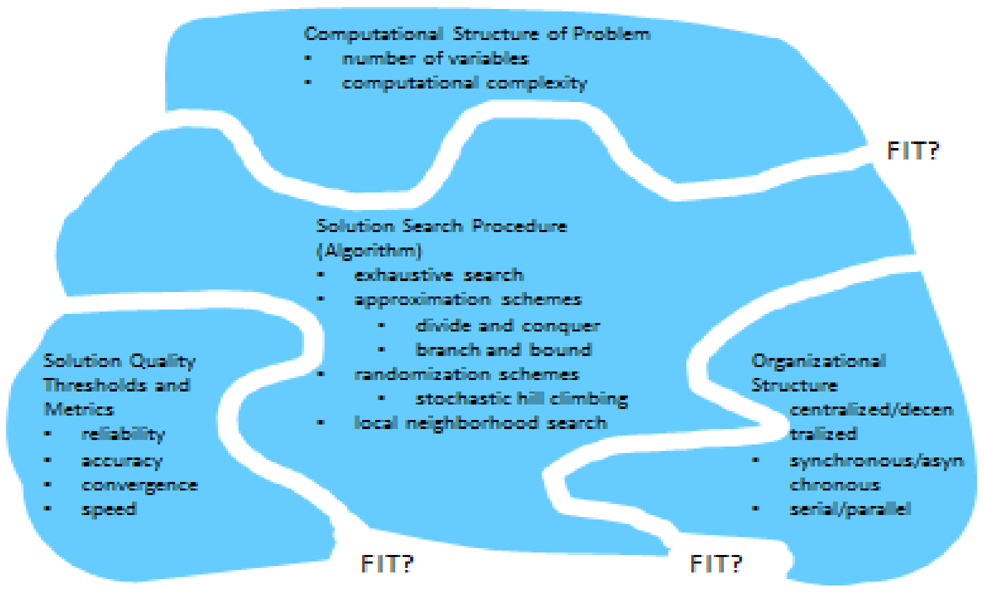


Figure 10: Four Fold Contingency of Outcome and Performance From an Algorithmic Perspective

Between the complexity class of the strategic problem and the strategic solution search procedures ('algorithms') of the firm. At the level of the canonical problems and solution search procedures, outcome depends on the degree to which the solution algorithm is adapted to the complexity and the computational structure of the problem. An exhaustive search of all the possible solutions to an *NP*-hard problem with a large number of input variables, for instance, is likely to exhaust the time and resources of the firm before a good enough solution is found. On the other hand, the discipline of exhaustive search through all alternative solution for a *P*-hard problem will provide highly reliable method by which the optimal set of actions is attained. Not only the computational complexity class and the number of variables matter to the selection of the best solution search procedure. Some approximation procedures – like divide-and-conquer – are well suited to some intractable problems – like the Knapsack Problem – but very poorly suited to other problems – like the Traveling Salesman Problem, which is amenable to a Local Neighborhood Search procedure. The first level of contingency of outcome on strategic problem solving process, then, has to do with the degree to which the solution algorithm matches the complexity class and the computational structure of a strategic problem. *Adaptive rationality* in this case acquires the meaning of the ability to change the solution procedure in response to the nature of the problem.

Between the strategic search procedures of the firm and its organizational structure and architecture. A second level of contingency arises at the level of fit between the problem solving procedure and the organizational structure or architecture that implements the solution process. A randomization procedure like stochastic hill climbing or an approximate search procedure like divide and conquer are optimally implemented on distributed organizational architectures that can work in parallel while at the same time keeping each other abreast of progress made along the various paths and branches associated with the smaller sub-problems. Exhaustive search, by contrast, can benefit from a fast, serial implementation of the problem solving procedure, which allows for instantaneous pooling of the solutions generated and evaluated along the way. Equally important to the organizational architecture of a problem solving process is the communication protocol associated with a distributed problem solving task. Distributed problem solving may require more or less synchronous sharing of valid and relevant information regarding the solution search space, depending on the kind of procedure used: while branch and bound approaches may be implemented using asynchronous communication protocols once the division rule for various parts of the solution space has been agreed to, stochastic hill climbing and genetic algorithm-based approaches will proceed far more quickly on the basis of a synchronous information sharing protocol, which keeps the various problem solving teams abreast of each others' progress.

Between solution search procedures of the firm and the metrics used to gauge the quality of the intermediate and final solutions. A third level of contingency arises at the level of the fit between the solution search procedures used by a firm and the metrics and thresholds for gauging the quality of the solution before, during and after the problem solving process. *Ex ante*, the specification of thresholds for adequate or good enough solutions will impact the usability of some algorithms as problem solving procedures. Genetic algorithms, for instance, may generate large numbers of inferior or dominated solutions before arriving at a global optimum. But, if these dominated solutions are excluded by the problem solving team because of the inadequate performance, then the algorithm will not converge to the optimum, precisely because the requisite diversity among partial solutions required for the performance of the procedure as a whole will have been curtailed. On the other hand, using clear metrics *ex ante* for the elimination of inferior solutions helps distributed problem solving teams with the use of branch-and-bound type methods, which rely for their success on the quick elimination of inferior solutions. 'Strategic patience' in problem solving, therefore, yields different payoffs for different procedures. Similarly, along-the way measures of the quality of solutions that emerge during the problem solving process can be more or less suitable to the procedures used. If a strategic management team expects linear or monotonic convergence of the problem solving process to an optimal solution – wherein each new iteration yields a better solution than the last – then this measure will essentially preclude the use of some non-linear or non-monotonic procedures like genetic algorithms and stochastic hill climbing, but even local neighborhood search that can generate multiple inferior or dominated solutions 'on the way to' the optimum solution.

Implications for the Imitability, Replicability and Transferability of Adaptive Advantage in On-line and Off-line Strategic Problem Solving. Understanding firm-level strategic outcome as the result of a three-fold fit among complexity class and structure of problems, the stock of solution procedures and the organizational structures and architectures used to implement solutions allows us to investigate the relationship between complexity, imitability and replicability [Rivkin, 2000; 2001; Szulansky and Winter, 2001] in a new and more precise way. The imitability of a focal firm's solutions is limited by the complexity of the problems provided that (a) the firm has access to a stock of solution procedures which produce more reliable and accurate solutions more quickly than its would-be imitators and/or (b) that the firm has achieved a fit between these solution procedures and its organizational architecture that is costly to imitate. Replication has traditionally been held to be an 'easy' problem relative to imitation, because the firm owns a 'template' for its own past success [Nelson and Winter, 1982; Rivkin, 2001].

However, replication of one business strategy in a new context varies in difficulty, according to how novel the new context is. At one end of the spectrum, a firm can replicate an algorithmically simple set of strategies in a market that has not changed much from its state at the time of the firm's previous success (same product feature set, same customers, same technology, same development constraints, same competitive landscape) - which would also make the strategy easier to imitate. At the other end of the spectrum, the firm can aim to replicate its strategy in a market in which (a) the computational structure and complexity of the problem has changed (because of technology changes, novel network effects at the level of competitors or customers, or a new set of competitors), or (b) the number of relevant variables has increased (more competitors, more relevant predictive variables in demand function, more technological options) even as the computational structure has remained constant (turning the problem into an intractable one), which likely entails that (c) the firm's stock of solution procedures needs to be modified to optimally adapt to the new problem structure and (d) that the firm's structure and architecture may also need to change to track the new solution procedures. The problem of replication, therefore, may be more usefully thought of as a problem of *transfer* of the firm's problem solving procedures and architectures to new domains of activity, which may be similar to the domains of the firm's original success (the domain of what is now called replication), or different in various degrees and aspects (new problem, new optimal algorithm, new optimal structure) therefrom. The 'new-ness' of the firm's strategic problem is, a function of a set of variables (complexity of problems, set of search procedures and adaptive organizational architectures) which the new modeling language allows us to explicitly model and parametrize.

Firms can and do survive shifts in their core market focus, which often bring significant shifts in the specific problems these firms have to solve. IBM morphed from a manufacturer of application specific integrated circuits, printed circuit boards and personal computers and laptops, into a data analytics, informatics and consulting business over ten years. At the same time, Nortel, Inc. could not replicate its success in the digital phone exchange and optical communications business in the fourth generation fixed wireless and telecommunications base station market. Circuit City applied its stock of problem solving routines and structures to the market for used cars – via CarMax – while at the same time Enron attempted an ultimately fatal diversification strategy based on 'loose associations and metaphors' [Rivkin, 2001]. A 'strategic problem solving advantage' can play an explanatory role in the outcome of such focal shifts by an incumbent company on the basis of a conception of the *transferability* of the complex problem solving advantage from one domain to another.

	Functional Adaptation to Complexity	Structural Adaptation to Complexity
On-Line Problem Solving	**Barriers to imitability (creating first mover advantage)**: imitator's limited ability to design organizational routines (including communication and coordination techniques and technologies) that embody adaptive complex problem solving algorithms of focal firm; **Barriers to transferability (limiting adaptation advantage)**: own limited ability to design organizational routines (including communication and coordination techniques and technologies) that embody new adaptive complex problem solving algorithms;	**Barriers to imitability (creating first mover advantage)**: imitator's limited ability to implement organizational routines (including communication and coordination techniques and technologies) that embody adaptive complex problem solving algorithms of focal firm; **Barriers to transferability (limiting adaptation advantage)**: own limited ability to implement organizational routines (including communication and coordination techniques and technologies) that embody new adaptive complex problem solving algorithms;
Off-Line Problem Solving	**Barriers to imitability (creating first mover advantage):** imitator's understanding of the computational structure of strategic problems by imitator; switching costs for sticky problem solving routines and interaction blueprint (executive strategy sessions, etc); **Barriers to Transferability (limiting adaptation advantage):** own understanding of the computational structure of strategic problems by imitator; switching costs for sticky problem solving routines and interaction blueprint (executive strategy sessions, etc);	**Barriers to imitability (creating first mover advantage)**: imitator's switching costs arising from re-allocation of incentives and decision rights to individuals and teams ('structural changes') **Barriers to transferability**: own switching costs arising from re-allocation of incentives and decision rights to individuals and teams ('structural changes')

Table 4: Summarizing the Implications for Imitability and Transferability of Functional and Structural Adaptations to Complexity Used for On-Line and Off-Line Strategic Problem Solving.

Table 4 summarizes barriers to imitability and transferability to the strategic search advantage derived from both functional and structural adaptations of both online and offline problem solving activities. Assume a 'high complexity environment' – one in which achieving globally optimal outcomes through strategic action requires the solution of an intractable problem. Then, the incumbent's advantage over a putative imitator is directly related to the imitator's ability to conceptualize and design (off line) and to implement (on-line) solution search procedures that can match or better the incumbent's own strategic search capabilities. The incumbent's transferability of its strategic solution search advantage to another market or to a new set of industry conditions relates to its ability to conceptualize and design (off line) and to implement and execute (on line) a set of strategic solution search procedures that are optimally matched to the new set of problems corresponding to the changed environment. Assuming that strategic search capabilities are sticky (resulting from inertia operating at both cognitive and behavioral levels, and from high switching costs associated with changing the allocation of decision rights and incentives in a firm, we can see our way to a set of predictions regarding the ability of an incumbent to transfer its strategic ingenuity, on the basis of the similarity between the problems the incumbent is currently solving and the problems it would have to solve in order to succeed in the new industry or changed environment. For instance, the canonical set of problems characterizing the informatics and 'big data' business – IBM's new focus – are sufficiently close to the kinds of hardware and software/firmware design problems the company had been previously solving. They can be represented by COVER and SET COVER problems (Appendix II) with large numbers of independent variables, representing logical consistency checks performed over the constraint sets of a VLSI design problem, or over the combinations of hypotheses that optimally explain a large data set. The computational perspective on firms predicts that this computational isomorphism of the problem domains should make for a smooth adaptation of IBM's dynamic search capabilities to the new problem domain: they are in the same complexity class (*NP*), have similar computational structures (COVER), have similar numbers of variables (>100) and should therefore respond in similar ways to the same set of approximation and randomization procedures for simplifying the search process.

If the complexity of problems represents a true hierarchy – as complexity grows in the number of independent variables of a problem statement and is discontinuous across the P-NP frontier, then we can further create horizons of transferability of strategic solution search capabilities: firms are less likely to be able to transfer such capabilities *upwards* in complexity (i.e. from NP-hard problems with fewer variables to NP hard problems with more variables; and from P hard problems to NP hard problems) than *upward* ; and they are more likely to be able to transfer such skills to *new* industries and environments presenting problems that yield to procedures that are *within* the firm's repertoire of dynamical search capabilities than to new industries presenting problems that which yield to solution procedures that are not.

7. Concluding Comments. The algorithmic language introduced here for the study of strategic problem solving yields a family of models of firm level problem solving at several levels: individual strategic managers, top management teams and organizations taken as a whole – which highlights the importance of adaptations to complexity as a source of adaptive advantage, and adds precision and resolving power to existing models of managerial rationality, adaptivity and dynamic search capability.

The complexity lens sheds light on fundamental questions such as: 'Why do firms as the specific pooled allocations of tasks to humans exist in the form they do?' and "What is the strategic value of organizational culture?' Novak and Wernerfelt, [2012] argue that firms constitute high-fixed, low-variable cost structural solutions to the problem of optimally assigning tasks to units of production in ways that minimize overall coordination costs by grouping together tasks requiring more frequent coordination into firms. Even in the absence of incentive mis-alignments, coordination games present coordinating agents with a multiplicity of equilibria whose selection or refinement is costly [Ganslandt, 2002]. The high fixed cost of firms as units of productive coordination correspond to the drawing of boundaries around the range of admissible coordinative equilibria, and therefore as a complexity driven adaptation of the value chain. More interestingly, perhaps, the methods for solving hard problems by approximate, local or stochastic methods considered in this paper involve tight coordination among the activities of several agents or teams: Intelligent randomization involves keeping track of the evolution of the search space and of dominated and 'promising' solutions in real time, and the constant exchange of information about the results of the local sub-searches. Local search involves keeping several 'almost good enough' alternatives on hand for quick re-insertion in the search process. Branch and bound methods involve a rough and iterative partitioning of the search space and the use of consistent heuristics for searching certain regions of that space rather than others. David Kreps' [Kreps, 1990] model of *organizational culture* as a set of focal points of repeated coordination games that have become common knowledge through repeated usage and are useful as methods for simplifying the problem of coordinative equilibrium selection because of their known-ness becomes directly relevant here, as adaptive solution procedures for *NP* hard problems require ongoing and fine-grained coordination among the activities of multiple problem solvers, and common knowledge of coordination rules and heuristics will make the difference between an efficient and an inefficient meta-algorithmic approach to solving an intractable problem. The computational perspective on organizational adaptation suggests that the clustering of tasks into firms will also vary with the computational complexity of the tasks and the degree of coordination required by the implementation of intelligently randomized or approximate implementations thereof.

The problem solving lens on strategy making and strategic process design suggests a path to a generic toolkit' for solving strategy problems and for the doing of strategy. Strategic competence, expertise and ingenuity exist should be treated separately from specific expertise in the technology, market and operational details of each firm: Problem solving 'prowess' should be transferrable between domains of practical expertise that confront a firm with problems of *similar* complexity and with *similar time and resource constraints*. Strategy making can be conceptualized as a 'hard-problem -solving activity', and the expertise of the strategist as an understanding of the *abstract* structure of strategic problems and the ability to conceive and deploy reliable and efficient procedures for the search of the solution space of such problems. Teaching strategy, on this view, should illuminate the algorithmic structure and computational complexity of strategic problems that are common across firms and industries. Leading strategy consulting firms (McKinsey & Co, The Boston Consulting Group Ltd., Booz, Allen and Hamilton Ltd., Bain&Co., Monitor Co.) fulfill a valuable function insofar as they function as generalized problem solvers that are able to abstract useful heuristics, representations and algorithms from across varied areas of practice and transfer them across firms and industries. A canonical language for describing strategy problems creates a 'problem solving skill transfer bridge' across firms engaged in apparently very different concrete problems of design, production, coordination, and competition that are computationally isomorphic: from banking to telecommunications, from software to semiconductors, from healthcare to transportation, from pharmaceuticals to petroleum refining.

I end with a qualification meant to assure those who fear that the modeling language herein makes the work of strategic managers, consultants and academics purely 'algorithmic'. While running a rule bound, step by step computational process is algorithmic (and therefore automatable), the problem of designing algorithms, meta-algorithms and heuristics for hard and intractable problems is not: Software *design* is not a task for which much successful software is currently written. This is for good reason: Designing meta-algorithms involves the matching of the solution space search heuristic to the structure of the overall problem and this matching process is not, in most cases, rule bound. Even if the view of strategies as algorithmically produced solutions to the problems the business faces over certain time scales is accepted, the role of the strategist as the (non-algorithmic) designer of these algorithms seems safe - at least for now.

<u>References</u>

Akerlof, G. A. 1970. The Market for "Lemons": Quality Uncertainty and the Market mechanism. *Quarterly Journal of Economics*, 84(3): 488-500.

Aragones, E., A. Postlewaite, I. Gilboa, and D. Schmeidler. 2003. Accuracy versus Simplicity: A Complex Trade-off. Mimeo, Yale University.

Arora, S., Barak, B., Brunnermeier, B. and Ge, R. 2010. Computational Complexity and Information Asymmetry in Financial Products . *Innovations in Computer Science (ICS) conference.*

Asano, T., and Williamson, D.P. 2002. Improved Approximation Algorithms for MAX SAT. *Journal of Algorithms,* 42: 173-202.

Back, T.1991. *Evolutionary Algorithms*. New York: Springer. .

Brooks, R. 1991. Intelligence without Representation, *Artificial Intelligence*. 47 139-159.

Brown, G.W. (1951) "Iterative Solutions of Games by Fictitious Play" In *Activity Analysis of Production and Allocation*, T.C. Koopmans (Ed.), New York: Wiley

Brueggemann, T. and Kern, W. 2004 An Improved Deterministic Local Search Algorithm for 3-SAT. *Theoretical Computer Science,* 329: 303-313.

Bryn, S., and L. Page. 1998. Bringing Order to the Web: A Hypertext Search Engine. Mimeo, Stanford University.

Bylander, T., D. Allemang, M.C. Tanner and J. Josephson. 1991. The Computational Complexity of Abduction. *Artificial Intelligence*. 49: 25-60.

Chapman, W.L., J. Rozenblitt, and A.T Bahill. 2001. System Design Is an *NP*-Complete Problem. *Systems Engineering*. 4: 222-229.

Charikat, M. 2000. Greedy Approximation Algorithms for Finding Dense Components in a Graph. *Proceedings of APPROX:* 84-95.

Chen, X., and X. Deng. 2006. Settling the Complexity of Two-Player Nash Equilibrium. *FOCS*. 261-272.

Cook, S. 1971. The Complexity of Theorem Proving Procedures, *Proceedings of the Third Annual ACM Symposium on the Theory of Computing.*

Cooper, G. 1990. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence.* 42: 393-405.

Cyert, R.M. and J.G. March. 1963. *A Behavioral Theory of the Firm*. New Jersey: Prentice-Hall.

Daskalakis, C., P. Goldberg, and C. Papadimitriou. 2006. The Complexity of Computing a Nash Equilibrium. *Journal of the ACM.*

Davis, J.P., K.M. Eisenhardt and C.P. Bingham. 2008. Complexity Theory, Market Dynamism and the Strategy of Simple Rules. *DRUID Working Paper.*

Denrell, J., C. Fang and D.A. Levinthal. 2004. From T Mazes to Labyrinths: Learning from model-Based Feedback. *Management Science,* 50(10): 1366-1378.

Denrell, J. and J.G. March. 2001. Adaptation as Information Restriction: The Hot Stove Effect. *Organization Science*, 12(5): 523-538.

Denrell,J., C. Fang and S.G. Winter. 2003. The Economics of Stretgic Opportunity. *Strategic Management Journal*, 24:977-990.

Denrell, J. 2007. Adaptive Learning and Risk Taking. *Psychological Review.* 114(1): 177-187.

Eisenhardt, K. and J.A. Martin. 2000. Dynamic Capabilities: What Are They?, *Strategic Management Journal,* 21:1105-1121.

Fogel, D.B. 1995. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, Piscataway.

Fortnow, L. 2009. The Status of P versus NP Problem. *Communications of the ACM.* 52(9). 78-86.

Garey, M.R., and D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP Completeness*. San Franscisco, Freeman.

Ganslandt, M. (2002), Communication, Complexity and Coordination in Games, in Zellner, A., H.A. Keuzenkamp and M. McAleer, eds., *Simplicity, Inference and Modeling: Keeping it Sophisticatedly Simple*, New York: Cambridge University Press.

Gavetti, G., D.A. Levinthal and J.W. Rivkin. 2005. Strategy Making in Complex Worlds: The Power of Analogy. *Strategic Management Journal*, 26:691-712.

Gavetti, G. and D.A. Levinthal. 2000. Looking Forward and Looking Backward: Cognitive and Experiential Search.*Administrative Science Quarterly*, 45:113-137.

Gavetti, G. and J.W. Rivkin. 2006. *On the Origins of Strategy: Action and Cognition over Time.*, Working Paper, Harvard University graduate School of Business Administration, Division of Research.

Ghosh, S.K., and Misra, J. 2009. *A Randomized Algorithm for 3-SAT*. Honeywell Technology Solutions Laboratory.

Gigerenzer, G, P. Todd, and the ABC Research Group. 1999. *Simple Heuristics that Make us Smart*. Oxford University Press, New York.

Gilboa, I., and E. Zemel. 1989. Nash and Correlated Equilibria: Some Complexity Considerations. *Games and Economic Behavior*. 1.

Holland, J. H. 1962. Outline for a Logical Theory of Adaptive Systems. *Journal of the ACM*. 9(3): 297-314.

Hromkovic, J. 2003. *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics*. 2nd edition, Springer, Heidelberg.

Karmarkar, N. 1984. A New Polynomial Time Algorithm for Linear Programming. *Proceedings of the 16th ACM Symposium on the Theory of Computing*. 302-311.

Karp, R.M. 1972. Reducibility Among Combinatorial Problems. In Miller, R.E., and J.W. Thatcher, eds., *Complexity of Computer Computations*. Plenum, New York.

Kauffman, S. 1969. Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. *Journal of Theoretical Biology*. 22: 437-467.

Kauffman, S. 1993. *The Origins of Order: Self Organization and Selection in Evolution*. Oxford University Press, New York.

Kreps, D.M. 1990. *Game Theory and Economic Modeling*. Oxford University Press, New York.

Lenox, M.J., Rockart, S.F., and Lewin, A. Y. 2006. Interdependency, Competition, and the Distribution of Firm and Industry Profits, *Management Science*, 52: 757-772.

Levinthal, D., and P. Ghemawat. 1999. Choice Structures, Business Strategy and Performance: An NK Simulation Approach. Working Paper 00-05, Wharton School.

Levinthal, D.A., and Warglien, M. 1997. Landscape Design: Designing for Local Action in Complex Worlds, *Organization Science*, 10(3); 342-57.

Lin, S., and B.W. Kernighan. 1973. An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research*. **21**. 498–516.

March, J. G.1991. Exploration and Exploitation in Organizational Learning. *Organizational Science*. 2: 71-87.

March, J. G., Simon H. A. 1958. *Organizations*. John Wiley, New York.Marr, D. 1982. *Vision*. H. Freeman and Co., San Francisco.

Marr, D. 1982. *Vision*. San Francisco: H. Freeman and Co.

Martello, S., and P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, New York.

Martin, R. 2007. *The Opposable Mind: How Successful Leaders Win Through Integrative Thinking*. Harvard Business School Press, Cambridge.

McKelvey, B. 1999. Avoiding Complexity Catastrophe in Coevolutionary Pockets: Strategies for Rugged Landscapes. *Organization Science , 10* (3): 294-321.

Michalewicz, Z., and D. Fogel. 2004. *How to Solve It: Modern Heuristics*. Springer, Heidelberg.

Minsky, M. 1961. Steps Toward Artificial Intelligence. *Proceedings of the Institute of Radio Engineers*. 49: 8-30.

Moldoveanu, M.C. 2011. *Inside Man*: *The Discipline of Modeling Human Ways of Being.* Stanford Business Books Stanford.

Moldoveanu, M.C. 2009. Thinking Strategically About Thinking Strategically: The Computational Structure and Dynamics of Managerial Problem Selection and Formulation, *Strategic Management Journal*. 30: 737-763.

Moldoveanu, M.C., and R. Bauer. 2004. On The Relationship Between Organizational Complexity and Organizational Structuration. *Organization Science*. 15(1): 98-118.

Moldoveanu, M.C., and J.A.C. Baum. 2008. The Epistemic Structure and Dynamics of Social Networks, *Social Science Research Network*. Paper 88795.

Moldoveanu, M. C., and R. L. Martin. 2008. *The Future of the MBA*. Oxford University Press, Oxford.

Moldoveanu, M.C. and R.L. Martin. 2009. *Diaminds: Decoding the Mental Habits of Successful Thinkers*. Toronto: University of Toronto Press.

Newell, A., & Simon, H. A. 1972. *Human Problem Solving*. New York: Prentice-Hall.

Novak, S. and Wernerfelt, B. (2012), On the Grouping of Tasks into Firms: Make-or-Buy with Interdependent Parts. *Journal of Economics and Management Strategy*. 21: 53–77

Page, L. 2001. Methods for Node Ranking in a Linked Database. *United States Patent*.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley, New York.

Pearl, J. 1990. *Causality*. Cambridge University Press, New York.

Peirce, C.S. 1998. *The Essential Peirce*. Chapter 12: Pragmatism as the Logic of Abduction, University of Indiana Press, Indiana: 226-241.

Porter, M. E. 1996. What is Strategy? *Harvard Business Review*. (Nov-Dec): 61-78.

Rivkin, J. 2000. Imitation of Complex Strategies. *Management Science*. 46: 824-844.

Rivkin, J.W. (2001). Reproducing Knowledge: Replication without Imitation at moderate Complexity. *Organization Science*: 12:274-293.

Rivkin, J. and N. Siggelkow. 2003. Balancing Search and Stability: Interdependencies Among Elements of Organizational Design. *Management Science*, 49(3): 290-311.

Rivkin, J. and N. Siggelkow. 2007. Patterned Interactions in Complex Systems: Implications for Exploration, *Management Science*, 53: 1068-1085.

Rubinstein, A.1993. On price Recognition and Computational Complexity in a Monopolistic Model. *Journal of Political Economy* 101: 473-484.

Rubinstein, A. 1986. Finite Automata Play a Repeated Prisoner's Dilemma Game, *Journal of Economic Theory*. 46: 145-153.

Schoening, U. 2002. A Probabilistic Algorithm for k-SAT Based on Limited Local Search and Restart, *Algorithmica*, 32: 615-623.

Siggelkow, N. 2002. Evolution Towards Fit. *Administrative Science Quarterly*, 47: 125-159.

Siggelkow, N., and Levinthal, D. 2005. Escaping Real (Non-Benign) Competency Traps: Linking the Dynamics of Organizational Structure to the Dynamics of Search. *Strategic Organization*. 3(1): 85-115

Siggelkow, N., and Levinthal, D. 2003. Temporarily Divide to Conquer: Centralized, Descentralized, and Reintegrated Organizational Approaches to Exploration and Adaptation. *Organizational Science*. 14: 650-669.

Siggelkow, N. and J. Rivkin. 2005. Speed and Search: Designing Organizations for Turbulence and Complexity. *Organization science*, 16(2):101-122.

Simon, H. A. 1991. Bounded Rationality and Organizational Learning. *Organizational Science*. 2(1): 125-134.

Simon, H. A. 1978. Rationality as Process and as Product of Thought. *The American Economic Review*. 68(2): 1-16.

Teece, D. and G. Pisano. 1994. The Dynamic Capabilities of Firms: An Introduction. *International Institute for Applied Systems Analysis Working Paper* WP-94-103.

Tversky, A., and D. Kahneman. 1986. Rational Choice and the Framing of Decisions. *Journal of Business*. 59: 251-284.

Wang, L., J. Zhang, and H. Li. 2007. An Improved Genetic Algorithm for TSP, *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*, Hong Kong.

Weinberger, E.D. 1991. Local Properties of Kauffman's NK Model: A Tunably Rugged Energy Landscape. *Physical Review*. 44: 6399-6413.

Weinberger, E.D. 1996.NP Completeness of Kauffman's *N-k* Model: A Tunably Rugged Energy Landscape. *Santa Fe Institute Working Paper 96-02-003.*

Wolpert, D. 2001. The Mathematics of Search. *NASA Ames Research Laboratory Working Paper.*

Wright, A. H., Thompson, R. K., and Zhang, J. 2000. The Computational Complexity of N-K Fitness Functions. *IEEE Transactions on Evolutionary Computation*. 44(4): 373-379.

Appendix I: P-Hard Problems, 'Easy' and 'Hard'. I take a 'short cut' to the exposition of problems with polynomial time solution algorithms (*P* hard problems), illustrated in Figure 11. The short cut consists of considering problems that are known to be solvable in polynomial time and showing how large classes of problems and sub-problems encountered in strategic problem solving can be understood as being algorithmically equivalent to them. This approach allows one to build a large scale, 'tree-structured' classifier for the problems of strategic management, which can then be applied to the task of classifying problems with respect to their expected time complexity and resulting solution costs.

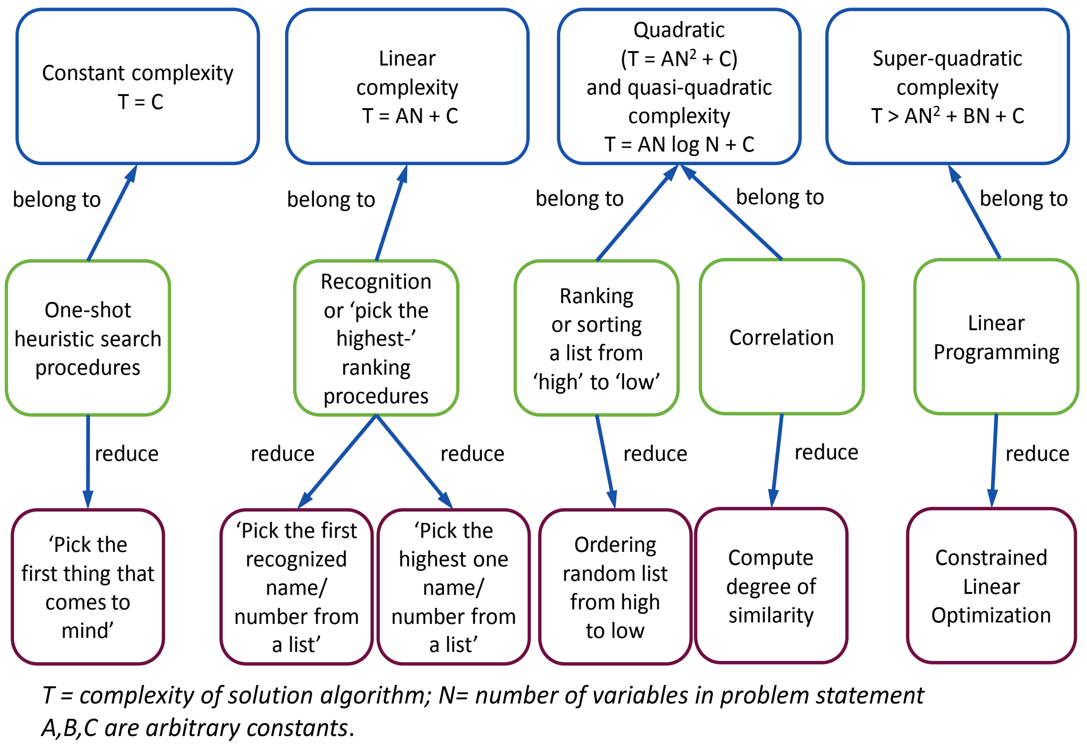

Constant complexity T = C
Linear complexity T = AN + C
Quadratic (T = AN² + C) and quasi-quadratic complexity T = AN log N + C
Super-quadratic complexity T > AN² + BN + C
belong to
belong to
belong to
belong to
belong to
One-shot heuristic search procedures
Recognition or 'pick the highest-' ranking procedures
Ranking or sorting a list from 'high' to 'low'
Correlation
Linear Programming
reduce
reduce
reduce
reduce
reduce
reduce
'Pick the first thing that comes to mind'
'Pick the first recognized name/ number from a list'
'Pick the highest one name/ number from a list'
Ordering random list from high to low
Compute degree of similarity
Constrained Linear Optimization
T = complexity of solution algorithm; N= number of variables in problem statement
A,B,C are arbitrary constants.

Figure 11: Complexity Regimes: I. The P-Class.

There are a few problems that make up 'the core' of strategy problem solving, and that have time complexity that is at most quadratic in the number of problem variables, i.e. $C(N)=O(N^2)$. *Constant* complexity problems are *invariant* to the size of the input. Although it sounds like no solution algorithm can be blind to the size of its input, the process of 'making strategy happen' presents frequent examples, in the forms of mental automatisms. A strategic manager can automatically discard market information that comes from a distrusted source, or, alternatively, automatically take advice that comes from a trusted source, even if the data that backs up the advice is present. If the problem is that of selecting the greatest total addressable market attainable from a given technological platform, then, either discarding the data altogether or taking the recommendation of the source data without parsing it oneself will obviously be 'blind' to the data set.

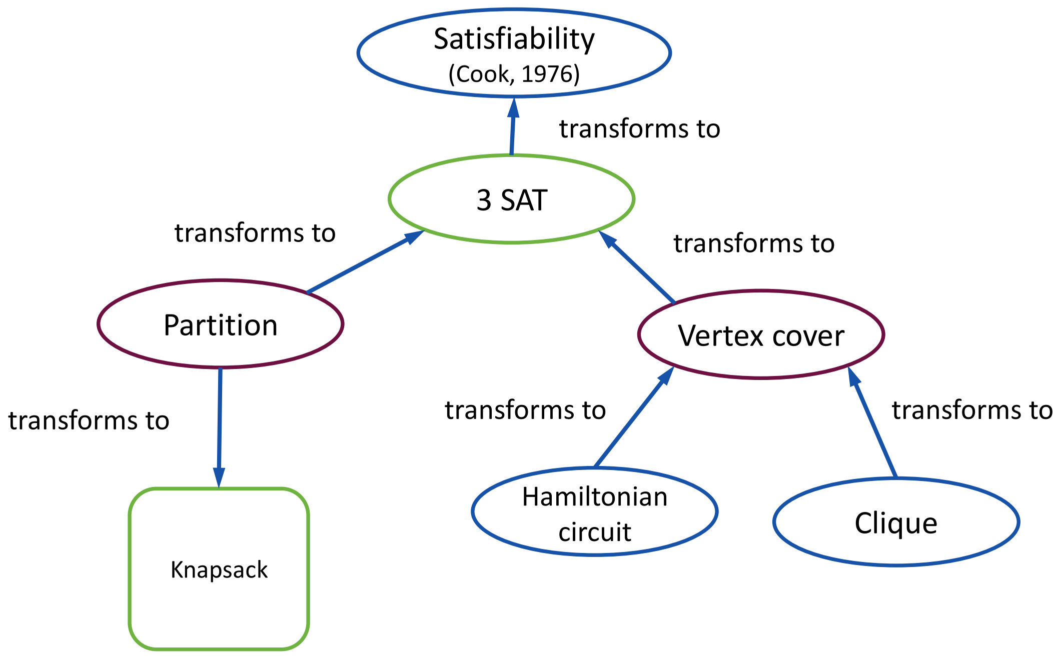
Linear complexity problems have solution algorithms with the property that $C(N) = O(AN)$, where A is some positive constant. They include 'frugal heuristics' [Gigerenzer et al, 1999] like the recognition heuristic: One can estimate the size of the largest rural suburban telecom services market in India, for instance, by picking the first (from a list of N) cities that one recognizes (or, to make the algorithm complete, picking at random if one recognizes no name). One can *rationalize* the use of this rule on the assumption that one is most likely to recognize the largest rural region. Using this heuristic to pick the best option from a list entails a worst case time complexity of $C(N) = N$, i.e., in the worst case, one has to read the entire list before getting to a city that one recognizes. The basic insight can be extended to 'chose the best' algorithms for lists under multiple criteria, in which case the number of M of criteria becomes a multiplier of the time complexity of the problem, i.e. $C(M,N) = MN$.

More sophisticated algorithms for efficiently processing random lists ordering to answer questions like *'Which is the best element under criterions c?'* can be devised, ranging from the tree searches that we encountered in the previous section $(C(M,N) = N \log_2(M))$ to multicriteria choice models that require re-visiting the list several times, and ordering according to each new criterion. 'Biases' [Tversky and Kahneman, 1986] in decision making under uncertainty appear, in this approach, as computational shortcuts for multi-criterion judgment formation, which may, in some cases, be ecologically adaptive.

Computing the c*orrelation* between two vectors or arrays is a good model for *analogical reasoning*, or for the kind of automatic processing involved in pattern recognition in low level or high level vision [Marr, 1982]. The time complexity of correlating two vectors of lengths N and M is given by: $C(Corr(X,Y))=2MN-1$ and is hence linear in the length of the vectors. A 'pattern' of behavior (a demand fluctuation, a competitor's response) can be encoded as an MxM sample array of samples, and the complexity of correlating such a pattern with a 'known' pattern stored in working memory will be proportional to M^2N^2. A good working measure of the complexity of simple pattern recognition (comparing an observed pattern of industry behavior with one of K patterns stored in memory) is thus $C(\text{"Pattern recognition"})=K x\, M^2N^2 x \log_2(K$, comprised of the M^2N^2 required for computing correlation coefficients and the $K\log_2(K)$ operations required to classify the pattern as 'most like' one of the patterns stored in memory.

Linear programming (*LP*) problems can be used to model *n*-variable linear optimization problems under *L* constraints, such as those arising from optimal price or quantity selection in either a monopoly or a perfectly competitive market, or profit-maximizing inventory planning under size and storage cost constraints. [Karmarkar, 1984] showed that the complexity of such problems is super-quadratic in the number of optimization variables, and linear in the length (in bits) of the total input, i.e. $C(n, L) = O(n^{3.5}L)$. *LP* problems have become a staple of both managerial practice and of training in managerial economics, and algorithms for solving *LP* problems have been 'automated' by numerous commercially available software packages and sub-routines.

Appendix II: *NP* Hard and *NP*-Complete Problems: The Intractable. I shall take a short cut to the exposition of intractable problems that mirrors the approach taken in the theory of computational complexity over the past 39 years [Cook, 1971; Karp, 1972; Garey and Johnson, 1979; Papadimitriou, 1994; Hromkovic, 2003]. Whereas *P*-hard problems are solvable by a deterministic algorithm in a number of operations *(C(N))* that is at most a polynomial function of the number of the problem variables, $(C(N) \leq P^k(N))$, non-deterministic polynomial-time problems (*NP* for short) can only be solved by a deterministic algorithm in a number of steps that is a super-polynomial function of the number of problem variables, i.e. $C(N) > P^k(N))$, and, typically, an exponential or super-exponential function of the number of variables: $C(N) \geq e^N$; or, by a *non*-deterministic algorithm (hence the name) in a number of steps that is a polynomial function of the number of variables. The short cut I shall take represents *NP*-hard problems in terms of a hierarchy (Figure 12) of problems that are reducible by polynomial-time transformations to a problem proven [Cook, 1971] to be *NP*-hard, the *k*-satisfiability (*kSAT*, with $k>2$) problem, discussed below. Subsequent proofs of *NP*-hardness [Garey and Johnson, 1979; Fortnow, 2009] take the reductive form of transformations of any problem onto either a *kSAT* problem or onto a problem that is polynomial time reducible to a *kSAT* problem.



-**Figure 12: Complexity Regimes: II. The NP-Class. T ≥ *a* exp(*bN*).**

NP hard/complete problems can be used to model and transform large classes of problems arising in business strategy and distinguish between problems that can be straightforwardly 'encoded' in the language of generalized or canonical problems and problems that can be 'transformed' into such problems by methods that have characterized the progress of the field of computational complexity theory to date, as follows:

<u>K-SAT and MAX-SAT</u>. The *KSAT* problem is a *decision problem*. It asks for an assignment of truth values to a set of elementary propositions that satisfies a *k*-variable formula or clause, expressed as a Boolean function of the elementary propositions. For example, the problem may be to find the set of truth assignments to the elementary propositions X_1, X_2, X_3, X_4 (where '0' denotes 'false' and '1' denotes 'True') that satisfy the formula $F=(X_1\hat{}\sim X_2\hat{}X_4)\&(X_1\hat{}\sim X_2\hat{}\sim X_3)=1$ ('True'), where '^', '&' and '~' are the standard operators 'or', 'and' and 'not' of Boolean logic. The intuition behind the problem being hard in the intuitive sense is that all possible truth assignments of X_1, X_2, X_3, X_4 must be checked against *F* in order to determine whether or not they satisfy it; and the problem is the first to be proven to be *NP* complete [Cook, 1971]. Thus for a deterministic search process of a *KSAT* solution, we have, $C(k)\approx e^K$. The associated *maximization problem*, *MAXSAT*, is that of finding a truth assignment to X_1, X_2, X_3, X_4 that maximizes the number of satisfied clauses (*F* has 2, each with 3 variables), and is *NP*-hard. *K SAT* can be used *directly* to encode the problem of the design of strategic contracts or relationships, where the individual variables encode possible states of the world that are relevant to the contract, the logical clauses encode contractual clauses and formulas encode the consequences of contingent agreements among firms (Figure 13). Truth assignments to individual variables represent eventualities, clauses then encode possible consequences to the firm, and the problem of contract design refers to the problem of determining conjunctions of states of the world that will lead to particular payoffs or consequences.

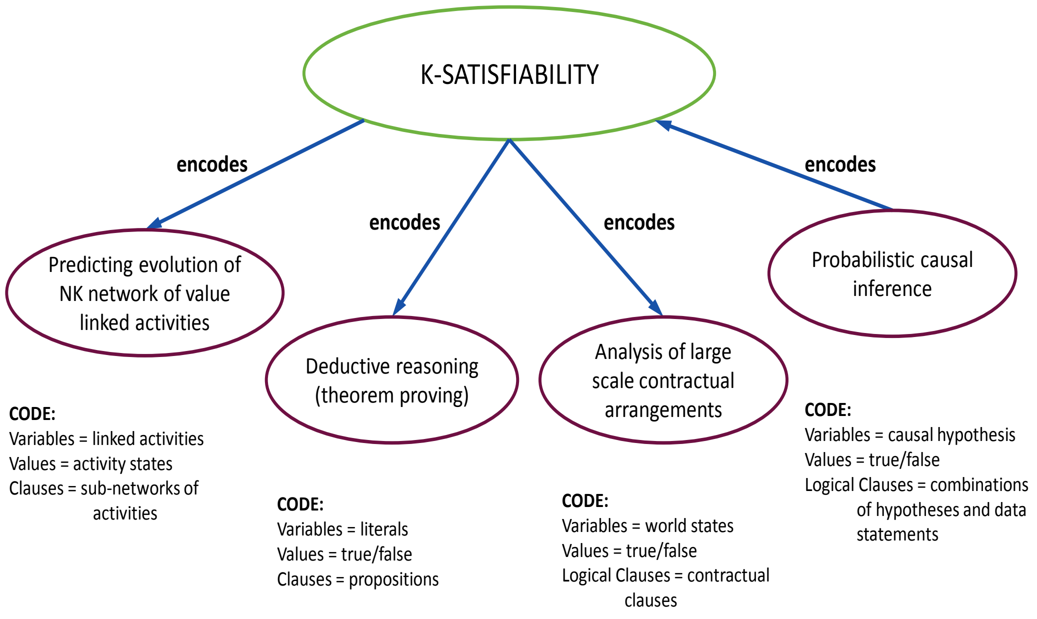

K-SATISFIABILITY
encodes
encodes
encodes
encodes
Predicting evolution of NK network of value linked activities
Deductive reasoning (theorem proving)
Analysis of large scale contractual arrangements
Probabilistic causal inference
CODE:
Variables = linked activities
Values = activity states
Clauses = sub-networks of activities
CODE:
Variables = literals
Values = true/false
Clauses = propositions
CODE:
Variables = world states
Values = true/false
Logical Clauses = contractual clauses
CODE:
Variables = causal hypothesis
Values = true/false
Logical Clauses = combinations of hypotheses and data statements

Figure 13: The NP Hard Problem K-SAT and the Business Problems It Encodes

Cook [1971] uses *KSAT* to encode 'theorem-proving procedures'. This entails that the problem can be used to model *deductive* reasoning more generally, which can be encoded as the problem of figuring out whether or not a formula (a 'theorem') is logically compatible with a set of assumptions (the 'axioms'). Cooper [1990] uses the problem to model the problem of causal inference using probabilistic networks [Pearl, 1990], which asks whether or not a set of causal hypotheses (which play the role of the axioms) provide a causal explanation for a set of data points (or, evidence statements). The modeling 'maneuver' that Cooper makes is to model conjunctions of causal hypotheses and single data points as Boolean formulas, wherein the problem of finding the set of hypotheses that provide a minimally acceptable 'inference network' becomes that of finding a truth assignment to the variables of *KSAT* that satisfy the resulting formulas.

Thus, *KSAT* can be used to encode not only 'deterministic' problems of deductive construction, but also problems that incorporate incomplete information and uncertainty, and which admit of multiple possible causal inferences. It can therefore be used to model the 'root cause analysis' that top management teams and consulting firms perform when trying to understand the most plausible set of causes for strategically important events (sudden downturn in demand for the firm's products) that can plausibly have been 'caused' by multiple causal factors (change in clients' tastes, introduction of a competitive product, introduction of a substitute, 'random' seasonal fluctuation) which may or may not interact with one another.

Rivkin [2000] showed that the problem of predicting the evolution of a Boolean network of *N* nodes (each of which can take on the value of 0 or 1 depending on its prior state and the current state of other nodes with which it has primary links) with $K>2$ links to other nodes maps into the *KSAT* problem and is therefore *NP* complete in its decision problem version ('Is future state *S* compatible with system model and initial conditions?'). If such a network is used to model the value-linked activity set of a business (with *K* denoting the number of activities whose states matter to the value added by any one activity), then the problem of purposive strategic change is *NP*-complete and therefore strategic design is 'intractable'. The conclusion depends on the validity of the representation of a set of the firm as a net of value-linked activities [Porter, 1996], and as one which behaves like a Boolean network (i.e. the links model deterministic relationships). Even in such a case, however, it will be seen below that the problem of strategic change, even though theoretically intractable, may not be practically intractable, as there are classes of meta-algorithms and heuristics that he been developed specifically for the purpose of providing 'good enough' solutions to the *KSAT* and other *NP*-hard problems.

KSAT problems (*3SAT* in particular) also model *abductive* reasoning, defined, since Charles Sanders Peirce [1998(1903)] as "inference to the best explanation". An abduction problem takes as input a set of data, *D*, to be explained, and set of hypotheses, $\{H\}$, and a mapping $e(\{h_i,\}, \{d_j\})$ from subsets of *H* to subsets of *D*. The problem asks for an *explanation* in the form of the minimal subset of *H* that completely explains *D*. [Bylander et al, 1991] reduce the problem of determining whether or not an explanation exists (i.e. a subset of *H* that explains all of *D*) to *3SAT,* by assigning a variable in the 3SAT problem and its negation to an incompatible set of hypotheses, and each Boolean expression (a function of the variables corresponding to the hypotheses) to a datum to be explained; in which case a complete explanation exists only if the Boolean expression is satisfiable by the assignment of truth values to the set of hypotheses. Abductive reasoning has been used to model managerial thinking 'in practice' [Martin, 2007; Moldoveanu and Martin, 2008], as it represents a pragmatic combination of the salient features of deductive and inductive reasoning as it emphasizes both 'validity' (in the form of explanatory coverage, or 'completeness') and reliability (in the form of generalizability, proxied for by parsimony). The reduction of the 'completeness' problem of abduction to KSAT suggests that seeking valid explanations alone can function as a source of explosive growth in time complexity of the managerial problem, as the number of causal hypotheses and data points increases. We will see below that satisfying the parsimony condition in abductive reasoning gives rise to an *NP* hard problem.

TSP. The *Traveling Salesman Problem* (TSP) (Figure 14) typically represents the problem faced by a salesman who must find the minimum-distance (or minimum time) circuit that takes him to each of *N* cities, given that he knows the distance (time) which separates any pair of cities. The problem is polynomial-time reducible to the HAMILTONIAN CIRCUIT problem [Karp, 1972] which in turn is polynomial-time reducible to the KSAT problem, and hence it is *NP* hard. The intuition behind its complexity measure is that the number of paths that need to be searched is proportional to *N!,* which entails, by Stirling's formula, that $C(N)=\left(\frac{N}{e}\right)^{N}\sqrt{2\pi N} > P^{k}(N)$, and the reduction to a known *NP*-hard problem assures that there is no polynomial time short cut to a polynomial time solution. The TSP can be used to encode (Figure 15) a number of strategically important logistical or operational problems, such as minimizing the temporal or spatial length of the paths of work-pieces on a factory floor, or maximizing the efficiency of a traveling sales force or of a distribution system.

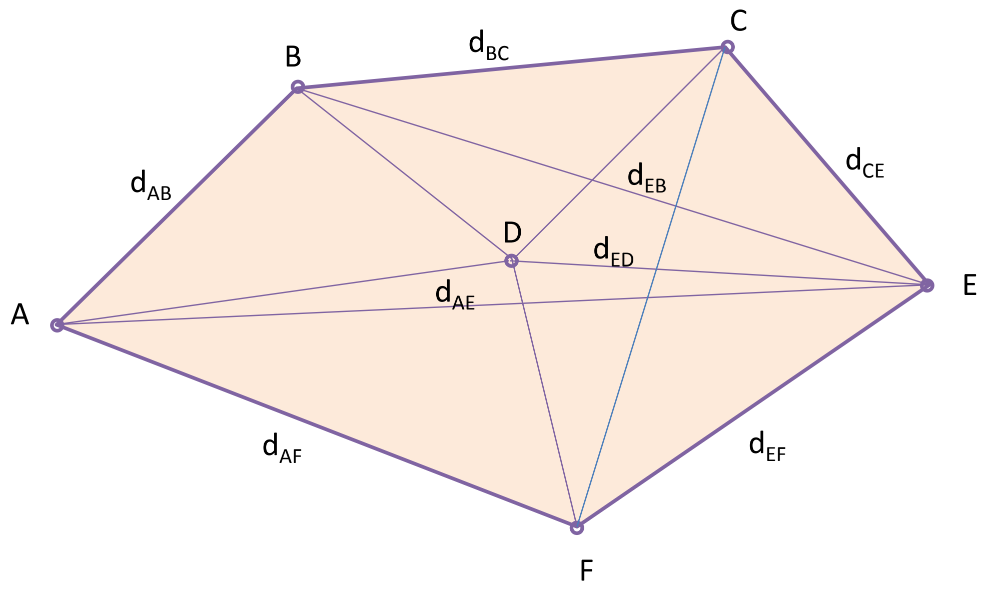


Figure 14: The Traveling Salesman Problem with N=6 Cities. The Goal Is to Find the Minimum Total Distance Path Connecting All of the Six Cities.

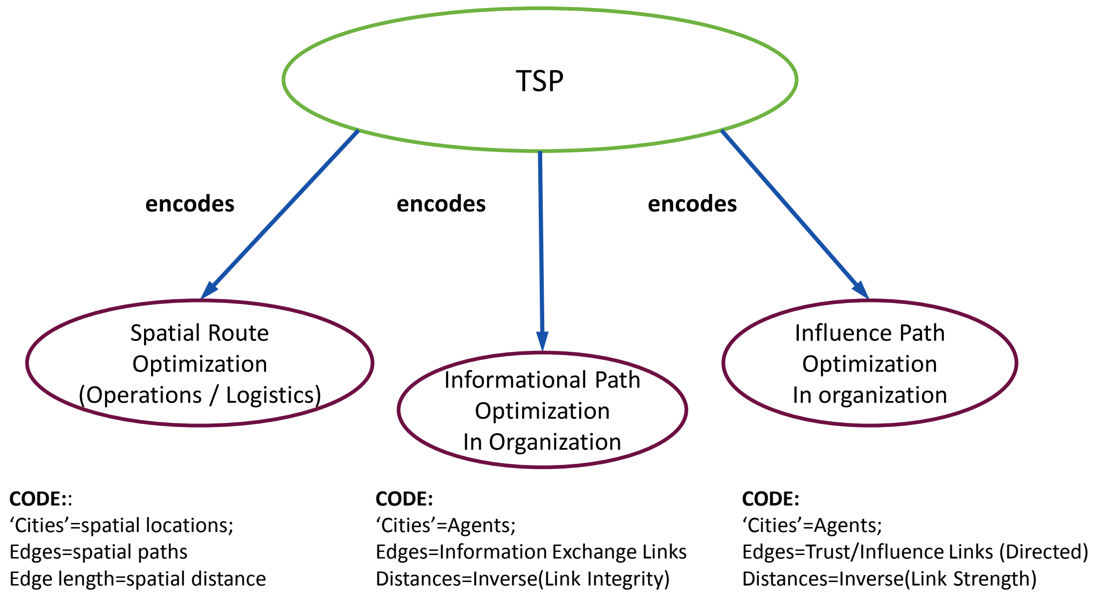

TSP
encodes
encodes
encodes
Spatial Route Optimization (Operations / Logistics)
Informational Path Optimization In Organization
Influence Path Optimization In organization
CODE:: 'Cities'=spatial locations; Edges=spatial paths Edge length=spatial distance
CODE: 'Cities'=Agents; Edges=Information Exchange Links Distances=Inverse(Link Integrity)
CODE: 'Cities'=Agents; Edges=Trust/Influence Links (Directed) Distances=Inverse(Link Strength)

Figure 15: The NP Hard Problem TSP and Business Problems It Encodes

It can also be used to encode problems that are directly relevant to organizational design or the optimization of the operation of a top management team. Mapping, for instance, individuals onto 'cities' and the affective distance among individuals (the inverse of the influence of individual *i* on individual *j*) onto spatial or temporal distances (the edges of the network) the *TSP* maps into an 'optimal influence strategy' problem, where the goal is to find the optimal 'persuasion path' through a senior management team. This problem in turn can represent both the 'CEO problem' ('how to persuade the members of my executive team or board of directors of a new strategic path?') and the problem of 'strategic selling' ('how to persuade the key decision makers and influential agents of a strategic customer of making a large scale commitment to my product, service or solution?') Alternatively, if we map the nodes onto individuals and edges onto information exchanges among individuals, whose length is measured by the inverse of the probability of truthful or trustful information transfer between two managers (the 'integrity' of the informational link between them), the *TSP* maps into an 'optimal information strategy design problem', whose solution helps strategic managers promulgate news and rumors most efficiently. In this case, the promulgator of the rumor would want to be both the start and the end point of the informational path of the rumor through the network so that he or she can 'verify' (or, authenticate) the fidelity with which the rumor has been transmitted through the network.

KNAPSACK. The *KNAPSACK* Problem (KSP) [Karp, 1972; Martello and Toth, 1990] represents the problem of optimally packing a knapsack of known total volume *V* with a set of *k* utensils out of *N* possible options so as to maximize the total utility of the set of utensils included, subject to all utensils fitting in *V*, given knowledge of the volume and the value of each utensil. Under the constraints are that no fraction of a utensil can be taken along ("0 or 1") and that each utensil can be included only once, KSP presents a search space that comprises 2^N possible options (the number of subsets of *N)*, providing an upper bound $C(N)=2^N$, which the reduction of *KSP* to a problem known to be polynomial-time reducible to *KSAT* (namely, PARTITION) also confirms as a lower bound on the complexity of an exact solution using a deterministic algorithm.

KSP is known to be a highly versatile modeling tool, having been used in the design public key cryptosystems (where it appears as the SUBSET SUM problem [Martello and Toth, 1990]), and it lives up to its versatility in encoding strategy problems. It can be used to represent the problem of strategic product design [Chapman, Rosenblitt and Bahill, 2001] under lumpy constraints that arise from existing platforms and developing technical standards (which map into the 'utensils'), the problem of the optimization of the cost structure of a manufacturing business by choosing over non-divisible activity sets with known costs and benefits and the problem of mapping technological platform features into the features of strategic products, under total cost-of-goods-sold constraints, among others. It can also be used (in its cryptographic form via the SUBSET SUM problem) to model the problem of decoding the jargon used by experts that seek to hijack the position power of top managers with their knowledge power by the use of professional codes as a form of public key cryptosystem, and the problem of 'infiltrating' technical standards proceedings that use jargon as a barrier to entry.

VERTEX COVER. The VERTEX COVER (VC) problem is a decision problem that relates to finding a subset *v* of at most *K* of the *N* vertices *V* of a graph *G(E,V)*, where *E* denotes the set of connecting edges, such that *v* will include the vertices that together touch all of the edges *E* of *G*. The VC problem was proven to be NP-complete by reduction to PARTITION [Karp, 1972] and a brute force algorithm will find the solution in $C(N,K)\approx 2^K N$. The associated *NP*-hard optimization problem is that of finding the *minimum* vertex cover of G, i.e. that of finding the minimal *K*. The *VC* problem straightforwardly encodes problems relating to the firm's 'network strategy'. It is well documented that a firm's position within its industry network matters to its strategic performance, which raises the problem of 'managing' the firm's 'networking strategy' [Moldoveanu, 2009] by seeking the network ties that maximize the firm's 'network strategic advantage'. What underlies the solution to most such problems is recognizing the 'core' of well-connected firms within the industry, i.e. the network's 'vertex cover'. The *VC* problem thus construed encodes the problem of 'strategic network sensing' – of mapping the firms within a network that together 'span' the entire network.

The VC problem also encodes problems related to understanding and manipulating the 'epistemic networks' [Moldoveanu and Baum, 2008] that arise within executive teams and are causally relevant to the ability of the team to coordinate or co-mobilize. An epistemic network is a network defined by a set of individuals *I*, a set of propositional beliefs *P* and a set of links among individuals and beliefs that denote the 'knows' or 'believes' operator. Individuals *A*, *B*, *C* and proposition *R* are linked by an epistemic path if *A* believes *B* believes *C* believes *P, for instance.* The problem of figuring out the set of central beliefs in a top management team maps onto the problem of figuring out the minimal set of propositions that together 'span' the set of individual members of the team – which is the minimal vertex cover of the epistemic network of beliefs of the top management team. These beliefs are important because they function as focal points in coordination games and scenarios and in mobilization scenarios [Moldoveanu and Baum, 2008], and thus finding and controlling them is important for narrowing down the set of equilibria of the coordination and mobilization games that the top management team plays. Relatedly, the problem of figuring out which top manager(s) are most 'in the know' (who knows most and who knows most about what others know) becomes the problem of figuring out the minimal vertex cover of the network of *agents* of the top management team.

CLIQUE is a decision problem (CP) that asks whether or not a graph *G(V,E)* of *N* vertices and *E* edges has a clique of size *k* (defined by a fully connected sub-network *SG* of *G*). The associated optimization problem asks for the minimal/maximal clique that graph *G* possesses [Garey and Johnson, 1979]. The intuitive time complexity of a brute force approach to CP is $C(N,k)=N^k k^2$, and the reduction of CP to VC [Karp, 1972] attests to the tightness of the time complexity bound. *CP* is also a versatile strategic problem modeling tool (Figure 16): it has been used [Gilboa and Zemel, 1989] in its decision form to model a problem considered 'prototypical' for strategic choice processes - that of finding a particular Nash equilibrium (NE) set of strategies that give a payoff of at least *P* in a competitive game (which includes: a Nash Equilibrium in which a player makes a certain minimum payoff, a Nash equilibrium whose support contains a certain strategy, and a Nash Equilibrium in which the aggregate payoff of the players exceeds a certain number). The problem of finding *a NE* is one that can in some cases be modeled by *LP* (as will be demonstrated below for a specific 2 player game). Subsequent work [Chen and Deng, 2006; Daskalakis, Goldberg and Papadimitriou, 2006] showed that the problem of finding the Nash Equilibrium of a game cannot, in general, be solved in polynomial time, as it is a member of a subclass of *TFNP* ('Total Function Non-deterministic Polynomial Time') - the function-theoretic equivalent of *NP*-complete decision problems and *NP*-hard optimization problems.

SET COVER. There are two versions of this problem: EXACT COVER and (simply) COVER. EXACT COVER (EC) asks for whether or not there exists a collection of pairwise disjoint subsets of a given, finite set S whose union is equal to S. [Aragones et al. 2003] show that EC can be used to represent the problem of linear regression, i.e. of finding the set of predictors $\{X_1, \ldots, X_K\}$ for a set of observations $\{Y_j\}$ that contains at most *k* elements subject to the correlation coefficient between the predictors and the predicted variable being at least *r*. The time complexity of a brute force solution to EC is intuitively given by the complexity all possible subsets of *S*, i.e. $C(EC(S))=2^{|S|}-1$, and the reduction of EC to *KSAT* [Karp, 1972] assures us that the problem is indeed in *NP*.

The problem of induction, on the other hand, i.e. of finding the minimal set of rules or generalizations that are consistent with a set of data, is transformable to the COVER problem, which asks whether or not there exists a set of *n* subsets of *S* whose union is equal to *S* [Aragones et al, 2003]. If the data are encoded as a matrix whose *(i,j)* entries represent the degree to which sample *i* has attribute *j*, then a *rule* is one that states that no sample with property *l* will fail to exhibit property *k*, for instance. The minimal set of rules, then, will be the smallest set of subsets of the data set that exactly correspond to a set of rules. COVER and EXACT COVER are versatile enough to encode other problems of strategic interest, such as the optimal design of teams comprising individuals with potentially overlapping skill sets (the set S is the set of skills needed for a task, the subsets of S are the skills corresponding to each individual) or conflicting personality characteristics for the optimal pursuit of tasks requiring certain sets of skills (software design) or personality types (sales).

Finally, the problem of *parsimonious abduction* – of figuring out the minimal set of hypotheses that together explain a data set – maps into a version of the COVER problem [Bylander et al, 1991], that asks for the minimum set of subsets of a set whose union is the set itself ('collectively exhaustive'). If we let the set in question encode H_{min} the set of explanatory hypotheses of at least minimal plausibility, then the problem of parsimonious abduction maps into the optimization version of the COVER problem. If we further specify that allowable explanations must consist of 'mutually exclusive' hypotheses, then the parsimonious abduction problem maps into the optimization version of the EXACT COVER problem.