

Changes of Mathematical State

Untangling a web of conflicting demands can be tough on computers

By IVARS PETERSON

The frazzled host of an impending dinner party is trying to come up with a seating plan that would satisfy his guests.

Angelina would like to sit beside Bertrand, but she can't stand Charles. Darva and Rick, who've recently split up, insist on being as far apart as possible. And Eve, whose belated acceptance has thrown the original seating plan into disarray, is willing to sit next to Bertrand or Charles but not Angelina.

Is there an arrangement that would please everyone? As the list of guests and their

various demands increases, the number of seating combinations that must be checked to avoid conflicts grows rapidly. Solving such a problem by trying out a vast number of different combinations can take years, even if the host enlists a powerful computer.

Sometimes the search for a solution is relatively quick, however. When guests impose few conditions, for example, many combinations meet the requirements. At the other extreme, guests may specify so many constraints that it's easy to recognize the problematic arrangements, and the host can then focus on the few that might work out.

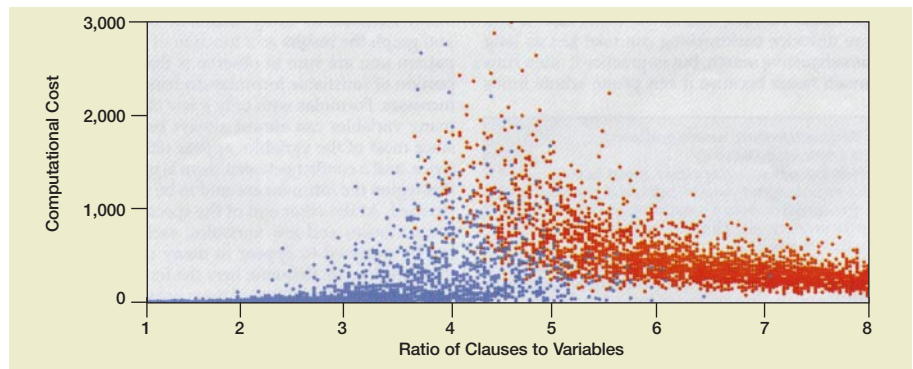
Of course, conflicting requests may make the problem impossible to solve. What if Bertrand refuses to sit next to Charles and Charles demands to sit next to Bertrand? Then, the quest is over as soon as the host realizes the demands are at odds and the problem is insoluble.

Researchers have discovered, to their amazement, that just as water abruptly freezes when the temperature dips below the freezing point, a computer-search problem can change very rapidly from soluble to insoluble. When for a given number of guests, the number of constraints exceeds a certain threshold, the host will realize that he might as well give up.

Interestingly, the hardest problems to deal with lie in the middle—when there are enough constraints to limit the number of good choices but not enough to immediately eliminate the bad ones or signal that the case is hopeless.

Researchers have found that a wide variety of search problems exhibit a transition phenomenon, such as from soluble to insoluble. Indeed, the analogy between rapid shifts in computational behavior and physical changes of state has inspired mathematicians, computer scientists, physicists, and others to explore physical models to obtain insights into precisely what makes many important computational problems hard to solve.

"Understanding intractability is one of the fundamental problems in computer science," says Jennifer Tour Chayes of Microsoft Research in Redmond, Wash. She described the current



AMERICAN SCIENTIST

Each point on this graph shows the relative computational effort required to solve a given satisfiability problem or show that it's insoluble. For so-called 3-SAT problems, researchers find a sharp transition between problems that are solvable (blue dots) and those that turn out to be impossible to solve (red dots) when the ratio of constraints to variables is about 4.25 and the computational effort is highest.

state of research on computational “phase transitions” earlier this year at the annual meeting of the American Association for the Advancement of Science in Washington, D.C. Concepts from physics could help elucidate the source of such difficulties, she notes.

“There are some properties of these abrupt phase transitions that we may be able to exploit,” remarks computer scientist Bart Selman of Cornell University. Such insights could lead to improved procedures, or algorithms, for solving challenging computational problems, whether they involve searching for optimal seating plans, developing airline schedules, or laying out complex circuitry on computer chips.

Working out a seating plan that would placate fractious dinner guests is a version of the logic puzzle in theoretical computer science known as the satisfiability (SAT) problem.

Computer scientists can translate such constraint-satisfaction problems into symbolic logic, where each variable is either true or false. Consider, for example, two variables, x and y , and the logical statement $(x \text{ OR } y) \text{ AND } ((\text{NOT } x) \text{ OR } (\text{NOT } y))$. The OR means that the clause $(x \text{ OR } y)$ is true if either x or y is true, and the AND means that the clause $(x \text{ AND } y)$ is true only if both x and y are true.

Solving this puzzle requires assigning a value of true or false to each of the variables so that the entire expression is satisfied—meaning that the statement must evaluate to true. Here, x must be true and y false, or vice versa.

Computer scientists rate the difficulty of problems by how long it takes a computer to solve them in the worst possible case. Satisfiability problems (see box) that involve expressions with only two variables in each clause (termed 2-SAT) can be solved in what is called polynomial time. In other words, as the number, n , of variables increases in the entire statement, the solution time grows modestly at a rate proportional to some power of n . In effect, computers can solve 2-SAT problems efficiently.

In contrast, when clauses each contain three variables, increasing the total number of variables means that the solution time grows exponentially in the worst case. Hence, systematically solving such a 3-SAT problem involving many variables can take an enormous amount of time.

The focus on worst cases, however, can be misleading. Researchers know from experience that solution times vary tremendously from case to case and depend on the specific search algorithm used to find an answer. Many search problems take much less time to solve than worst-case scenarios would suggest.

Hence, “it makes a lot of sense to study typical cases,” Chayes says.

In earlier work on randomly generated instances of the 3-SAT problem, Selman, Scott Kirkpatrick of the IBM Thomas J. Watson Research Center in Yorktown Heights, N.Y., and their coworkers learned that, on average, the hardest problems occur when the ratio of the number of clauses to the number of variables is about 4.25. Below this critical value, nearly all cases are satisfiable, and above that value, almost all of them are unsatisfiable.

Logic Puzzles and Satisfiability

- Two variables (x, y), two clauses, two variables per clause (2-SAT):

$$\underbrace{(x \text{ OR } y)}_{\text{Clause 1}} \text{ AND } \underbrace{((\text{NOT } x) \text{ OR } (\text{NOT } y))}_{\text{Clause 2}}$$

This formula is satisfiable if x is true and y is false, or vice versa.

- Three variables (x, y, z), two clauses, two variables per clause (2-SAT):

$$(x \text{ OR } y) \text{ AND } ((\text{NOT } x) \text{ OR } (\text{NOT } z))$$

Out of eight possible assignments of true and false to the three variables, four satisfy the formula.

- Three variables (x, y, z), three clauses, two variables per clause (2-SAT):

$$(x \text{ OR } y) \text{ AND } ((\text{NOT } x) \text{ OR } (\text{NOT } z)) \text{ AND } ((\text{NOT } x) \text{ OR } z)$$

Adding clauses generally reduces the number of assignments that satisfy the resulting formula. In this case, only two out of eight possibilities satisfy the formula.

- Three variables, two clauses, three variables per clause (3-SAT):

$$(x \text{ OR } y \text{ OR } z) \text{ AND } ((\text{NOT } x) \text{ OR } (\text{NOT } y) \text{ OR } z)$$

Six out of eight possible assignments of true and false to the three variables satisfy the formula.

For 2-SAT problems, the threshold occurs when the ratio of clauses to variables is 1.

These results fit the observation that hard problems cluster near the boundary between satisfiability and unsatisfiability, where it's difficult to tell whether solutions exist or not. Moreover, such a concentration of hard cases occurs at the same critical value for a wide range of search methods, says Tad Hogg of the Xerox Palo Alto (Calif.) Research Center. In other words, "this behavior is a property of the problems rather than of the details of the search algorithm," he says.

Not only do these dramatic changes in computational difficulty resemble the abrupt changes in a physical system's state at a certain parameter value, they also mirror a distinction made by physicists.

Physical phase transitions come in two flavors. In a first-order transition, such as the freezing of water when chilled, the jump from one state to the other is discontinuous. In a second-order transition, such as the demagnetization of iron when heated, the change is continuous, exhibiting a gradual change in behavior rather than a sharp break.

Remarkably, computational experiments by Selman and his coworkers recently showed that in 2-SAT problems, the transition from satisfiable to unsatisfiable is continuous. In 3-SAT problems, the transition is much more abrupt.

Selman and his colleagues are definitely onto something in highlighting the types of phase transition, Chayes says. However, "it's not that simple," she contends. "There are exceptions."

Selman admits, "There are subtle issues here." For example, the discontinuous phase transition found for the 3-SAT case is itself quite complicated. "We are still studying further properties of this transition," he notes.

"The empirical results are interesting and, if done well, can lead to insights which we can then try to verify rigorously," comments mathematician Alan M. Frieze of Carnegie Mellon University in Pittsburgh.

Mathematicians and computer scientists, for example, have shown theoretically that the threshold for the 3-SAT phase transition must lie between 3.2 and 4.5, and new efforts are gradually narrowing that range. This is consistent with Selman's empirical finding.

Theoretical constructs called spin glasses, which serve as simple models of magnetic systems, show an analogous phase behavior, Selman and others have demonstrated. This model starts with an array of magnetic particles, each one oriented either up or down (equivalent to variables that are true or false). A particle's orientation can flip from one state to the other with a certain probability, and each orientation influences neighboring orientations. Getting this so-called spin glass to settle into a lowest-energy state, in which all the particles comfortably coexist without further flipping, is equivalent to solving a satisfiability problem.

Although often a solution arises quickly, computer simulations show that difficulties arise when particles happen to settle into certain configurations. Patterns can get frozen prematurely, and then the spin glass as a whole can't readily reach the state with the lowest energy.

In search problems, similar behavior could occur if a computer making random choices locks in some of the critical variables with the wrong values, then spends an enormous time fruitlessly trying combinations involving those values.

Chayes and her coworkers have investigated similar behavior in a physical model known as the Potts magnet. Unlike a spin-glass particle, a Potts particle can take on any one of three or more different values. This simple change greatly complicates the theory, and the resulting behavior can be very complex. In this case, knowing the type of phase transition by itself isn't enough to account for the model's behavior, Chayes says.

By getting a better understanding of how such complexities arise, especially near a phase transition, researchers can look for ways to weed out hopeless cases and focus their efforts on solving the rest.

“We have found that there are efficient heuristic methods that have a good chance of identifying those [troublesome] variables early on in a search,” Selman says.

In airline scheduling, for example, some connections may pose special difficulties. One might first want to schedule the main flights between hubs, then later fill in the shorter commuter flights to and from the hubs, Selman suggests. “If you get certain destinations right, the rest of the problem is easy,” he says.

Selman, Kirkpatrick, and Cornell’s Carla P. Gomes have just started a major effort to employ their theoretical results to develop a computational system that adapts to the difficulty of the problem at hand.

To obtain an estimate of the expected complexity of a computational task, they identify where a problem lies with respect to the relevant phase transition, Selman says. If the problem has characteristics that put it at the threshold, the user may be able to relax the constraints to shift the problem away from the transition, into a realm where it is more easily soluble.

In practical terms, this could mean insisting that the airplanes always land in Chicago, no matter where else they go or how many people they carry, simply because the rest of the scheduling problem then becomes much easier.

“There is still quite a bit of work to be done before we can apply the insights developed for SAT to hard real-world computational problems,” Selman says. “In any case, the results on phase transitions and complexity have given us a new set of tools for attacking such computational challenges.”

Satisfiability problems serve as important test cases in such studies because they share crucial characteristics with thousands of other problems now known to be computationally difficult. What is learned about the behavior of satisfiability problems can then be applied to a variety of other thorny computer problems.

For example, the idea that particularly hard-to-solve instances of a problem occur at a computational phase transition may prove helpful in developing schemes for encrypting data. Moving toward a phase transition would make a code particularly tough to crack.

Nonetheless, “while the results are encouraging, a major open question is how the results for random [satisfiability] problems relate to those encountered in practice,” Hogg emphasizes. In real-world problems, for instance, correlations among the constraints can shift the location of the transition point and throw off predictions based on randomly chosen instances.

Fruitful interactions among theory, experiment, and application continue to drive the study of computational phase transitions.

“Looking at a problem from a new perspective can add new insight,” says Carnegie Mellon’s Lenore Blum. “The idea of introducing the concept of phase transitions into [computational] complexity has this potential. Even if it does not characterize complexity, figuring out to what extent it may or may not be relevant would be extremely interesting.” □