Position statement for EUROCAL 85

Computing with Geometric Objects

J. Nievergelt
Informatik, ETH, Zurich

The notion of "computing" in the course of time

The types of computer applictions dominant at different times may be classified into three generations according to their influence on the development of computing.

The first generation, characterized by numerical computing, led to the development of many new algorithms. It transformed numerical analysis from a craft to be practiced by every applied mathematician into a field for specialists. It soon became obvious that writing good (efficient, robust) numerical software requires so much knowledge and effort that this task cannot be left to the applications programmer. The development of large portable numerical libraries became one of the major tasks for professional numerical analysts.

The second generation, hatched by the needs of commercial data processing, led to the development of many new data structures. It focused attention on the problem of efficient management of large, dynamic data collections, mostly under batch processing conditions. Searching and sorting were recognized as basic operations whose time requirements turned out to be the bottleneck for many applications.

We are now on the threshold of a third generation of applications, dominated by computing with pictorial and geometric objects. This change of emphasis is triggered by today's ubiquitous interative use of personal computers, and their increasing graphics capabilities. It is a simple fact that people absorb information fastest when it is presented in pictorial form, hence computer graphics and the underlying processing of geometric objects will play a role in the majority of computer applications. The field of computational geometry has emerged as a scientific discipline during this past decade in response to the growing importance of processing pictorial and geometric objects. It has already created novel and interesting algorithms and data structures. Geometric computation may well replace the more traditional fields of numeric computation and of data management as the major research area in algorithm analysis.

Computational geometry - theory and practice

During the seventies goemetric problems caught the attention of researchers in concrete complexity theory. They brought to bear the finely honed tools of algorithm analysis and achieved rapid progress. Elementary problems (e.g. determining intersections of simple objects such as line segments, aligned rectangles, polygons) yielded elegantly to general algorithmic principles such as divide-and conquer or plane sweep. But in many instances a surprisingly large increase of difficulty showed up in going from two to three dimensions: for example, intersection of polyhedra is still an active research topic where major efficiency gains are to be expected. The theory of computational geometry, although well underway, has as yet explored only a fraction of its potential territory.

The practice of computational geometry is even less well understood. Many important geometric problems in computer-aided design, in geographical data processing, in graphics do not lend themselves to being studied and evaluated by the asymptotic performance formulas that the algorithm analyst cherishes. For example, asymptotics does not help in answering the question whether we can access an object in one disk access or two, thus being able to display it "instantaneously" on the designer's screen - realistic assumptions about the size of today's central memories are needed. Nor will asymptotics settle the argument raging in the CAD community between proponents of boundary representations and adherents of constructive solid geometry - taste, experience, and type of application are the relevant parameters. And below the highly visible issues of object representation, data structures and algorithms, hide the tantalizing

(Continued on p. 18)

20. A.K. Lenstra, *Polynomial factorization by root approximation,* report IW 242/83, Mathematisch Centrum, Amsterdam.

21. H.W. Lenstra, Jr., *Integer programming with a fixed number of variables,* Math. Oper. Res. **8** (1983), 538-548.

22. M.O. Rabin, *Probabilistic algorithms in finite fields,* SIAM J. Comput. **9** (1980), 273-280.

23. Palfy, *A polynomial bound for the orders of primitive solvable groups.* J. of Algebra, July 1982, 127-137.

24. A. Schönhage, *Factorization of univariate integer polynomials by diophantine approximation and by an improved basis reduction algorithm,* to appear in proceedings 11th international colloquium on automata, languages and programming, Antwerp 1984.

25. A. Shamir, *A polynomial time algorithm for breaking the Merkle-Hellman cryptosytem,* Proceedings 23th IEEE symposium on foundations of computer science (1982), 145-152.

26. H. Te Riele, *Mertens' conjecture disproved,* CWI Newsletter **1** (1983), 23-24 (Centrum voor Wiskunde en Informatica, Amsterdam).

27. P. Van Emde Boas, *Another NP-complete partition problem and the complexity of computing short vectors in a lattice,* Rep. Dep. Math. 81-04, University of Amsterdam, April 1981.

28. J. Von zur Gathen, *Hensel and Newton methods in valuation rings,* Math. Comp., to appear.

29. J. Von zur Gathen, *Factoring sparse multivariate polynomials,* Proceedings 24th IEEE symposium on foundations of computer science (1983), 172-179.

(Continued from p. 19)

details the numerics of computational geometry - such as the problems caused by braiding straight lines.

The state of the art: what is it, what should it be?

Today's commercially available software in computer graphics and CAD has not yet taken into account the results of computational geometry. Straightforward algorithms are mostly used whose theoretical efficiency is poor as compared to known results. Perhaps the straightforward algorithms are better in practice than theoretically optimal ones, but such difficult questions have hardly been investigated, as CAD systems development today is so labor intensive that all resources are absorbed by just getting the system to work, and algorithm analysis has so far largely restricted itself to theoretically measurable performance.

We know by analogy with numerical analysis what the next step should be in the maturing process of computational geometry: The development of efficient, portable, robust program libraries for the most basic, frequent geometric subroutine library of CAD, thus exposing theoretical results to a severe practical test. The interaction between computational geometry and computer-aided design promises to be mutually beneficial.