DNA-based Computation Times

Yuliy Baryshnikov[1] Ed Coffman[2] Petar Momčilović[3]

1. Bell Labs, Lucent Technologies, Murray Hill, NJ 07974
2. Department of Electrical Engineering, Columbia University, New York, NY 10027
3. IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

February 21, 2004

Abstract

Speed of computation and power consumption are the two main parameters of conventional computing devices implemented in microelectronic circuits. As performance of such devices approaches physical limits, new computing paradigms are emerging. Two paradigms receiving great attention are quantum and DNA-based molecular computing.

This paper focuses on DNA-based computing. This paradigm can be abstracted to growth models where computational elements called tiles are self-assembled one by one, subject to some simple hierarchical rules, to fill a given template encoding a Boolean formula. While DNA-based computational devices are known to be extremely energy efficient, little is known concerning the fundamental question of computation times. In particular, given a function, we study the time required to determine its value for a given input. In the simplest instance, the analysis has interesting connections with interacting particle systems and variational problems.

1 Introduction

The elementary logic unit of DNA computing is the *tile* modeled as shown in Figure 1 as a marked or labeled square. In the simplest version, the label values are 0 or 1, and they break down into two input labels on one edge and two output labels on the opposite edge. A computational step consists of one tile bonding to others according to given rules that match input labels of one tile to the output labels of one or two adjacent tiles.[1] Figure 1 shows a simple example. Successive bonding of tiles performs a computation, e.g., the evaluation of a Boolean formula on given inputs, in a *self-assembly* process guided by a *template.* The tiles are DNA-based molecular structures moving randomly, in solution, and capable of functioning independently and in parallel in the self-assembly process; the template is needed to properly structure the self-assembly, in particular to impose the sequential constraints on self-assembly needed to produce the desired computation. In contrast to classical computing paradigms, the random phenomena of self-assembly create a randomness in the time required to perform a given computation. The research reported

[1]In the DNA lexicon, the notion of bonding is also commonly referred to as gluing, sticking, or attaching.

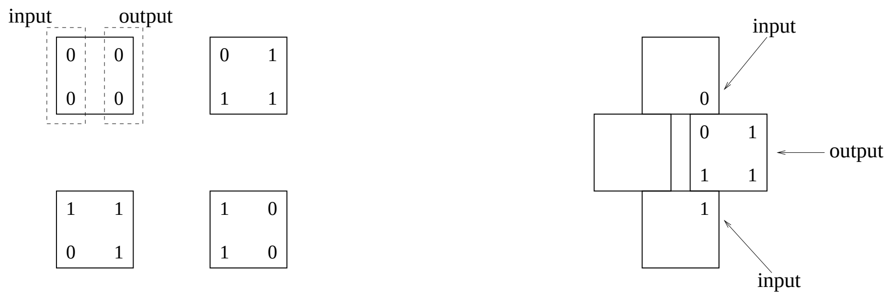

Figure 1: The set of four tiles on the left implements the operation $\oplus$. A tile glues to other tiles only if their corner labels match. On the right operation $0 \oplus 1 = 1$ is performed. The input tiles are preassembled and the correct output tile simply attaches itself to the pattern, effectively obtaining the value of the output bit.

here characterizes stochastic computing times within computational paradigms based on self assembly.

The notion of computing with tiling systems originated with Wang [17] over 40 years ago, but the modern theory emerged within the last several years in the work of Winfree [19], and Rothemund and Winfree [15]. The early theoretical work on general computation focused chiefly on various measures of complexity, in particular, program-size and time complexity [1, 15, 18], but Adleman et al [1, 2] investigated interesting combinatorial questions such as the minimum number of tile types needed for universality, and stochastic optimization questions such as the choice of concentrations that leads to minimum expected assembly times.

The analysis of random phenomena of self assembly has produced few results to date. Adleman et al [3] inaugurated this line of research with an analysis of a baseline linear model of a random self-assembly process. (See [6] for a variant of the linear model.) Baryshnikov et al [5] extended the results to incremental self-assembly systems in which structures grow only one tile at a time, and they introduced a novel control mechanism for maximizing yield in reversible processes. The scale of tile systems in the discrete setting is so large that, to date, the techniques of hydrodynamic limits have shown the most promise. This is further illustrated in the studies of this paper.

Our brief review has focused on mathematical modeling and analysis of general DNA computation, as opposed to constructs solving specific combinatorial problems. Nor have we referred to the extensive literature on the intriguing experimental side of DNA computation. We refer the reader to [2] for a useful summary of these matters and an extensive bibliography.

To fix ideas, we describe the recently proposed techniques of Carbone and Seeman [7] in Section 2. The next section will then fit this scheme into the abstraction of growth models. Section 3.1 studies computation times as the times to grow rectangular constructs, then Section 3.2 generalizes our timing analysis to the growth of arbitrary structures in two dimensions.

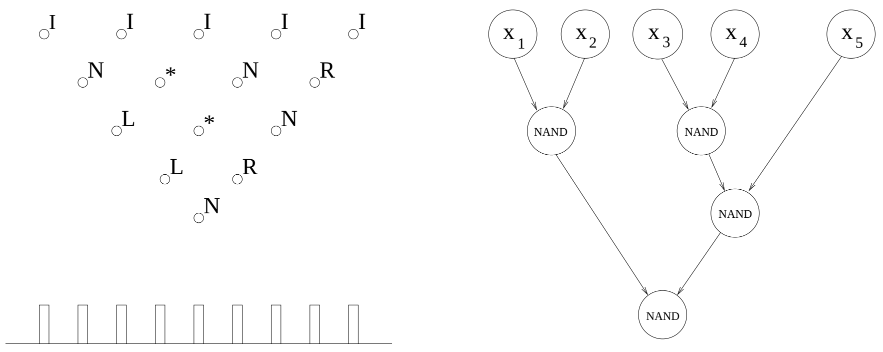

Figure 2: An example of a template and the Boolean formula it encodes [7]. Under the template of labeled pawns one can see its vertical section. In this example four **N**-tiles are used for computation while ten tiles are needed in total (besides the input).

2 DNA computation of Boolean formulas

The template governing computations is a triangular array of nodes with i nodes at level i, $i \geq 1$, as illustrated in Figure 2 [7]. Physically, it may be viewed as an array of posts or nodes, called *pawns*, glued to a metallic substrate. Boolean formulas in the form of trees are mapped onto the template by suitably typing the nodes according to function. Such mappings are not unique in general, but the choice of mapping is of no concern here. As illustrated in Figure 2, the nodes are of four types:

1. input nodes, denoted **I**, where the tiles representing the input are superimposed on the leaves of the tree. Inputs to these tiles are left unspecified.

2. gate nodes, where tiles perform a gating function. The gates in Figure 3 are all NAND gates denoted by **N**.[2]

3. forwarding nodes, where information at a tile input is passed down to the diagonally opposite output; right to left is denoted by **R**, and left to right is denoted by **L**.

4. null nodes, denoted by *****, where tiles perform no function at all. These nodes are introduced in order to meet the technical requirement that all nodes of the template be involved in the mapping (to every node there will correspond a tile).

Tiles are classified in the same way. Physical identification of tile type is encoded by another label, called the *middle* label, which derives from properties of the molecular structure.

Physically, the computation begins by pre-assembling the input tiles to the leaves of the template. Next, the template is put in solution along with the tiles of the various types. The template acts as a lower layer; in an upper layer, tiles then glue to the template by self-assembly; sequential constraints of the hierarchy of gates are ensured by the fact that a tile can bond only if it bonds to tiles at both of its inputs. Each time a gate tile bonds, a Boolean operation is effectively performed. Attachment of the root tile signals completion of the computation and carries the result in its output labels.

[2]Recall that a NAND gate with Boolean inputs x and y outputs the value 1 if and only if either x or y or both have the value 0; and recall that any Boolean function can be implemented using only NAND gates.

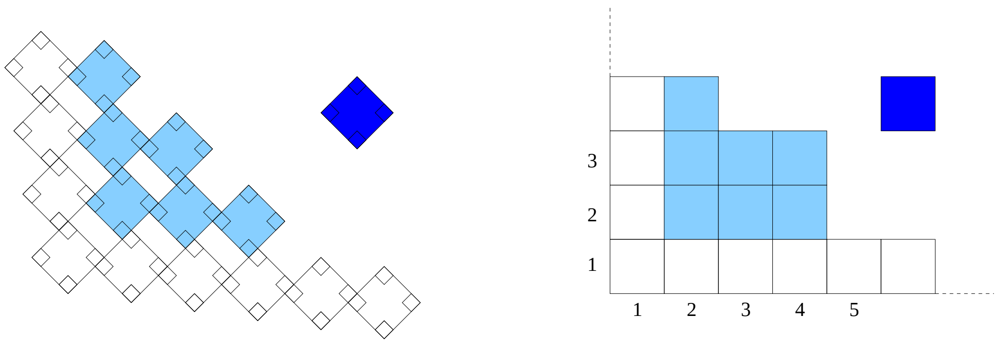

Figure 3: Growth models of DNA-based computation. The actual process of self assembly (left) can be equivalently represented as a growth process (right). There is one-to-one correspondence between tile on the left and squares on the right. White tiles (squares) represent the input.

3 Growth models

In a reference theory for self assembly, it is natural to take the times between successive tile placements as independent, exponential random variables; for convenience, we take the mean as the unit of time. As described, the tiling assemblies of the last section are growth processes, although the computations themselves are processes of coalescence which terminate in a single value at the root of the template. As we shall see, it is more useful to focus on the equivalent growth times. In the abstraction introduced below, we relate the times to grow (self-assemble) for constructs or patterns to classical theories of particle processes; growth is again subject to rules analogous to those governing the self-assembly process of the previous section.

In the next subsection we examine a computational template of a rectangular shape. There we outline the basic ideas that will be used in the subsequent section to argue about the speed of computation on templates having arbitrary shapes.

3.1 Rectangle computation

Figure 3 (left) shows an example in which an initial set of tiles (input) is placed along the two lines (white tiles), and growth proceeds outward in the positive quadrant: the placement of a new tile is allowed only if there are already tiles to the left and below the new tile's position which match labels as before. The left-and-below constraint is equivalent to requiring a newly added tile to bond at both of its input labels. In the example of Figure 3, new tiles can be added only to the three "notches." The eventual output of the computation is represented by the dark tile which can be attached only after all tiles to its left and below are in place. The tiles that perform computation, i.e., non-input and non-output tiles, are lightly colored in Figure 3.

An abstraction of this process operating in the positive lattice is shown on the right in Figure 3. In this model, in which connections to well understood, cognate processes are most easily seen, a new square can be added only in vacant positions having a square both to the immediate left and immediately below the position. Only input squares can lie along the coordinate axes, so under the placement constraint and any finite initial set of squares,

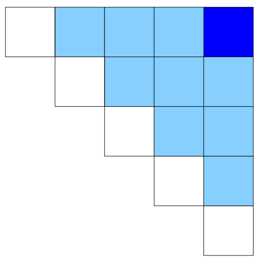

Figure 4: An abstraction of the device device described in [7] (see Figure 2) that computes a single bit upon the completion of the computation (assembly). The input tiles are shown in white while the output tile is dark.

a growth process terminates in finite expected time in a rectangular shape. For example, it is easy to see that, after 8 more tiles (squares) are added to the assembly of Figure 3, a rectangular array results (including the (final) output tile). As in the left figure, the result of the "computation" is depicted by the dark square in the upper-right corner. Clearly, there is a one-to-one mapping between the processes on the left and right in Figure 3. In particular, the template shown in Figure 2 maps to the triangle shown in Figure 4; as before, the dark tile represents the output while the white tiles carry the input information.

The fundamental quantity of interest is the computation time, or equivalently, the time until the final square (represented by the dark square in position (M, N) of Figure 3) is in place. We denote this random completion time by $C_{M,N}$. Let $T_{i,j}$ be the time it takes for a tile to land at position (i, j) once the conditions are favorable; that is, once both positions $(i, j-1)$ and $(i-1, j)$ are tiled. Recall that our basic assumption is that the $T_{i,j}$'s are independent exponential random variables with unit means. Similarly, by $C_{i,j}$ we denote the time until the square (i, j) becomes occupied. Then the completion times can be written in terms of attachment times $T_{i,j}$ by means of the following recursion

$$C_{i,j} = \max(C_{i-1,j}, C_{i,j-1}) + T_{i,j},$$

with initial conditions $C_{i,1} = 0$ and $C_{1,j} = 0$ for $1 \leq i \leq N$ and $1 \leq j \leq M$. The recursion simply reflects the fact that a tile can be attached only if tiles to its left and below are already in place. By unwinding the recursion one can write the completion time as

$$C_{M,N} = \max_{\pi\colon (1,1)\to(M,N)} \sum_{(i,j)\in\pi} T_{i,j},$$

where the maximum is taken over all paths π from $(1,1)$ to (M,N) consisting of segments going north or east only. We remark that the basic recursion formula above can be thought of as an instance of an iterative multiplication of matrices in the so-called *max-plus* algebra, also known as *idempotent algebra* (see [13, 8]). In this formulation, the random values of interest, i.e., the completion times of a computation, are analogous to the matrix elements of the product of a large number of large random matrices.

Alternatively, $C_{M,N}$ is the time when the M-th particle and N-th hole exchange positions in the totally asymmetric simple exclusion process (TASEP) on the integers starting from the megajam configuration: the positions (integers) to the left of the origin are all occupied by particles, and all positions to the right of the origin are empty. At independent, unit-mean exponentially distributed times, particles attempt to move to the right. A move actually occurs only if the adjacent position is a hole, i.e., is unoccupied. We further exploit the correspondence between the TESAP process and the assembly process on the plain in the next subsection.

Remarkably, the exact hydrodynamic-scale behavior of the TASEP process is known. As the rectangular DNA computer becomes large, or equivalently, as N, M grow to infinity such that M/N tends to a positive constant, one has [11, p. 412]

$$\lim_{N\to\infty} \frac{C_{M,N}}{(\sqrt{M}+\sqrt{N})^2} = 1. \tag{1}$$

The preceding formula quantifies the degree of parallelism in the computation. Consider, for simplicity, a square template of size $N \times N$. While the number of tiles is N^2, the computational time is linear in N since $C_{N,N} \approx 4N$ for large N. Expression (1) also allows one to estimate required attachment times given the template size and a constraint on the computation time. For example, let the required computation time be 1 second and let the template consist of $10^3 \times 10^3 = 10^6$ tiles. Then, given that the attachment times are iid. exponential random variables their common mean needs to be $1/(4 \cdot 10^3) = 0.25$ milliseconds.

We conclude this subsection with two observations. First, prior to the computation the set of input tiles needs to be prefabricated. Potentially, the input can be linear in N and, therefore, the prefabrication of the input might be the actual bottleneck of the computation. Second, our analysis assumes that no errors occur in the process of computation, i.e., a tile never bonds to an incorrect place on the template. While the main trade-off in microelectronic circuits is between the speed of computation and power consumption, it appears that in DNA-based computational devices it is between the the speed of computation and the correctness of the output. In particular, higher speed requires smaller attachment times, i.e., higher molecular mobility, which can result in higher error rates.

3.2 Computation of arbitrary shapes

In this subsection we examine a DNA computation of an arbitrary shape, say $\lambda\mathcal{D} \subset \mathbb{R}^2$, such that some segments on the boundary of $\mathcal{D}$ correspond to the data, and some to an output, or read-off region (see Figure 5; the input area is the bold line in the southwest, and the output line can be seen in the northeast). We are interested in the case when λ, the scale parameter, is large. In this regime, the number tiles, and, therefore, elementary logic operations in the DNA computer, is large. As in the previous subsection, the computation is a sequence of additions of tiles (subject to the left-below condition) to the original template. Before addressing the computational time on such a device, we describe an additional property of the TASEP process.

This equivalence of the particle process and our assembly problem on the lattice is well-known, but we briefly remind the reader of the mapping from one sample space to the other. We focus on an unbounded template (lattice), such as the one in Figure 6, to which tiles can be attached. Consider a profile of the self-assembly process as shown in Figure 6. The profile is the staircase created by the sequence of boundary tiles; these are the dark tiles in

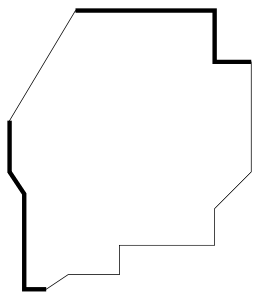

Figure 5: An example of a 2D computational template with a non-rectangular shape on a plane. The two bold lines represent the input (southwest) and output (northeast) regions. The computation (assembly of tiles) begins at the input segment and proceeds towards the output segment.

Figure 6 that touch, either at a vertex or along a side, the untiled region. Scan the profile from upper left to lower right in a sequence of steps, each being a move left to right across the top of a tile at the boundary or down the right side of a tile at the boundary . In this scan produce a sequence of 0's and 1's, a sample of the TASEP, such that, at each step to the right a 0 is added to the sequence so far, and at each step downward a 1 is added.

Suppose that at some time instance the boundary between the tiled and untiled sections is a random set of tiles with parameter $p \in (0,1)$. That is, for every tile on the boundary there exists a boundary tile under it with probability p and that is independent of all other tile positions. In this case the corresponding TASEP process has Bernoulli distribution of particles (ones). Since Bernoulli measure is invariant with respect to TASEP, one concludes that the statistical shape of the boundary remains the same and that the boundary moves to the right at a long-term average speed of q [12, Theorem 4.17] (by symmetry the boundary moves upwards at a long-term average speed of p). Let $v(x,y)$ be the time needed for the boundary to reach point (x,y) (on the hydrodynamic scale). Given that v_x and v_y are partial derivatives of v, the preceding implies that $v_x = q^{-1}$ and $v_y = p^{-1}$ and, thus, since $p+q=1$, one has

$$v_x^{-1} + v_y^{-1} = 1. \tag{2}$$

Next, in the domain $\mathcal{D}$, consider the function $u(x,y)$, the hydrodynamic limit of the propagation time from the input region on $\partial\mathcal{D}$ to a point (x,y). The claim is that at a point where u is smooth (i.e., where u is continuously differentiable), function u satisfies the following PDE (Hamilton-Jacobi equation):

$$H(u_x, u_y) := u_x^{-1} + u_y^{-1} - 1 = 0, \tag{3}$$

where u_x and u_y are partial derivatives of u. To show that u satisfies this PDE, one could approximate locally level lines of u by random lines described in this section and representing the (random) Bernoulli configuration with an appropriately chosen parameter p to match the local slope of the level curve of u. The propagation speed for such initial configurations is given by (2) and a straightforward coupling argument implies that u satisfies the Hamilton-Jacobi equation locally.

The classic theory, as it relates to variational problems and partial differential equations of first order (see, e.g. [4, Section 9.46]), implies that any function which locally solves

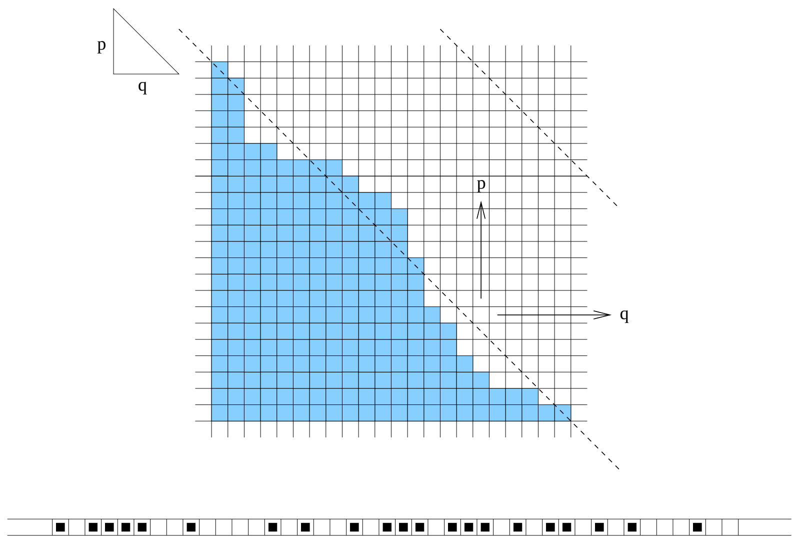

Figure 6: Boundary of the tiled and untiled regions and its mapping to the particle process. In this particular case $p = q = 1/2$. When the corresponding particle process is in steady state (Bernoulli distribution of particles), the boundary moves to the right at a long-term average speed of q (by symmetry, the vertical speed is p). The angle of the boundary remains the same.

$H(u_x, u_y) = 0$ can be represented also as the extremal action function extremizing the following functional

$$\int L\left(\frac{d\xi}{dz}, \frac{d\eta}{dz}\right) dz,$$

where L is the Legendre dual to H, i.e.,

$$L(x, y) = \inf_{(\xi,\eta):\, H(\xi,\eta)=0} (x\xi + y\eta).$$

Evaluating the infimum for the particular choice of $H(\xi, \eta) = \xi^{-1} + \eta^{-1} - 1$ (see (3)), one obtains

$$\begin{aligned} L(x, y) &= x\frac{\sqrt{x} + \sqrt{y}}{\sqrt{x}} + y\frac{\sqrt{x} + \sqrt{y}}{\sqrt{y}} \\ &= (\sqrt{x} + \sqrt{y})^2. \end{aligned}$$

Hence, we have established the computational time in terms of a variational problem:

Theorem 1. *The time $C_{\lambda\mathcal{D}}$ required to complete computation on a DNA computer of shape $\lambda\mathcal{D}$ is given by*

$$\lim_{\lambda\to\infty} \lambda^{-1} C_{\lambda\mathcal{D}} = \sup_{\gamma} \int \left(\sqrt{\frac{d\xi}{dz}} + \sqrt{\frac{d\eta}{dz}}\right)^2 dz, \tag{4}$$

where the supremum is taken over all piece-wise smooth increasing curves $\gamma = (\xi(z), \eta(z)) \in \mathcal{D}$, $0 \le z \le 1$, starting at the input region and ending at the output region of $\mathcal{D}$.

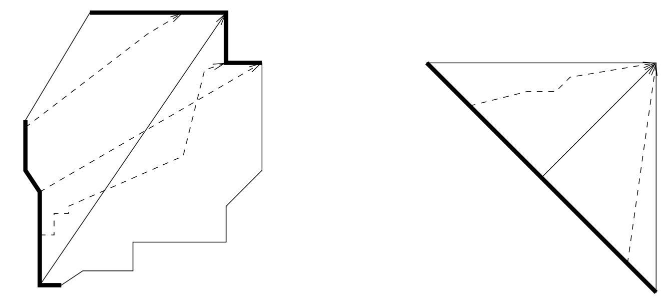

Figure 7: Two examples of 2D DNA-based computational devices. The one on the right corresponds to the one proposed in [7]. A few piece-wise linear paths from the input areas to the output areas are shown. The paths that determine the computational times (maximize the integral in (4)) are shown with solid lines.

Remark 1. It is easy to prove that the parts of the extremals *in the interior* of $\mathcal{D}$ are straight-line segments; see Figure 7 for examples.

Remark 2. The construction of the function L implies that the functional (4) is parametrization invariant. That is, it depends only on the trajectory $\{\gamma(z)\}_{0\leq z\leq 1}$.

4 Concluding remarks

This result allow one to determine the hydrodynamic limits of the computation times for 2D DNA-based logical devices. The fluctuations around the hydrodynamic scaling can in fact be estimated. In the particular situation of homogeneous attachment rates, these fluctuations can be studied using combinatorial methods and the powerful machinery of Riemann-Hilbert problems [9, 14]. In more general settings, less precise yet more robust methods are applicable (see, e.g. [16]).

Next we discuss the case when attachment times are not identically distributed exponential random variables, i.e., attachment rates are *nonhomogeneous* and/or *non-Markovian.* In such a case, coupling methods ([10]) allow one to prove once again a convergence of the computation-time function u, in the hydrodynamic scaling, to a solution of a homogeneous first order PDE. However, recovering exactly the structure of the PDE appears to be out of reach. Nevertheless, the Markovian setting allows one to bound the corresponding Hamiltonian H if the random attachment times can be bounded by exponential variables. These bounds can also be used, evidently, to put some a priori bounds on the solutions of the PDE. In general, in the hydrodynamic limit, the computation time could be expressed as the extremal value of the following functional

$$\int L\left(\xi,\eta,\frac{d\xi}{dz},\frac{d\eta}{dz}\right)\,dz.$$

Although, the analytic form of L might not be known, the functional can, in principle, be estimated by simulation. In Figure 8 we show simulation results for two attachment times.

We point out that there is a clear analogy with geometric optics, since we look for the extremals of a Lagrangian that is homogeneous, and of degree-1 in velocities. The space in

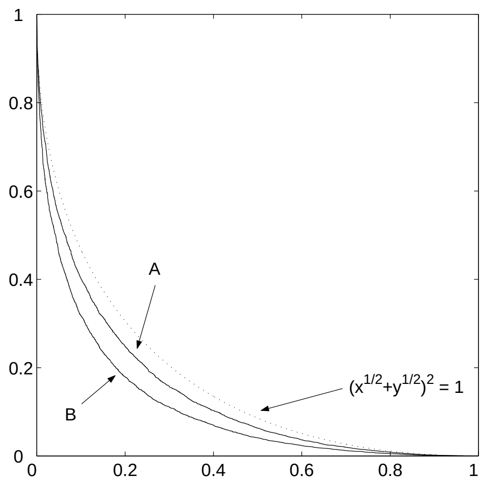

Figure 8: Estimated forms of the curve $L = 1$ when attachment rates are not exponential random variables. Attachment times are equal in distribution to $(1-\alpha)T1_{\{B=0\}} + \alpha T1_{\{B=1\}}$, where B is Bernoulli with parameter 1/2 and T is exponential of unit mean. The parameter α is set to 0.5 (A) and 0.9 (B). The curve $(\sqrt{x}+\sqrt{y})^2 = 1$ shown with the dashed line corresponds to the case of homogeneous attachment times ($\alpha = 1/2$).

which the extremals are being sought consists of all (piece-wise) curves connecting boundary manifolds. The term *reverse-optic law* serves as a mnemonic for remembering the expression for computation times. In particular, when considering the computation of a *single* bit one has:

$$\begin{array}{ll} \text{DNA computing} & \sup \int \left(\sqrt{\frac{\partial \xi}{\partial z}} + \sqrt{\frac{\partial \eta}{\partial z}} \right)^2 dz \\ \text{Geometric optics} & \inf \int \sqrt{\left(\frac{\partial \xi}{\partial z}\right)^2 + \left(\frac{\partial \eta}{\partial z}\right)^2}\, dz \end{array}$$

Finally, we note that the idea of using *max-plus* algebra in software and hardware design appeared earlier (see [13, 8] and references therein). However, the extension of this circle of ideas to the case of *stochastic* local propagation, crucial for the analysis of DNA computations, is, to our knowledge, new.

References

[1] L. Adleman, Q. Cheng, A. Goel, and M.-D. Huang. Running time and program size for self-assembled squares. In *Proc. ACM Symp. Th. Comput.*, pages 740–748, 2001.

[2] L. Adleman, Q. Cheng, A. Goel, M.-D. Huang, D. Kempe, P. Moisset de Espanés, and P. Rothemund. Combinatorial optimization problems in self-assembly. In *Proc. ACM Symp. Th. Comput.*, pages 23–32, Montreal, Canada, 2002.

[3] L. Adleman, Q. Cheng, M.-D. Huang, and H. Wasserman. Linear self-assemblies: Equilibria, entropy, and convergence rates. In Elaydi, Ladas, and Aulbach, editors, *New progress in difference equations*. Taylor and Francis, London (to appear).

[4] V.I. Arnol′d. *Mathematical methods of classical mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1997. Translated from the 1974 Russian original by K. Vogtmann and A. Weinstein, Corrected reprint of the second (1989) edition.

[5] Y. Baryshnikov, E. Coffman, and P. Momčilović. Incremental self-assembly in the fluid limit. In *Proc. 38th Ann. Conf. Inf. Sys. Sci.*, Princeton, New Jersey, 2004.

[6] Y. Baryshnikov, E. Coffman, and P. Winkler. Linear self-assembly and random disjoint edge selection. Technical Report 03-1000, Electrical Engineering Dept., Columbia University, 2004.

[7] A. Carbone and N.C. Seeman. Circuits and programmable self-assembling DNA structures. *Proc. Natl. Acad. Sci. USA*, 99(20):12577–12582, 2002.

[8] J. Gunawardena. An introduction to idempotency. In *Idempotency (Bristol, 1994)*, volume 11 of *Publ. Newton Inst.*, pages 1–49. Cambridge Univ. Press, Cambridge, 1998.

[9] K. Johansson. Shape fluctuations and random matrices. *Comm. Math. Phys.*, 209:437–476, 2000.

[10] S. M. Kozlov. The method of averaging and walks in inhomogeneous environments. *Russ. Math. Surv.*, 40:73–145, 1985. Translation from Usp. Mat. Nauk 40 (1985) 61-120.

[11] T. Liggett. *Interacting Particle Systems*. Springer-Verlag, New York, 1985.

[12] T. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*. Springer-Verlag, Berlin, 1999.

[13] G.L. Litvinov and V.P. Maslov. The correspondence principle for idempotent calculus and some computer applications. In *Idempotency (Bristol, 1994)*, volume 11 of *Publ. Newton Inst.*, pages 420–443. Cambridge Univ. Press, Cambridge, 1998.

[14] N. O'Connell. Random matrices, non-colliding processes and queues. In *Seminaire de Probabilites XXXVI*, volume 1801 of *Lecture Notes in Math.*, pages 165–182. Springer, Berlin, 2003.

[15] P. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. In *Proc. ACM Symp. Th. Comput.*, pages 459–468, 2001.

[16] M. Talagrand. A new look at independence. *Ann. Probab.*, 23:1–37, 1996.

[17] H. Wang. Proving theorems by pattern recognition, II. *Bell System Technical Journal*, 40:1–42, 1961.

[18] E. Winfree. Complexity of restricted and unrestricted models of molecular computation. In R. Lipton and E.B. Baum, editors, *DNA Based Computing*, pages 199–219. Am. Math. Soc., Providence, RI, 1996.

[19] E. Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, Pasadena, 1998.