Theoretical Computer Science 408 (2008) 99–105

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

The unpredictable deviousness of models[☆]

Franco P. Preparata

Department of Computer Science, Brown University, Providence RI, USA

A R T I C L E   I N F O

Keywords:
Algorithmic research
Computation models
Computational geometry
Parallel computing
Sequencing by hybridization

A B S T R A C T

Computation models are central to algorithmic research. A model, designed to capture the essential features of a technology dispensing with irrelevant and burdensome details, is a judicious compromise between simplicity and fidelity (or reflectivity). This approach has unleashed an enormous amount of valuable algorithmic research over the years. However, the pursuit of simplicity may filter out details, once deemed irrelevant, which may later reassert their significance under either technological pressure or more careful scrutiny, in which case the inadequacy of the model cripples the validity of the derived results. Examples of this situation, drawn from computational geometry, numerical parallel computation, VLSI theory, and computational biology, will be reviewed and examined in detail.

1. Introduction

Over the past four decades the design and analysis of algorithms has been a vibrant area of computer science research. While sustained technological innovation improved the performance of digital computers, and the introduction of high-level programming languages eased the task of programming and enabled program portability, it was widely perceived in the early days of the computer era that the development of efficient algorithms for the solution of given problems had to be a main research focus, since the adoption of a superior algorithm could achieve accelerations unattainable by conceivable technological improvements (and would consistently outperform its competitors on future platforms).

Such widely held perception, however, needed to be placed on a solid quantitative basis. It was obvious that better algorithmic performance had to translate into a more economical use of resources, thereby revealing a natural engineering component in the emerging discipline of computer science. The few available computers had different organizations, instruction repertories, and memory structures, all differences which complicated a meaningful quantitative evaluation of performance. However, the specific differences appeared to be of secondary importance with respect to the large body of the shared features. Indeed, the extant machines all shared the following features: a store for instructions and data, an accumulator register, and an arithmetic-logic unit; the flow of information between these components was governed by a control unit, which executed instructions fetched from the store. The identification of these common characteristics is tantamount to the conception of an abstract machine of which the existing machines are instantiations. The abstract machine – basically, the "stored-program computer" conceived in the mid forties by Von Neumann, Eckert, and Mauchley[1] – became a "model" of a vast class of computing platforms.

Why is a model useful? Abstractly, one may say that any description we offer of an extant "system" is inherently a model of that system, (where a "system" here is supposed to evolve in time). But this use of the word "model" conceals the idea of "simplification", and a model is more accurate (or "faithful") to the extent in which more details of the system are represented in it. Obviously, the more details that are reflected in the model, the more onerous it becomes to predict its

[☆] Material leading to this paper was presented in keynote lectures in Summer 2006.
E-mail address: franco@cs.brown.edu.

[1] Regrettably, the names of Eckert and Mauchley are rarely mentioned.

0304-3975/$ – see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.tcs.2008.08.019

time-evolution. The "art" of modeling may therefore be characterized as the dialectics between *simplicity* and *reflectivity*. Simplicity is essential to guarantee formal tractability of the model, i.e., the ability to make predictions about its evolution; reflectivity strives to ensure that essential features of the system are retained in the model and only secondary items are filtered out. Ultimately, the validity of a model is measured by how faithfully the formally derived predictions are applicable to real-world concrete systems. The elegance and effectiveness of a model is therefore expressed by its simplicity and reflectivity.

In connection with algorithm design and analysis, this philosophy took shape in the early days of the computer eras and was eloquently expressed in the books by Knuth [1] and by Aho, Hopcroft and Ullman [2]. The latter, in particular, spelled out the outlook which informed algorithmic research thereafter: A model of computation is a trade-off between simplicity and reflectivity, and provides the cost structure which forms the basis of performance evaluation. Indeed,

A model of computation specifies the primitive operations that may be executed and their respective costs.

Within this general philosophy, the model of the RAM (Random Access Machine) was formalized, and was inspired by the prevalent features of most real computers. Such machines were synchronous, had similar instruction sets and comparable word sizes, and each address of a sufficiently large store (the "main memory") was accessible in constant time. Therefore, the RAM model has a random-access memory of arbitrary size, a single accumulator, and word-size adequate to hold the data of interest.

One central feature of the computation model viewpoint is that the use of a resource (such as time or space) is not to be estimated in absolute terms. An absolute estimate would be unnecessarily clumsy and complicated when comparing the performance of competing algorithms (it is intuitive that a "better" algorithm will remain better as technology evolves). Rather, with reference to time, for example, one would identify a key operation repeatedly executed by an algorithm (such as comparisons in sorting), and since the actual running time is closely "proportional" to the number of such executed operations, one would determine how this number grows with the size of the problem. The rate-of-growth approach implies that each performance measure is asserted modulo a multiplicative constant and expressed as a function of parameters specifying the problem size.

The viewpoint of the RAM model, eminently acceptable due to its apparent reflectivity of reality, by its inherent simplicity unleashed vigorous algorithmic research that, over the decades, has produced an impressive body of knowledge. However, any simplification implies the selection of the features not to be represented in the model; but details that in good faith may appear secondary or irrelevant when the model is first formulated may be likely to reassert their significance when, under the presence of technological innovations, the model reaches beyond its realistic confines.

In fact, not surprisingly, once it is formalized and accepted by a research community, a model takes a life of its own. It becomes itself the only reality and defines the "rules of the game".

The most significant illustration of this potential danger is the occasional misuse of the "asymptotic-analysis viewpoint" (which – it must be stressed – is a unique, powerful feature of algorithmic analysis, adopted today in all areas of computer science). As noted above, the necessity to abstract from specific machines leads one to neglect multiplicative constants and to focus exclusively on the rate of growth of functions expressing the use of resources. Neglect of constants was wisely intended as a liberation from burdensome inessential details, not as a license to neglect constants no matter what their size. Indeed, the definition of the asymptotic notation, when referring to problem size n, explicitly contains a clause such as: "there exists an integer n_0 such that for all $n \geq n_0$", with the intent that any appraisal of performance must take into consideration how the critical parameter n_0 compares with values of n of practical significance. Unfortunately, explicit reference to this clause is normally avoided, leading sometimes to dubious, or outlandish, statements of declaring superior, or even "optimal", algorithms that under no circumstances will ever be implemented, contrary to the conventional interpretation of the word "optimal". An example of this excess was the declared "optimality" of the Ajtai–Komlos–Szemeredi sorting network [3], which – while representing a true masterpiece of combinatorial acumen – in no conceivable practical instance will ever outperform the suboptimal, but eminently elegant, sorting networks of Kenneth Batcher [4].

Not all unwarranted claims of this type are so obviously egregious; others are much more subtle, so that a critical avoidance of this type of abuse should become an essential part of algorithm design.

While the abuse of the asymptotic notation is the most obvious uncritical use of a computation model, it is certainly not alone, as has been noted by many researchers. Although, in general, model development in algorithmics has been very successful and productive, one should not forget that a model is inherently an approximation, albeit extremely productive, of reality. Being remiss in critically scrutinizing the applicability of the model to specific situations may be the source of very serious disappointments when trying to translate algorithms into executable programs.

In the next sections I shall review some cases, part of my personal experience, which substantiate the shortcomings of a blind acceptance of computation models. Parts of the material are succinct paraphrases of results presented in papers I have authored or co-authored. I also honestly recognize that I am myself responsible for some of the exuberance which I expose in this paper; as they say, the passage of time normally makes people wiser.

2. Computational geometry: Intersections of line segments

The analysis and designs of algorithms for geometric applications became an active area of algorithmic research in the mid-seventies. The issue of the appropriate computation model was readily disposed of, since the RAM provided the natural functional background, with the proviso that it be augmented with the capability to operate on *real numbers*. Thus,

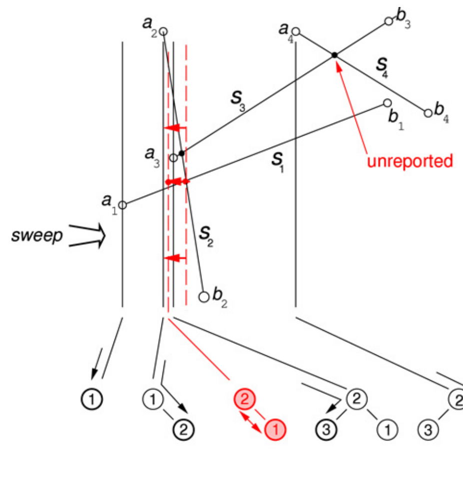

Fig. 1. Point a_3 is incorrectly placed to the right of point $s_1 \cap s_2$, which causes the erroneous inversion of these two segments in the data structure.

Computational Geometry has traditionally adopted the arithmetic model of exact computation over the real numbers. Perhaps superficially, the potential inadequacy of standard floating-point technology was overlooked. On the positive side, the proposed model of the real RAM has been extremely productive in terms of algorithmic research, since it has permitted a vast community to focus on the elucidation of the combinatorial (topological) properties of geometric problems, thereby leading to sophisticated and efficient algorithms for their solution. However, real-number arithmetic in actual computers is inherently approximate. Whereas approximations may be acceptable in geometric constructions (such as computing the coordinates of the intersection of two segments), they become very critical in the evaluation of conditions that control the branchings of a program. Thus, approximate arithmetic affects not only the quality of the results (a tolerable shortcoming) but in some cases even the validity of specific algorithms (an unacceptable outcome).

Against this background, we shall revisit the segment-intersection problem, which is stated as follows:

Given a set S of n straight-line segments in the plane, report all their intersections.

For a succinct review, the standard algorithm (by Bentley and Ottmann [5]) is a plane sweep realized by a translation of a straight line. The procedure first sorts the $2n$ segment endpoints by increasing x-coordinates and stores the sorted points in a priority queue X (referred to as the "event-schedule"). Next, the algorithm begins sweeping the plane from left to right with a vertical line L, and maintains a data structure Y that represents the segments of S currently intersected by the sweep-line L, ordered according to the ordinates of their intersections with L. Intersections only occur between segments that become adjacent in Y, so that any time a new adjacency is created it is tested for a possible intersection. Adjacencies are created in Y either by insertion/deletion of segment endpoints as they are encountered in the plane sweep, or by order exchanges caused by intersections. An intersection, upon detection, is inserted into X according to its abscissa.[2] The Bentley–Ottmann algorithm is easily proved correct, since it maintains the correct order of the sweep-line intersections with the segments and adjacency in this order is a necessary condition for an intersection to occur.

As alluded to above, the approximate real-number arithmetic is not problematic in the computation of the coordinates of an intersection, as an output result; problematic, instead, is the insertion of this intersection into the X data structure, since this operation involves comparing the intersection's abscissa with other stored abscissae.

Consider, for example, the simple situation illustrated in Fig. 1. We denote $x(p)$ the abscissa of point p, and $(a_ib_i) \cap (a_jb_j)$ the intersection of two segments; the Y-order is stored in a binary search tree, whose time-evolution is also illustrated at the bottom of Fig. 1. First, segments s_1 and s_2 are correctly inserted; next, suppose that the approximate arithmetic yields the erroneous result

$$x((a_1b_1) \cap (a_2b_2)) < x(a_3),$$

causing the erroneous inversion of s_1 and s_2. Therefore, when segment s_3 is inserted the Y-order is incorrectly placed in the data structure, and it can be verified that $(a_3b_3) \cap (a_4b_4)$ may not be reported because s_3 and s_4 do not appear adjacent in Y.

The described incorrect behavior is due to the fact that the correct insertion of an intersection into X is based on the evaluation of a predicate, expressed by the sign (or the zero) of a polynomial in the coordinates. Specifically, given abscissa x_0 and segments $((x_{11}, y_{11}), (x_{12}, y_{12}))$, $((x_{21}, y_{21}), (x_{22}, y_{22}))$, the predicate in question is expressed by the polynomial

$$x_0 \begin{vmatrix} x_{12} - x_{11} & y_{12} - y_{11} \\ x_{22} - x_{21} & y_{22} - y_{21} \end{vmatrix} - \begin{vmatrix} x_{12} - x_{11} & x_{11}y_{12} - x_{12}y_{11} \\ x_{22} - x_{21} & x_{21}y_{22} - x_{22}y_{21} \end{vmatrix}$$

which has degree 3.

[2] Of course, a given intersection may be detected several times; the standard algorithm typically resolves multiple detections by performing a preliminary membership test for an intersection in X and omitting insertion if the intersection has been previously recorded.

We conclude that the implementation would certainly be correct if we specify the arithmetic model so that polynomials of degree 3 are correctly evaluated over the integers. The original specification of the computation model (the real RAM) clearly did not foresee the relevance of arithmetics, and removed this important detail from the model. As a way to obviate the shortcoming, we proposed [6] that the model be augmented with the specification that, if the application involves the evaluation of predicates of *degree d*, the corresponding polynomials must be correctly evaluated over the integers, possibly with *d*-fold multiple precision. In other words, we should replace the real RAM with the *degree-based RAM* [6]. A substantial modification of Bentley–Ottmann's algorithm with predicates of degree 2 was presented in [7].

3. Parallel numerical computation: Matrix inversion

A variety of models have been suggested for parallel computation. The differences mainly concern how the various processing units are physically (or logically) interconnected and how they exchange information. However, in each of these different models the processing units conform generally to whatever model is adopted in the corresponding sequential setting (basically, the RAM model). As we noted earlier, the RAM model, when applied to numerical computations, contemplates unit-time arithmetic operations.

Within this model, vigorous research has focussed on the design of fast algorithms for numerical problems. Specifically, as is typical of parallel algorithmics, the main objective can be summarized as the design of algorithms that accelerate the computation while possibly preserving the overall number of operations. As is well known, the gold-standard of acceleration is a so-called NC-algorithm, i.e., an algorithm running in time polylogarithmic in the problem size.

Linear algebra has always been a main focus of research. Whereas matrix multiplication is trivially parallelizable, matrix inversion, stated as

Given a nonsingular matrix A, compute its inverse A^{-1}

appeared immediately as a substantially more difficult problem. Therefore the elegant NC-algorithm of Csanky [8] was an exhilarating surprise for the research community.

With regard to matrix inversion, the quality of the results can be analyzed from two alternative viewpoints, starting from an exactly representable matrix (i.e., a matrix whose entries are integers of maximum modulus M):

(1) Evaluate the number of digits required to compute its rational inverse (using exact integer arithmetic).
(2) Resort to an arithmetic based on fixed-length representations and estimate upper-bounds to the resulting round-off error (using floating-point reals).

The now classical Csanky algorithm is summarized as follows. Given an $n \times n$ matrix A, the Cayley–Hamilton Theorem states that A satisfies its characteristic polynomial $\phi(\lambda) = c_0 + c_1\lambda + \cdots + \lambda^n$, i.e., $\phi(A) = 0$. As a consequence, if A is nonsingular,

$$A^{-1} = -\frac{1}{c_0}(c_1 + c_2A + \cdots + A^{n-1})$$

Assuming that the powers $A^2, A^3, \ldots, A^{n-1}$ are easily computed in the given model, the problem is reduced to the computation of $\phi(\lambda)$. The latter problem is solved on the basis of Leverrier's Theorem, which relates the coefficients of $\phi(\lambda)$ to the traces of the powers of A (using Newton's identities), through the following equation[3]:

$$\begin{vmatrix} 1 & 0 & \ldots & \ldots & 0 \\ s_1 & 2 & \ldots & \ldots & 0 \\ s_2 & s_1 & \ldots & \ldots & 0 \\ \\ s_{n-1} & s_{n-2} & \ldots & s_1 & n \end{vmatrix} \begin{vmatrix} c_{n-1} \\ c_{n-2} \\ c_{n-3} \\ \\ c_0 \end{vmatrix} = - \begin{vmatrix} s_1 \\ s_2 \\ s_3 \\ \\ s_n \end{vmatrix}.$$

We recall that $A^{-1} = \mathrm{Adj}(A)/\det(A)$. Considering first the exact-arithmetic approach, we note that there are matrices A for which entries of $\mathrm{Adj}(A)$ are relatively prime to $\det(A)$; one example is given by the trivial tridiagonal matrix

$$\begin{vmatrix} -M & -1 & & & \\ -1 & -M & -1 & & \\ & & \ldots & & \\ & & -1 & -M & -1 \\ & & & -1 & -M \end{vmatrix}.$$

This implies that $\det(A)$ must be exactly represented; therefore, by resorting to modular arithmetic – using Hadamard's bound and letting M denote a bound on the size of the entries of A – we conclude that a word-size of

$$\log \det(A) \le \log(M\sqrt{n})^n = \frac{n}{2}\log n + n \log M$$

bits is certainly adequate for correctness. Unfortunately, such word-size is also necessary, because, for each n, there exist matrices whose determinant closely approximate Hadamard's bound [9]. The conclusion is that, if exact rational inverses

3 s_j denotes the trace of A^j.

are sought, the time performance of Csanky's algorithm is not as stated, since the instruction time is dramatically different from a constant (as is generally assumed in the model).

Alternatively, one may ask whether Csanky's algorithm can be rescued if we relax the requirement of exact rationals and resort to standard fixed-length approximations, such as floating-point computation for some fixed mantissa length b. To my knowledge, no detailed analysis of this situation has been carried out; however, there is documented experimental evidence that floating-point implementations of Csanky's algorithm expose numerical instability, even when the matrix A has desirable numerical properties (low condition number) [10]. To further substantiate this point, let us consider the case of a trivially invertible matrix, on which the blind application of Leverrier's method results in gross failure. One such trivial $n \times n$ matrix is $A = kI$, for some small integer k. The trace s_j of A^j is $s_j = nk^j$, and, according to the standard model of floating-point computations with b-bit mantissa, the corresponding relative error is approximately $\lfloor nj/b \rfloor 2^{-b}$. Let $L(A)$ denote the (lower-triangular) Leverrier matrix of A. It is also interesting to observe that even in the trivial case $A = kI$, for some small k (a trivially invertible matrix), the floating-point representations of the trace parameters s_j, for sufficient large j ($k^j > 2^b$), are affected by substantial errors. In the inversion of the resulting triangular Leverrier matrix, these errors translate into errors affecting the coefficients of $\phi(\lambda)$. If the latter errors are to be bounded by 2^{-b}, then word-size $\Theta(n)$ appears necessary [9].

The conclusion of this discussion is that the adopted parallel computation model is not adequate. Upon closer analysis, the parameter "numerical accuracy", which was hastily glossed over in the formulation of the model, turns out to be critical in a realistic analysis of proposed algorithms.

4. Interconnections for parallel computation: The non-scalable hypercubes

Prompted by significant technological innovations (Very Large Scale Integration, VLSI) which moved parallel computation from the realm of intellectual speculation to that of engineering realizations, the design of interconnection networks for parallel computation ("parallel architectures") became a major research focus in the eighties. Whereas programming models, such as the P-RAM, made reference to a single address space accessible in parallel at unit-cost by the various processing units, a complementary effort was aimed at the conception of actual networks of processors, so that exchange of information occurs between directly connected processors. In the network model, closer to the engineering of parallel computation, one could actually trace the computation as it physically evolved. An important objective of this research was therefore the elucidation of the communication structures of different algorithms, thereby forming classes of problems on the basis of their communication structures, identified as "algorithmic paradigms", to be matched to their best suited interconnection. Most of the results in this area were organically and conclusively presented in Tom Leighton's compendium, "Introduction to parallel algorithms and architectures" [12].

Naturally, the research goal was the reduction of computing time while trying not to increase the overall computational work of the most efficient sequential counterparts. Assuming, as usual, that each processor (node) conformed to the standard RAM model (i.e., unit time for arithmetic operations), computation time hinged on the modeling of communication time. Here the story of parallel computation is therefore strongly intertwined with that of its enabling technology VLSI. The technology of digital circuits fabricated by integrating within the same material both devices and interconnections, in realizations that are simultaneously functionally correct, energetically efficient, and parsimonious (in terms of used material), called for compact layouts. Indeed, the following general model of synchronous VLSI computation, which dispensed with the specific details of the various manufacturers, was formulated by Clark Thompson [11] and well received by the research community:

(1) Networks (devices and wires) are mapped to a layout grid, two-dimensional but possibly three-dimensional. The characterizing layout parameter is the *feature size* λ, expressing the width of real-estate domain assignable to a wire (so that manufacturing guarantees continuity and insulation, as appropriate). [It was correctly expected that λ would decrease as manufacturing evolved.]
(2) The transmission time over each internode connection is constant.

The second axiom of the model, obviously a significant simplifying factor, was suggested by the extant technology and implementation practice. Such an assumption is certainly legitimate when the actual transmission time is negligible with respect to the communication set-up time.

The Thompson model was central to the development of the so-called *area–time theory* of VLSI, whose objective was the intent to more closely relate parallel computation to its technology. It is based on the notion of *information exchange*, i.e., the amount I of information (number of bits) that must be exchanged across a layout section, that leaves about one half of the input data on either side. Since the section length is easily related to the layout area, we obtain an elegant relationship (trade-off) between area and time. The "area–time trade-off" confirms the intuition that, for a given problem, faster algorithms can be implemented in larger layouts. Indeed, for a number of problems (merging, FFT, integer multiplication, etc.) it is possible to exhibit a whole spectrum of network implementations, all meeting the area–time lower bound (and therefore viewed as optimal) and spanning a wide range of computation times. The lower end of this range is typically $O(\log n)$, if $I = \Omega(\text{poly}(n))$. Such a finding gave special status to networks of diameter logarithmic in the problem size – such as trees and hypercubes – for the attainment of maximal speed-ups. Indeed, it is not surprising that an early, very celebrated, commercial parallel computer, the CM-2 of Thinking Machines Inc., had a hypercubic interconnection (in fact, an interconnection very close to the cube-connected-cycles [13], an efficient hypercube emulator).

However, the area–time model, and all derived computational consequences, rests on the two axioms given above. There is no doubt on the validity of the space-axiom, parameterized by λ. Much more problematic is the time-axiom, which was originally scrutinized in a rather superficial way. As technology evolves, the interconnection length becomes significant and the original axiom can be maintained only as long as all transmission edges have identical, or near-identical, length (as happens in mesh-connected networks, for example). Indeed, it is legitimate to envision a "limiting" technology [14] where transmission time grows at least linearly with wire length, thereby invalidating Thompson's model.

One wishes to ensure the scalability of asserted properties, i.e., their independence of the problem size within a given technology. For example, a number of problems, referred to as "transitive functions", such as FFT on n points, satisfy an area–time bound of the form

$$AT^2 = \Omega(n^2).$$

A hypercubic network achieves the bound for time $T = O(\log n)$ and area $A = O((n/\log n)^2)$; however, as we scale up the problem size, since the layout must contain wire of length $\Omega(n)$, we must also have $T = \Omega(n)$.

This shows that a detail, reasonably overlooked because irrelevant in the extant technology, turns out at a later stage to invalidate a computation model. Indeed, nonscalable hypercubic architectures are not likely to lead to massive parallel systems in the future.

5. A lesson from computational biology: Sequencing by hybridization

Computational biology is a newcomer to the general area of algorithmic research. The enormous amount of accumulated data and the "digital nature" of molecular biology naturally call for the application of algorithmic techniques to the solution of a host of biological problems.

A central application in molecular biology is the *sequencing of nucleic acids*, i.e., the determination of the sequence of nucleotides of a chosen fragment of a DNA/RNA molecule. In recent years, a radically new technique has been proposed [15] as a potential alternative to the traditional sequencing by gel electrophoresis. This technique, called *Sequencing by Hybridization* (SBH), in its original formulation obtains all fixed-length substrings (probes) of a target sequence and algorithmically attempts the reconstruction of the sequence using the collection of such probes.

SBH is based on the use of a chip fabricated on a glass substrate with photolithographic techniques. The active area of the chip is structured as a matrix, each region of which (called a *feature*) contains a very large number of copies of a specific oligonucleotide of length w, biochemically attached to the chip surface. The chip is immersed under controlled conditions within a solution of suitably labeled target DNA, a copy of which will bind to an oligonucleotide if the oligonucleotide is complementary, in the Watson–Crick sense, to one of its substrings. The labeling of the target allows visualization of the chip features containing binding oligonucleotides. The described experiment provides all substrings of length w of the target: the next task is the algorithmic reconstruction of the target from the set of its substrings (called the *spectrum*).

The adopted model is extremely precise and was perhaps motivated by the high accuracy of DNA replication and RNA transcription. The hybridization experiment is assumed to be noiseless, i.e., the spectrum is *exactly* the set of all substrings of the target (with no multiplicity detail). This model, for its combinatorial neatness, was naturally very appealing to computer scientists and became the unchallenged paradigm. SBH research proceeded in subsequent years mostly as an algorithmic exercise, with the expectation that the corresponding laboratory technology was being fine-tuned.

SBH research, however, was soon confronted with a discouraging theoretical difficulty: Sequence reconstruction based on the use of substrings as probes was apparently inadequate to compete against electrophoresis. In fact, analytical considerations revealed that with probes of length k, only target sequences of length $c2^k$ (with $c < 1$) could be reliably reconstructed. For technologically feasible values of k, SBH appeared outclassed by electrophoresis.

The combinatorial shortcoming was eventually remedied by the introduction of non-contiguous (gapped or spaced) probes, conforming to a fixed pattern of probing and nonprobing nucleotide positions [16]. The idea was to realize nonprobing positions by means of "universal bases", i.e., natural or artificial bases exhibiting hybridization non-specificity. With this combinatorial fix-up (where the notion of *subsequence* replaced the original notion of *substring*), it could be analytically shown that gapped probes of adequate length with k specified nucleotides could reliably achieve $O(4^k)$ sequencing lengths, thus injecting new interest into SBH.

As a closer connection with the laboratory was being made, it was soon realized that biochemical difficulties far outweighed combinatorial hurdles. The originally adopted model of noiseless hybridization was an oversimplification, and a coarse one at that, of the physical reality: differences in "melting temperatures" for different oligonucleotides, formation of secondary structures preventing hybridization, and inadequate nonspecificity of proposed universal bases, caused a rejection of the original combinatorial model [17]. The accumulated knowledge perhaps awaits a transplant in related applications.

6. Closing remark

The lesson to be drawn from the preceding case analysis is two-fold: effective computation models are vital to the vigorous development of algorithmic research. However, the savvy computer scientists should be aware of their shortcomings, and should constantly verify that a model is being applied within the confines of its original formulation.

References

[1] D.E. Knuth, The Art of Computer Programming, Addison-Wesley, 1968.
[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Algorithms, Addison-Wesley, 1974.
[3] M. Ajtai, J. Komlos, E. Szemeredi, An $O(n \log n)$ sorting network, in: Proc. 15th ACM Symp. on Theory of Computing, 1983, pp. 1–9.
[4] K. Batcher, Sorting networks and their applications, in: Proc. of the AFIPS Spring Joint Comput. Conf., 1968, pp. 307–314.
[5] J.L. Bentley, T.A. Ottmann, Algorithms for reporting and counting geometric intersections, IEEE Transactions on Computers 28 (9) (1979) 643–647.
[6] G. Liotta, F.P. Preparata, R. Tamassia, Robust proximity queries: An illustration of degree-driven algorithm design, SIAM Journal on Computing 28 (3) (1998) 864–889.
[7] J.-D. Boissonnat, F.P. Preparata, Robust plane sweeps for intersecting segments, SIAM Journal on Computing 23 (5) (2000) 1401–1421.
[8] L. Csanky, Fast parallel matrix inversion algorithms, SIAM Journal on Computing 5 (4) (1976) 618–623.
[9] B. Codenotti, M. Leoncini, F.P. Preparata, The role of arithmetic in fast parallel matrix inversion, Algorithmica 30 (4) (2001) 685–707.
[10] J.W. Demmel, Trading off parallelism and numerical accuracy, CS-92-179 University of Tennessee, June 1992 (Lapack Working Note 52).
[11] C.D. Thompson, Area–time complexity for VLSI, in: Proc. 11th ACM Symp. on Theory of Computing, 1979, pp. 81–88.
[12] F.T. Leighton, Introduction to Parallel Algorithms and Architectures, Morgan Kaufmann, 1992.
[13] F.P. Preparata, J. Vuillemin, The cube-connected-cycles: A versatile network for parallel computation, Communications of the ACM 24 (5) (1981) 300–309.
[14] G. Bilardi, F.P. Preparata, Horizons of parallel computation, Journal of Parallel and Distributed Computing 27 (2) (1995) 172–182.
[15] W. Bains, G.C. Smith, A novel method for DNA sequence determination, Journal of Theoretical Biology 135 (3) (1988) 303–307.
[16] F.P. Preparata, E. Upfal, Sequencing-by-Hybridization at the information-theory bound: An optimal algorithm, Journal of Computational Biology 7 (3–4) (2000) 621–630.
[17] F.P. Preparata, Sequencing-by-Hybridization revisited: The analog spectrum proposal, IEEE/ACM Transactions on Computational Biology and Bioinformatics 1 (1) (2004) 46–52.