Discrete Dynamics in Nature and Society, Vol. 6, pp. 145–146
Reprints available directly from the publisher
Photocopying permitted by license only

Book Review

Complexity and Real Computation, By Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. Springer-Verlag, New York, 1998. xvi+453 pages. Includes subject index. $39.95.

The geometric viewpoint has served to unify Western science for millenia. Although originally conceived as a faithful model of the sensible world, geometry continually conquers new ground because it gives us a way to apply spatial intuition to abstract problems. Indeed, one of the major tasks of twentieth century geometry was the extension of geometric ideas to more and more abstract situations. An important aspect of this was the extension of algebraic geometry, which studies solutions of polynomial equations, to encompass equations over more or less arbitrary domains. This was undertaken systematically by van der Waerden, Weil, and their followers, culminating in a complete reworking of the subject during the 1950's.

Let us fast-forward to the 1990's and imagine that these researchers are reborn as theoretical computer scientists. They would look around them and see geometry used to prove lower bounds on the difficulty of computational problems. For example, if the set of inputs for which a certain decision should be affirmative has many connected components, any algorithm to make this decision must involve a lot of branching. They would also see attempts to compare the complexity of different problems. But to make such comparisons, one must express these problems in the same way, for example, by encoding all inputs as integers. It then makes sense to ask whether it is more difficult to sort a list of numbers, or to see which of them are prime.

Our newly minted computation theorists would realize, of course, that casting all problems into integer form is artificial in many cases, including most of those to which geometric arguments most naturally apply. They would wish for a theoretical treatment of computation that applies equally well to problems about real numbers, complex numbers, rational numbers, integers, or even denizens of the more exotic structures of abstract algebra.

This book grants their wish.

Let me now describe its technical contributions in more detail. Central to any theoretical discussion of computation is a precise definition of "algorithm". The book provides this in the following way. Suppose the inputs to some computation come from a commutative ring R. An algorithm – here called a *machine* – is essentially a Turing machine on whose tape cells are written elements of R. Already this is enough to allow interesting things to be said. For example, there is no algorithm to decide if the sequence z, z^2+z, $(z^2+z)^2+z, \ldots$ remains bounded. Of course we should not only discuss the existence of algorithms but also their cost. To allow this, the

authors assume that R is given a notion of size – here called *height* – and then define the cost of a computation to be the number of steps times the maximum, over all times, of the height of intermediate results.

The book then discusses generalizations of the famous *P vs. NP* problem. This problem, which asks in essence whether checking a proof is easier than finding one, is central to theoretical computer science. The new framework allows one to define the classes P and NP for any R, recovering the ordinary classes by taking R to the integers mod 2. Major results here include the exhibition of *NP*-complete problems for $\mathbf{C}$ and $\mathbf{R}$ (consistency of multivariate polynomial equations and of one degree 4 equation, respectively), and a version of Lefschetz's principle, stating that the *P vs. NP* question will have the same answer for every algebraically closed field of characteristic zero.

The middle portion of the book discusses the difficulty of solving linear and nonlinear equations. Although it assumes more mathematical knowledge than the rest of the book (mainly differential geometry and topology), it can pretty much be read on its own. It takes a viewpoint closer to numerical analysis than theoretical computer science. The following result is typical of the menu. Define, as is usual, the *condition number* of a square matrix A to be $\mathrm{cond}(A) = ||A||\,||A^{-1}||$, using the operator norm. If b is determined to m bits of precision and we solve $Ax = b$, then we cannot hope to know more than $m - \log_2 \mathrm{cond}(A)$ bits of x. How much precision is lost for an "average" problem? The authors prove that if the n^2 entries of A are i.i.d. standard normal (mean 0, variance 1), the expected value of $\log_2 \mathrm{cond}(A)$ is $O(\log n)$.

The final part of the book studies, for the special case where R is the real numbers, analogs of some topics of recent interest in discrete complexity theory. These include randomization, parallel computation, algorithms restricted in their operations (*e.g.*, addition only), and using mathematical logic to define complexity classes.

Of course, no book review is complete without a quibble or two, so here are mine. Although certainly sufficient for issues related to polynomial time, the book's cost model (number of steps times cost of the most expensive step) can give a misleading idea of algorithm performance. The formal treatment of computation, from the point of view of dynamical systems, may also be difficult for computer scientists to digest. Finally, although the book discusses computation over arbitrary commutative rings, the relevant algebra (except for the Nullstellensatz) plays a surprisingly small role. One suspects that the model is not yet in final form (why not computation over groups?), there is room to apply more commutative algebra, or both.

Nevertheless, the book is readable (graduate students will get through it), authoritative (the authors invented the area, and Smale is a world-class geometer to boot), interesting, and timely. Supplanted by a few readings in discrete complexity theory and the necessary exercises, it would be a great book for a graduate course in theoretical computer science. Every mathematics or engineering library should have it. The authors are to be commended for taking the time to explain their recent research to a wider audience.

Eric Bach,
Computer Sciences Department,
University of Wisconsin – Madison,
1210 W. Dayton St.,
Madison, WI 53706.
e-mail: bach@cs.wisc.edu