NEEDED: AN EMPIRICAL SCIENCE OF ALGORITHMS

J. N. Hooker

Carnegie Mellon University, Pittsburgh, Pennsylvania

Revised December 1993

Deductive algorithmic science has reached a high level of sophistication, but its worst-case and average-case results seldom tell us how well an algorithm is actually going to work in practice. I argue that an empirical science of algorithms is a viable alternative. I respond to misgivings about an empirical approach, including the prevalent notion that only a deductive treatment can be "theoretical" or sophisticated. NP-completeness theory, for instance, is interesting partly because it has significant, if unacknowledged, empirical content. An empirical approach requires not only rigorous experimental design and analysis, but also the invention of empirically-based explanatory theories. I give some examples of recent work that partially achieves this aim.

There are two ways to study the performance of algorithms. One is analytical and relies on the methods of deductive mathematics. The other is empirical and uses computational experiments.

Only the first approach has developed into a science, and I believe that many researchers and practitioners feel, deep down at least, that this science is inadequate to its task.

True, many brilliant results have been proved regarding the worst-case or average-case complexity of algorithms. A number of hard-won bounds have been derived on the quality of the solution delivered by inexact algorithms, both in the worst case and in the average case.

But these results do not usually tell us how an algorithm is actually going to work on practical problems, or why. The complexity results are asymptotic or apply to a worst case that seldom occurs. The average-case results presuppose a probability distribution over randomly generated problems that is typically unreflective of reality. Furthermore, results of either kind are usually obtained for the simplest kinds of algorithms. The complex algorithms typically used in practice, not to mention the all-important tricks that are engineered into commercial codes, are currently beyond the reach of deductive algorithmic science.

A mathematical breakthrough cannot be ruled out. Someone may invent powerful new methods that can answer algorithmic questions that clever researchers can now only chip away at. But in the meantime, the theorems seem harder and harder to prove. After years of intense effort, we still find ourselves making very strong and therefore unrealistic assumptions to get results.

The only alternative on the horizon seems to be computational testing. My thesis is that the empirical approach is a viable alternative that should be pursued more consciously and more rigorously. In other words, we should try to build an empirical science of algorithms.

Computational experiments are already widely reported in scholarly publications. But these efforts fall short of science on several levels. To begin with, the testing is usually quite informal, at least in the OR literature. One occasionally sees tests conducted according to the principles of experimental design, or results analyzed using rigorous statistical methods. But only occasionally. Not even minimal standards of reproducibility are observed by most authors. Crowder, Dembo and Mulvey (1978) found the state of affairs sorry enough that they would not go so far as to recommend reproducibility by others, but only that an investigator be able to reproduce *his or her own* results.

It is symptomatic of the situation that, in OR and computer science, one cannot publish reports that an algorithm does *not* perform well in computational tests. Negative results are as important as positive results and are routinely reported in other empirical sciences. But the OR and computer science communities do not judge the publishability of results, whether they

be positive or negative, primarily on the basis of their value as empirical knowledge. Positive results are published because they are a practical selling point for the "theoretical" work that occupies the first 90% of the article. Negative results are relegated to obscurity because they do *not* demonstrate the applicability of the "theory." We recognize their importance on an unofficial level but must rely on grapevines and email to find out about them.

Advantages of an Empirical Science

An empirical science of algorithms would immediately sidestep several of the problems that beset a purely deductive science.

- It does not rely on proving hard worst-case and average-case theorems.
- Unlike worst-case analysis, it can focus on typical problems.
- Unlike average-case analysis, it need not restrict itself to a simple and unrealistic distribution of random problems.
- It can finesse the issue of *characterizing* a typical class of problems.

The last point deserves expansion. We want to know the performance of an algorithm on typical problems, but we often do not know how to characterize them. So even if we could carry out average-case analysis for any problem distribution we wish, we would be unsure about what distribution to use.

Empirical science is equally stymied by this problem if it approaches it in the usual way. The usual way is to collect a set of benchmark problems and compare algorithmic results on them. But any choice of problems is open to the criticism that it is unrepresentative.

There is another way, however. One can investigate *how algorithmic performance depends on problem characteristics.* The issue of problem choice therefore becomes one of experimental design. Rather than agonize over whether a problem set is representative of practice, one picks problems that vary along one or more parameters. Many investigators already do this

informally, as when exploring the effect of matrix density on algorithmic performance. Over time, one may discover the important characteristics and learn to predict how an algorithm will perform on a given problem class.

Empirical Science Involves Theory

Let us next dispose of the notion that empirical work in algorithms is somehow the opposite of theory. True, in its presently impoverished state it often consists of little more than tables of computational results on the last pages of journal articles. But it can be every bit as theoretical as, say, complexity theory.

In its early stages, an empirical study of an algorithm might well involve nothing more than running a few tests to see what happens. But after a while one develops an informed hunch about what is likely to affect performance. This is a *hypothesis*. The hypothesis is then tested empirically, using time-honored techniques of experimental design and statistical analysis. Eventually one may put together a unifying and coherent picture that appears to explain why certain factors are important. This is a *theory*. From the theory one can deduce consequences that can be put to the test.

This sort of development is a well-trod path in empirical sciences. It shows that theory, far from being antithetical to empirical science, is its culmination. Anyone who thinks that empirical science is nontheoretical should take a look at quantum electrodynamics.

A source of confusion is that the word 'empirical' has one sense in which it refers to something based purely on observation without theoretical depth. But I use the word in its classical sense, which simply means something that is ultimately answerable to experience. As for theory, it is again symptomatic that, within the OR and computer science communities, the word often connotes a deductive science consisting solely of theorems and proofs. It is time we broke out of this mindset.

I do not mean to say that theorems and proofs play no role in empirical science. To deduce a consequence of a theory for testing is in a sense to prove a theorem. But unlike the situation in

deductive science, the "theorem" cannot be accepted as true unless it squares with observation.

How do I know that deducing the consequences of an empirical theory of algorithms will not be as hard as proving worst-case and average-case results? I do not. Deducing consequences of empirical theories is sometimes very hard. It took many years to derive the possibility of black holes from Einstein's gravitational field equations. But since we know that theorem-proving in a deductive science of algorithms is hard, it behooves us to give empirical science a chance.

Why the Resistance to Empiricism?

If an empirical science of algorithms is so viable an alternative, why has it not caught on? Why do many people actively resist the idea?

I can identify several reasons, none of which I think are legitimate. I will begin with the most frivolous and progress to the most serious.

One practical reason researchers do not invest more energy in rigorous empirical work is that it is sometimes considered lowbrow or unsophisticated. There is at least a widespread impression that it is hard or impossible to get a purely empirical algorithmic study published in some prestigious journals. Behind this, in part, is a lack of standards for distinguishing high-quality empirical work.

But the acceptability of empirical work seems to be growing even as this article goes to press. Journal editors can be encouraged to seek out referees who have done rigorous empirical studies. Refereeing standards will evolve, particularly as the empirical science develops.

Another objection to empirical work is that it is inherently irreproducible. Everyone knows that the performance of an algorithm (e.g., number of elementary operations required) can vary by an order of magnitude with the details of implementation. The data structures, coding style, compiler, machine, etc., all matter. How can testing show an algorithm to be efficient or inefficient when one can only test the algorithm-*cum*-implementation?

The problem is one of distinguishing the phenomenon (here, the algorithm) from the apparatus used to investigate it (here, the data structures, code, etc.). This is an old problem

in empirical science that is attacked by *developing a science that governs the apparatus.* An astronomer, for example, can test a theory regarding the brightness of stars by photographing them through a telescope. Another astronomer may get different results on a different night. But their results can be reconciled if they understand how atmospheric conditions and the construction of their telescopes affect the photograph. Similarly, we need an understanding of how data structures and coding practices affect our observations of algorithmic behavior.

Perhaps this is a little too facile, however, since whereas it is easy to distinguish stellar brightness from atmospheric interference, etc., it is not so easy to distinguish algorithms from the data structures and code that embody them. Perhaps an algorithm is not really well specified until implemented. Or to put it differently, the very act of implementing (i.e., observing) an algorithm alters the algorithm being observed.

It is not so obvious that deductive science is better positioned to deal with this problem than empirical science. But this aside, it is again a problem that empirical science has dealt with. In quantum physics, the method of observation is notoriously inseparable from the phenomenon measured. Admittedly this poses a profound conundrum that by many accounts remains unresolved. But quantum physics has enjoyed spectacular success, both in theoretical power and in practical application, despite it. Perhaps there is a role for an uncertainty principle in algorithmic science. In any event, I think we can agree that this empirical endeavor is beginning to look less and less like a humdrum affair of tabulating CPU times.

The Main Objection

The main source of uneasiness over an empirical approach, however, seems to run deeper. It is too much like verifying that opposite interior angles are equal by measuring them with a protractor. The behavior of an algorithm over any finite period of time is in principle deducible, using formal methods, from a statement of an algorithm. It seems fundamentally wrong-headed to use empirical methods to study what is essentially a formal system.

I have two responses to this objection. One is that the natural phenomena we study

very successfully with empirical methods may themselves be deducible in a formal system. In fact this was the prevailing view in the early days of modern science. Such luminaries as Descartes, Leibniz, and Newton believed, and Kant allowed for the possibility, that physics could in principle be studied with the same deductive methods as geometry, if only we had the intelligence to do it. It is only our dim-wittedness that obliges us to use empirical observation as a crutch. Even some recent physicists have suggested, off the record at least, that this view deserves reconsideration, partly because it is hard to imagine how the numerical values of fundamental physical constants could be explained except in the manner we explain how π has the value it does.

I do not claim that nature reflects an underlying formal system. My point is that many of the founders of modern science saw (and some recent scientists see) nothing contradictory or wrong-headed about studying a formal system with empirical methods. So I think the onus is on those who see impropriety to defend their view.

My second response is more substantive. Studying algorithms at the level of a formal system presupposes a form of reductionism, which is the view that one can and should explain a phenomenon by reducing it to its ultimate constituents. Reductionism works in some contexts but fails miserably in others, and I think it often fails in algorithmic science.

Consider the theory of plate techtonics, which explains the formation of the continents by the way pieces of the earth's crust float on magma. One might attempt to explain the same phenomenon by reducing it to quantum physics, perhaps by formulating Schrödinger's equation for every atom in the earth and showing that our present topography corresponds to the simultaneous solution of the equations. But this approach presents two difficulties. There is the obvious practical problem that no mortal can carry it out. But even if we could do it, even if we could *deduce* earth's topography, this is no way to *explain* it—even in principle. We would get a computer printout of numbers that provide no insight whatever as to *why* continents look the way they do. The fundamental problem is not that there are too many numbers to absorb, but that even if we could absorb them, they would not tell an explanatory

story. Geologists, meanwhile, point out that the eastern coast of the Americas is shaped like the western coasts of Europe and Africa because they once collided, etc., and it all starts to make sense.

The key to explanation is obviously finding *the right level of analysis*—the level of floating plates rather than swarming atoms. Perhaps investigating the behavior of algorithms with formal methods is like applying quantum physics to geology. Even if one can in principle *deduce* what the algorithms are going to do, it is beyond human powers to do so, and even if we did, we would not *understand why* they behave as they do. Quantum mechanics is not a deductive science in the way that complexity theory is, but the situations seem otherwise analogous. (To take an analogy that does involve deductive science, one can ask: does the computer-implemented proof of the 4-color map lemma, granting that it *proves* the lemma, *explain* why the lemma is true?)

Steps in the Right Direction

The OR and computer science communities have taken two steps toward an empirical science of algorithms. Both are helpful, but both fall short of the goal. A Ph.D. thesis by Catherine McGeoch (1986a) nicely surveys much of the research in this area.

One step involves statistical methods. A few researchers have used rigorous methods of experimental design to concoct computational tests and statistical analysis to evaluate the results, and there seems to be a rising level of interest in these matters. One of the first efforts in the OR literature was a sophisticated application of experimental design principles by Lin and Rardin (1980). Golden and Stewart (1985), and more recently Amini and Racer (1992), used rigorous statistical methods to analyze test results. Eddy (1977) and Hart (1983) used less elaborate analyses. Barton (1987) discussed experimental design for comparing optimization procedures, and McGeoch (1992) variance reduction techniques. Crowder, Dembo and Mulvey (1978) as well as Hoaglin and Andrews (1975) addressed the issue of computational reporting. Miser (1993) discussed the role of empirical validation in operations research generally.

This movement toward rigorous methods is commendable, but it is only one ingredient of empirical science. Another key ingredient is the development of empirically-based theories that can be submitted to rigorous testing.

A second encouraging step toward an empirical science is the *heuristic* use of experimentation. By this I mean the practice of using experiments to suggest hypotheses about the behavior of algorithms. Bentley *et al.* (1983,1984) took this approach to bin packing algorithms and discovered theorems later proved by Shor (1984). Manacher and Zobrist (1983) took a similar approach to some matching problems, McGeoch (1986a,1986b) and McGeoch and Tygar (1991) to sorting, and Rivest (1976) to self-organizing search. Varga (1990) discussed the use of experimentation in mathematics generally, and a popularized exposition was the cover story in a recent *Scientific American* (Horgan 1993). There is even a new journal, *Experimental Mathematics*, founded at the University of Minnesota Geometry Center.

We all know that mathematicians have long used heuristic experimentation, but the positive aspect of the recent trend is that we are owning up to it in print. It falls short of empirical science because experimentation is used only to suggest theorems that may someday be proved. *Experimental* mathematics is not necessarily *empirical* mathematics, because in the latter experimentation not only suggests hypotheses but is the ultimate basis on which they are accepted or rejected. Yet heuristic mathematicians perform a valuable service by bringing mathematical experimentation out of the closet.

Empirical Science in Disguise

The empirical approach may seem more acceptable if we note the extent to which it has already crept into what is ostensibly our deductive science of algorithms. This has occurred in nothing less than one of the crowning achievements of "theoretical" (i.e., deductive) computer science: NP-completeness theory.

NP-completeness theory is, strictly speaking, a theory of problems rather than algorithms. But it bears importantly on algorithms because no known algorithm can solve any NP-complete

problem in polynomial time.

The reader can consult Garey and Johnson (1979) for an introduction to NP-completeness theory, but I will state the essentials briefly. Roughly speaking, a problem belongs to the class NP if it is possible to *verify*, in polynomial time, that a given solution for it is in fact a solution. ('Polynomial time' means that the time required increases no faster than some polynomial function of the problem size, which is measured by the number of bits needed to represent the problem in a computer.) We know that at least some of the problems in NP can in fact be *solved* in polynomial time. These comprise the class P, which is a subset of NP. A large variety of apparently hard problems also belong to NP.

A class of problems in NP is NP-complete if a polynomial-time algorithm for solving them (supposing such an algorithm exists) can be used to solve *any* problem in NP in polynomial time. Since no such algorithm has been found, an NP-complete problem class is regarded as *characteristically* hard in some sense, although everybody recognizes that it may contain easy problems among the hard ones.

The empirical content of NP-completeness theory becomes evident when we note that any problem class in NP that *contains* an NP-complete class is itself NP-complete on that basis alone. Consider, for instance the famous class TSP of traveling salesman problems. The standard proof of NP-completeness for TSP shows that one could solve any satisfiability problem in polynomial time if he could solve any problem in a certain subset T_0 of TSP in polynomial time. Since the class of satisfiability problems is NP-complete, it follows that TSP is NP-complete.

What interests us here is that this argument shows equally well that T_0 is NP-complete, where T_0 is a very small subset of very special traveling salesman problems. Let us grant that this shows that the problems in T_0 are characteristically hard in some sense. But on what ground do we infer that the much larger superset TSP is a class of hard problems?

Recall that *any* class of problems in NP that contains T_0 is *ipso facto* NP-complete. Consider the class P$'$ that consists of all the problems in P (i.e., all problems soluble in polynomial

time) and the very special traveling salesman problems in T_0. P$'$ is no less NP-complete than TSP, because it contains T_0. But it seems odd to say that the problems in P$'$ are characteristically hard, since they include all the easy problems in the world.

What, then, gives us the right to say that TSP consists of characteristically hard problems? Perhaps it is the fact that all the problems in TSP share a structural similarity with the problems in T_0. But this only postpones the question: on what ground do we say that the TSP structure, rather than some other structure shared by problems in T_0, makes T_0 hard?

I suggest that we regard TSP as a hard class because *we in fact find problems in TSP to be hard in practice.* Again, we acknowledge all along that TSP contains many easy problems. But when one generates larger and larger TSP problems according to almost any reasonable scheme that is not designed to produce easy problems, their difficulty tends to explode. It is this fact, I submit, that justifies our saying that TSP contains characteristically hard problems. It the fact that, to a large degree, makes the NP-completeness result for TSP *interesting.* And it is an empirical fact.

TSP is what philosophers of science sometimes call a natural kind. To use Goodman's (1965) well-known example, let us define 'grue' to mean 'green until 2000 A.D. and blue thereafter.' Observing a large number of emeralds justifies a generalization that emeralds are green. But the same observation, even if it takes place before 2000, fails to justify a generalization that emeralds are grue. The reason is that 'green' is a natural kind where emeralds are concerned and 'grue' is not. We have found empirically that properties like green relate to the order of nature in a way that properties like grue do not. Similarly, TSP appears to be a natural kind where problem difficulty is concerned, whereas P$'$ is not. Again, we found out about this empirically.

Some Examples

To make my proposal for empirical algorithmic science more concrete, I will describe some examples from recent work. None of these efforts fully exemplify the empirical approach, and

I know of none that do. But by pointing out what is missing in each case, perhaps I can more clearly indicate the goal I have in mind.

NP-completeness Theory

I have already argued that there is something covertly empirical about NP-completeness theory. I will try to evaluate it as an overtly empirical theory.

The basic law of empirical NP-completeness theory is that NP-complete problem classes tend to contain hard problems. From this one can deduce testable predictions: that certain natural problem classes are characteristically hard because they are NP-complete. The theory's main strength is the richness and subtlety of the methods with which NP-completeness proofs are constructed. Their ability to make fine distinctions and derive deep and surprising results is reminiscent of the best empirical theories.

The main weakness of the theory, however, is its lack of explanatory power. It says little or nothing to justify its natural kinds. Physical theory, taken as a whole, makes it seem reasonable that green should be a natural kind and grue not one. Although it seems equally reasonable to regard traveling salesman problems as a natural class and P' (defined earlier) as an unnatural one, nothing in NP-completeness theory or even complexity theory as a whole seems to warrant these judgments.

Beyond this is the obvious vagueness in the notion of a characteristically hard problem class. This makes it difficult to design experiments to check the theory's predictions.

Local Search

Gent and Walsh's (1993) recent study of a local search heuristic for the satisfiability problem has some elements of empirical science.

The satisfiability problem is to determine whether a given set of formulas of propositional

logic can be true simultaneously. An example might be,

x_1 or not-x_2

not-x_1 or x_3

not-x_3

The problem is to decide whether there is an assignment of truth values (true or false) to the atomic propositions x_1, x_2, x_3 that makes all the of the above formulas true. (In this case, there is.) A 3-SAT problem is one in which there are exactly three terms (*literals*) in each formula.

Gent and Walsh attacked this problem by trying to find an assignment of truth values that maximizes the number of true formulas—that is, by solving MAXSAT, the maximum satisfiability problem. They used a local search heuristic, which operates on a network of solutions (truth value assignments) in which each node is connected by arcs to *adjacent* solutions. Each adjacent solution is obtained by reversing the truth value of one atomic proposition.

The heuristic starts at some node and moves to the adjacent node that satisfies the most formulas. When there is a tie, it can pick a node randomly. This step is repeated until all the adjacent solutions are no better than the current one (which is therefore a *local maximum*), or until a predetermined number of iterations are completed. (Actually, Gent and Walsh permitted the algorithm to move to a worse adjacent solution if none were equal or better, but this rarely happened.) If the procedure finds a solution that satisfies all the formulas, the satisfiability problem is solved. Otherwise the problem remains unsolved, because there may be such a solution that was not found.

Gent and Walsh solved a large set of randomly-generated 3-SAT problems in which the number M of formulas is a fixed multiple of the number N of atomic propositions. They plotted the solution value against the iteration number, expressed as a multiple of N. Remarkably, they found that the plot is almost identical from problem to problem, even for different values of N. Furthermore, the solution value curve shows a clear transition from a hill-climbing phase to a plateau phase. They also plotted, against the same abscissa, the number of adjacent solutions tying for the best. This curve makes a corresponding transition from a sawtooth shape to a

plateau. Each tooth of the sawtooth portion is twice as big as the previous tooth. Gent and Walsh used regression analysis to quantify these relationships.

Remarkable as they are, these results do not comprise the sort of theory we seek because they are purely descriptive and offer no explanation. But one is reminded of Bohr's early theory of the atom, which reflected some remarkable relationships that were consistent with observation but were only later explained by quantum mechanics. Gent and Walsh's results may similarly be the first step toward an empirical theory. It is perhaps more plausible, however, that they reflect a probabilistic theorem that someone may eventually prove, as Gent and Walsh themselves suggest. In this case the empirical work is heuristic rather than theoretical, in the sense discussed earlier.

This work may violate the spirit of empirical theory in another respect. I suggested earlier that algorithmic performance is ideally predicted as a function of problem characteristics, so as to avoid the issue of how the problems are generated. Here performance is described as a function of only one parameter (the ratio M/N) and may rely heavily on the problem distribution Gent and Walsh used.

Linear Programming Decomposition

Bogetoft, Ming and Tind (1992) used empirical methods to study the behavior of a mathematical programming algorithm. Their work illustrates, in a very rudimentary way, two types of explanatory empirical theories.

This work uses a variation of Dantzig-Wolfe decomposition to study the extent to which decentralized decision making is possible when the decision makers have different preferences. The object is to get the headquarters and subsidiaries of a firm to agree on how to operate the firm even though they differ somewhat on what is important. One obvious way is for headquarters simply to dictate policy. But in a large firm where it is more efficient to distribute decision making authority, headquarters might concern itself with the constraints that involve the big picture, whereas the subsidiaries worry about constraints that are relevant locally. Headquarters gives the subsidiaries an incentive to take the big picture into account by

appropriately adjusting the price of resources they obtain through headquarters. If this does not work, headquarters looks at what the subsidiaries are doing and adjusts incentives so as to correct their course.

The classical theory of Dantzig-Wolfe decomposition states that, after a finite number of adjustments, the behavior of the divisions will be optimal for the firm as a whole, *if they have the same preferences* (i.e., the same objective function). But what happens when there is not full agreement on preferences? Bogetoft *et al.* found empirically that the resulting decision is remarkably close to being acceptable to all parties even when they have substantially different preferences.

To understand this work properly, it is necessary to look at the mathematical programming formulation. It is a multiple-objective linear programming problem:

$$\begin{aligned} \max \quad & \mu C x \\ \text{subject to} \quad & Ax \leq a \\ & Bx \leq b \\ & x \geq 0 \end{aligned} \tag{1}$$

Each row of the matrix C corresponds to one of the objective functions, and μ is the vector of weights that headquarters assigns to the several objectives. These weights reflect the preferences of the decision maker. Headquarters takes into account the "big picture" constraints $Ax \leq a$, which can be regarded as constraints on resources that the divisions must obtain through headquarters. The divisions concern themselves with the constraints $Bx \leq b$.

The decentralized decision problem is formulated as a master problem to be solved by headquarters and a subproblem to be solved by subsidiaries. The master problem is,

$$\begin{aligned} \max \quad & \sum_{i \in I} C y_i \lambda_i \\ \text{subject to} \quad & \sum_{i \in I} A y_i \lambda_i \leq a \\ & \sum_{i \in I} \lambda_i = 1 \\ & \lambda_i \geq 0, i \in I, \end{aligned}$$

and the subproblem is,

$$\begin{aligned} \max \quad & (\nu C - uA)x \\ \text{subject to} \quad & Bx \leq b \\ & x \geq 0. \end{aligned}$$

In general the subproblem decomposes into several problems, one for each subsidiary, but we can suppose for simplicity that there is but one subsidiary. Note that the subsidiary may have different preferences than headquarters, because it may assign a different vector ν of weights to the multiple objectives. When the weights are the same ($\mu = \nu$), we have the classical Dantzig-Wolfe decomposition.

Each column y_i of the master problem is an extreme point (or extreme ray) of the subproblem's feasible set. It represents one way the corporation can be run that is consistent with the constraints $Bx \leq b$ imposed on the subsidiary. All other feasible ways of running the corporation correspond to convex combinations of the y_i's, using weights indicated by the variables λ_i. The master problem's constraints prohibit combinations that exceed the resource limits.

The decision process goes as follows. Headquarters makes an initial decision by solving the master problem with only a small subset I of the its columns. It then charges the subsidiary a price for each resource, as given by the optimal value of the vector u of dual variables corresponding to the resource constraints $Ax \leq a$. The subsidiary uses these prices to make its own decision, using its own set of preferences, by solving the subproblem. This decision is given by an extreme point y_i of the feasible set, which is added to the master problem. If this new column allows headquarters to obtain a better solution, the process repeats, and otherwise it terminates with an equilibrium decision. Termination always occurs after finitely many iterations.

The authors generated random problems governed by six parameters, including a) the number of variables, master problem constraints, subproblem constraints, and objectives, as well as b) parameters indicating how much the objectives differ and how much the headquarters

preferences μ differ from the subsidiary preferences ν. They measured the mutual acceptability of the resulting decisions by noting how often they are efficient (i.e., pareto-optimal) solutions of the original problem (1). (They used other measures as well.) They found that about 90% of the solutions are efficient even when the weights μ and ν are generated independently, rising to 100% as μ and ν become more similar.

More interesting for our purposes is the further discovery that the decisions are more likely to be efficient when a greater portion of the constraints are in the subproblem. Roughly put, collaborative decision making is easier when subordinates have more information than the boss rather than vice-versa, even when they disagree with the boss.

To analyze this study we must realize that it constructs two empirical theories. One is a theory of hierarchical decision making based on a linear programming model and algorithm. I just noted one consequence of the theory, which can be tested empirically.

The second theory, which is the one that interests us, addresses the behavior of the algorithm used in the first theory. Unlike Gent and Walsh, the authors try to explain as well as describe their results. Actually they offer two quite different if very rudimentary explanations, each of which may be the germ of an empirical theory. One is that if the subsidiary "has fewer constraints, there are also fewer expreme points from which to select. In this case there is less chance of obtaining an efficient solution" (p. 19). This could be interpreted as an idea for a probabilistic proof, in which case the empirical work here is only heuristic. But suppose this idea is elaborated into a probabilistic *rationale* that does not reach the status of proof, perhaps because a rigorous proof is too hard to construct. This might be done by making certain simplifying assumptions or approximations in the argument that are not mathematically justified. The resulting theory would be truly empirical because the ultimate test of truth is whether it squares with computational testing. Nonetheless it is not wholly in the spirit of empirical theorizing, because it explains phenomena with concepts associated with same type of reductive analysis (probability theory) that a deductive treatment might use.

The authors' other explanation goes as follows: "When relatively few constraints are present

at the central level then relatively more information is present in the subunit [subsidiary]. Hence the subunit improves its possibilities to generate the right proposals to be considered." This is an embryonic but substantial advance in empirical theorizing, because the master and subproblem are viewed as *decision making units* with access to a certain amount of *information*, rather than pieces of mathematics to be analyzed in a theorem-proof mode. Indeed the real-world pheonomenon the algorithm is designed to model provides the concepts for explaining the algorithm's behavior! The explanation is too vague in its present form to have much predictive power. But when one attempts to understand intuitively what is going on here, he is almost irresistably drawn to an explanation of this kind. It may therefore form the basis for a useful empirical theory.

Satisfiability Algorithms

Another empirical study of satisfiability algorithms (Hooker and Vinay 1993) develops somewhat further the notion of explanation via probabilistic rationale. It deals with branching algorithms, which are among the more promising methods for solving the satisfiability problem (see Harche *et al.* 1993, Mitterreiter and Radermacher 1993). These methods operate by fixing a chosen variable to true and then to false, and in either case attacking the resulting subproblem in the same way. Critical to their success is the branching rule, i.e., the way the variable to be fixed is chosen. Despite a number of deep results in the probabilistic analysis of satisfiability algorithms (e.g., Franco 1986, Purdom 1990, and many other papers), we cannot explain why the most effective algorithms work and predict when they will work.

Hooker and Vinay propose two simple probabilistic models of how branching rules affect the performance of algorithms. The matter is too involved to describe in detail here, but briefly, one model says that branching rules work well when they try to create subproblems that are satisfiable with high probability. This is the justification given by Jeroslow and Wang (1990) for their very effective branching rule. The other model says that an effective branching rule should try to create subproblems that result in the generation of a large number of unit clauses (formulas containing just one literal) further down the search tree.

Both models make simplifying approximations and abandon the possibility of proving a theorem. But both make specific predictions that can be tested empirically. Computational experience soundly refutes the first model and tends to confirm the second (to the extent that empirical confirmation is possible; see Popper 1965). In particular, the original justification for the Jeroslow-Wang rule is wrong, and the alternative model suggests a new branching rule that is apparently superior to other known rules.

This study falls short of the empirical ideal in at least two respects, however. It relies on a set of benchmark problems (albeit a very large one), and it reverts to probabilistic concepts that would be used in a deductive analysis.

Simulated Annealing

Fleischer and Jacobson (1992, 1993) used concepts from information theory to try to explain the performance of simulated annealing heuristics. Their work shows that sophisticated theoretical analysis can form the basis for an empirical theory.

Simulated annealing heuristics, which derive ultimately from the work of Metropolis *et al.* (1953), are similar to local search heuristics in that they repeatedly move to an adjacent solution (see Aarts and Korst 1989 for a detailed treatment). They differ, however, in that they choose a solution randomly from adjacent solutions and move to this solution if it is no worse than the current one. If the solution is worse, the move is nonetheless made with probability $e^{-\Delta/T}$, where Δ is how much worse it is, and T is the "temperature." The process is repeated as long as one desires, while the temperature is gradually reduced. Mitra *et al.* (1986) showed that for a properly chosen cooling schedule, the probability that the heuristic will reach an optimal solution in any given move approaches one aymptotically as the number of iterations increases.

The simulated annealing process can be modeled with a (strongly ergodic) inhomogeneous Markov chain. At every step of the algorithm there is a matrix P whose elements p_{ij} give the probability that the algorithm will move to solution j if it is now at solution i. The chain is inhomogeneous because P changes with each iteration, due to the changing temperature.

Fleischer and Jacobson begin with a known connection between the entropy H of a homogeneous (constant temperature) Markov chain and its behavior. Namely, the number of possible sequences of n steps that are "typical" sequences is proportional to e^{nH}. A typical sequence of steps is one in which each solution occurs with a relative frequency that is close to the steady-state probability of that solution if the algorithm were to run forever. The entropy of a chain is a measure of the average amount of information obtained when it moves one step. More precisely, the entropy of each row i of P is $-\sum_j p_{ij} \ln p_{ij}$, and the entropy of P and therefore the chain is an average of the row entropies, weighted by the steady-probabilities of reaching the corresponding solutions i in any given step. Thus a high-entropy chain is less likely to deviate from typical behavior.

When the temperature falls appropriately and the algorithm runs long enough, a typical sequence of steps is one that reaches mostly optimal solutions (because of the result of Mitra *et al.* that optimal solutions become increasingly probable). So if the idea of entropy could be extended to inhomogeneous chains, high-entropy chains ought to reach optimal solutions with greater probability. Fleischer and Jacobson do essentially this by defining the entropy H^* of a sequence of steps to be the sum of the entropies of the matrices P used along the way. They show that this measure bounds below and asymptotically approximates a related but uncomputable entropy measure that relates to performance in the way described.

The theory predicts that, other things equal, simulated annealing algorithms with high H^* ought to perform better in a given number of steps. But empirical confirmation is complicated by the fact that other things are not equal; high-entropy chains tend to gather less information in each step and may therefore run longer before generating typical (i.e., near-optimal) sequences of steps. Fleischer and Jacobson qualitatively analyze this effect, but not in a way that makes specific numerical predictions. They do find experimentally that higher entropy chains obtain better solutions, but it is unclear to what extent this confirms the theory.

Despite problems of confirmation, the theory links a measurable trait of an annealing algorithm—the entropy of the associated Markov chain—with the algorithm's performance.

The entropy can be computed exactly only for very small problems, but it can conceivably be estimated for large problems by sampling or by partially analyzing the probability matrix P. It therefore represents an interesting advance in empirical theory.

Tabu Search

My final example is more in the way of a research proposal. It concerns the tabu search heuristic, sometimes called a metaheuristic because it can be realized in so many ways. Tabu heuristics have met with considerable success in solving hard combinatorial problems, but there is little or no theoretical understanding of when or why they work well. See Glover (1989,1990a,1990b) for a thorough discussion of tabu heuristics.

Like simulated annealing, tabu search in its most basic form is a local search with one additional wrinkle. When the procedure reaches a local minimum, it keeps going to the best adjacent solution, even if it is worse than the current one. The process is repeated a predetermined number of iterations, keeping track of the best solution found so far.

A *tabu list* provides the heuristic with short term memory, to prevent cycling through the same solutions over and over. It is a list of the *moves* that have been made recently, where a move is a way that one alters a solution to get to an adjacent one. Any move that reverses a move on the list is tabu (more often spelled 'taboo' in other contexts), meaning that it cannot be used. Tabu searches can also use long-term memory based on frequency, etc.

Consider again the network of solutions, where each node is connected to adjacent solutions. The motivating idea for the model I propose is to view the nodes as points on an undulating landscape, so that nodes connected by arcs are close in some spatial sense. The elevation of each node is the cost of the corresponding solution, and the object is to find a point of low elevation.

The tabu search operates by finding a steepest possible downhill hill path until the bottom of a basin (a local minimum) is reached. At this point the heuristic climbs out of the basin until it reaches the rim, whereupon it descends into another basin, on the chance it contains a lower point. The tabu list is supposed to prevent the search from falling back down into a

basin before it climbs out. (The *basin* surrounding a local minimum is the set of nodes from which some local search leads to the local minimum. The *rim* of the basin is the set of its nodes that are adjacent to at least one node outside the basin.)

Tabu search can be interpreted as a form of implicit enumeration. Once the search descends into a basin and finds a local minimum, there is no point in visiting any of the other nodes in the basin (except to climb out again). The unvisited nodes are implicitly enumerated. My theory states that the performance of the heuristic can be explained by the efficiency of this implicit enumeration. If the heuristic visits only a small fraction of the nodes in a basin, it should in effect enumerate a large number of solutions in a short time and therefore perform well.

We can take the length of a path in the network to be the number of nodes on it. The distance between two nodes is the length of a shortest path between them. Given these definitions, we should expect the tabu heuristic to visit a small fraction of the nodes in a basin when the number of nodes in the basin is large relative to the distance from the rim to the local minimum. Let this distance be the radius of the basin, defined as the mean distance from rim nodes to the local minimum. (If there are multiple local minima, we can take the one resulting in the smallest mean.)

We can now propose a hypothesis, to be tested empirically:

- Tabu search works better when the ratio of average basin size to average radius (which we call the *search ratio*) is large.

Intuitively we can think of the search ratio as being large when the network is embedded in a higher dimensional space, since the volume-to-radius ratio for spheres and other bodies increases as the number of dimensions increases. So tabu search should work better in higher-dimensional spaces, albeit this idea of dimensionality is not formally part of the model.

This little theory is useful only if conjoined with a theory of how the basin characteristics depend on the problem type, so that one can predict performance for a particular problem. Ryan (1993a, 1993b) has proved upper bounds on the size of basins in certain graph color-

ing, knapsack, and set covering problems. But I would propose empirical investigations that characterize the search ratio as a function of problem characteristics.

The theory also leaves much room for elaboration. Problem landscapes have many features other than the size and diameter of basins, and these may help explain tabu performance. The landscape may consist essentially of a plain that is pockmarked with small basins, or of large hills and valleys that are similarly indented. Tabu search may find better solutions on the latter type of manifold because it tends to spill out of each basin it visits into a lower one.

Other Possibilities

The empirical theories I have surveyed are not only embryonic but also conservative. There is room for a good deal more imagination. For instance, empirical theories often find it useful to postulate an internal structure for the objects under investigation, as Rutherford suggested an orbital model of the atom to explain its scattering of Xrays. Perhaps this can be done for algorithms. The postulated structure, however, need have no relation to the sequence of steps or subroutines in a statement of the algorithm. The steps pertain to a *formal* explanation of its behavior, which we are trying to avoid here. In fact it may be good policy to forget all about the formal structure of the algorithm, to free one's imagination to invent empirically-inspired explanations.

A biological level of explanation, for instance, may be useful, since algorithms can be viewed as organisms that act upon their environment. This is not as far-fetched as it sounds. Farmer and Belin (1990) argue that computer viruses (which are essentially algorithms) possess most and conceivably all of the formal characteristics of living organisms.

It is hard to be more specific, since one cannot propose interesting and plausible empirical theories on the first day of work. Empirical science takes time to build. I suggest we get started.

Acknowledgments

This work is partially supported by ONR grant N00014-92-J-1028. I thank the referees and several email correspondents for comments and information that helped me improve an earlier draft circulated electronically.

References

Aarts, E., and J. Korst 1989. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley, New York.

Amini, M. M., and M. Racer. 1992. A variable-depth-search heuristic for the generalized assignment problem, working paper, Civil Engineering, Memphis State University, Memphis, TN 38152 USA.

Barton, R. R.. 1987. Testing strategies for simulation optimization, in A. Thesen and H. Grant, eds., *Proceedings of the 1987 Winter Simulation Conference*, IEEE Press, New York, 391-401.

Bentley, J. L., D. S. Johnson, F. T. Leighton and C. C. McGeoch. 1983. An experimental study of bin packing, in *Proceedings, 21st Allerton Conference on Communications, Control and Computing*, University of Illinois, Urbana, IL, USA, 51-60.

Bentley, J. L., D. S. Johnson, F. T. Leighton, C. C. McGeoch and L. A. McGeoch. 1984. Some unexpected expected-behavior results for bin packing, in *Proceedings, 16th Symposium on Theory of Computation*, ACM, New York.

Bogetoft, P., C. Ming and J. Tind. 1992. Price-directed decision making in hierarchical systems with conflicting preferences, DASY paper no. 3/92, Institute of Computer and Systems Sciences, Copenhagen Business School, Julius Thomsens Plads 10, DK-1925 Fredericksberg C, Denmark.

Crowder, H. P., R. S. Dembo and J. M. Mulvey. 1978. Reporting computational experiments in mathematical programming, *Mathematical Programming* **15**, 316-329.

EDDY, W. F. 1977. A new convex hull algorithm for planar sets, *ACM Transactions on Mathematical Software* **3** 398-402.

FARMER, J. D., AND A. D'A. BELIN. 1990. Artificial life: The coming evolution, manuscript 90-003, Santa Fe Institute, 1120 Canyon Rd., Sante Fe, NM 87501 USA.

FLEISCHER, M. 1993. Assessing the performance of the simulating annealing algorithm using information theory, Ph.D. thesis, Operations Research Dept., Case Western Reserve University, Cleveland, Ohio USA 44106.

FLEISCHER, M., AND S. H. JACOBSON 1992. The entropy of inhomogeneous Markov chains with applications to simulated annealing, manuscript, Operations Research Dept., Case Western Reserve University, Cleveland, Ohio USA 44106.

FRANCO, J. 1986. On the probabilistic performance of algorithms for the satisfiability problem, *Information Processing Letters* **23** 103-106.

GAREY, M. R., AND D. S. JOHNSON. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco.

GENT, I. P., AND T. WALSH. 1993. An empirical analysis of search in GSAT, manuscript, Artificial Intelligence Dept., University of Edinburgh.

GLOVER, F. 1989. Tabu search—Part I, *ORSA Journal on Computing* **1**, 190-206.

GLOVER, F. 1990a. Tabu search—Part II, *ORSA Journal on Computing* **2**, 4-32.

GLOVER, F. 1990b. Tabu search: A tutorial, *Interfaces* **20** (4), 74-94.

GOLDEN, B. L., AND W. R. STEWART. 1985. Empirical analysis of heuristics, in Lawler, Lenstra, Rinnooy Kan and Schmoys, eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, pp. 207-249.

GOODMAN, N. 1965. *Fact, Fiction and Forecast, 2nd ed.*, Bobbs-Merrill, Indianapolis.

HARCHE, F., J. N. HOOKER AND G. THOMPSON 1993. A computational study of satisfiability algorithms for propositional logic, to appear in *ORSA Journal on Computing.*

HART, R. R. 1983. The average height of binary search trees, masters thesis, University of California at Irvine, Irvine, CA 92717 USA.a

HOAGLIN, D. C., AND D. F. ANDREWS. 1975. The reporting of computation-based results in statistics, *The American Statistician* **29**, 122-126.

HOOKER, J. N., AND V. VINAY 1993. Branching rules for satisfiability. Manuscript in preparation.

HORGAN, J. 1993. The death of proof, *Scientific American* **269** 92-103.

JEROSLOW, R., AND J. WANG 1990. Solving propositional satisfiability problems, *Annals of Mathematics and AI* **1** 167-187.

LIN, B. W., AND R. L. RARDIN. 1980. Controlled experimental design for statistical comparison of integer programming algorithms, *Management Science* **25** 1258-1271.

MANACHER, G. K., AND A. L. ZOBRIST. 1983. Probabilistic methods with heaps for fast-average-case greedy algorithms, in F. P. Preparata, ed., *Advances in Computing Research: Computational Geometry*, JAI Press, Greenwich, CT, 261-278.

MCGEOCH, C. C. 1986a. Experimental analysis of algorithms, Ph.D. thesis, CMU-CS-87-124, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

MCGEOCH, C. C. 1986b. An experimental study of median-selection in quicksort," *Proceedings of 24th Allerton Conference on Computing, Control and Communication*, University of Illinois, Urbana-Champaign, IL, USA, 19-28.

MCGEOCH, C. C. 1992. Analyzing algorithms by simulation: variance reduction techniques and simulation speedups, *Computing Surveys* **24** 195-212.

McGeoch, C. C., and D. Tygar 1991. Optimal sampling strategies for quicksort, *Proceedings of 28th Allerton Conference on Computing, Control and Communication*, University of Illinois, Urbana-Champaign, IL, USA, 62-71.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller 1953. Equation of state calculations by fast computing machines, *Journal of Chemical Physics* **21** 1087-1092.

Miser, H. J. 1993. A foundational concept of science appropriate for validation in operations research. *European Journal of Operations Research* **66** (1993) 204-215.

Mitra, D., F. Romeo and A. Sangiovanni-Vincentelli 1986. Convergence and finite-time behavior of simulated annealing, *Advances in Applied Probability* **18** 747-771.

Mitterreiter, I., and F. J. Radermacher 1993. Experiments on the running time behaviour of some algorithms solving propositional logical problems, to appear in *Annals of Operations Research.*

Popper, K. R. 1965. *Conjectures and Refutations*, New York, Harper and Row.

Purdom, P. 1990. A survey of average time analyses of satisfiability algorithms, *Journal of Information Processing* **13** 449-455.

Rivest, R. 1976. On self-organizing sequential search heuristics, *Communications of the ACM* **19**, 63-67.

Ryan, J. 1993a. The depth and width of local minima in discrete solution spaces, manuscript, Mathematics Dept., University of Colorado, Denver.

Ryan, J. 1993b. The depth and width of local minima for graph coloring problems, manuscript, Mathematics Dept., University of Colorado, Denver.

Shor, P. W. 1984. The average-case analysis of some on-line algorithms for bin packing, in *Proceedings, 25th Symposium on Foundations of Computer Science*, IEEE, 193-200.

TENENBAUM, A. 1978. Simulations of dynamic sequential search algorithms, *Communications of the ACM* **21**, 790-791.

VARGA, R. S. 1990. *Scientific Computation on Mathematical Problems and Conjectures,* Regional Conference Series in Applied Mathematics, v. 60, SIAM, Philadelphia.