DNA computing based on chaos

DNA Computing based on Chaos

Gabriele Manganaro
Dip.Elettrico Elettronico e Sistemistico
Universita' degli Studi di Catania
V.le A.Doria, 6 - 95125 Catania, Italy
tel. +39-95-339535, fax +39-95-330793
email: man@dees.unict.it

Jose Pineda de Gyvez
Dept. of Electrical Engineering
Texas A&M University
College Station - TX77843-3128 - U.S.A.
tel. +1-409-845-7477, fax +1-409-845-7161
email: gyvez@pineda.tamu.edu

Abstract: In this paper a new approach for the realization of the DNA computing paradigm is presented. It exploits the natural richness of the chaotic dynamics to efficiently generate and process coded binary sequences following the DNA computing framework introduced by Leonard M. Adleman. The new method is discussed and some simulation results regarding the directed hamilitonian path problem are presented.

1. Introduction

In 1994 Leonard M. Adleman presented a revolutionary paper [1] in which an instance of the Directed Hamiltonian Path Problem was solved using DNA synthesis. Essentially, he proved that some computationally untractable combinatorial problems (NP-complete problems in particular) could be efficiently solved by exploiting DNA. In his synthesis procedure, DNA strands are used to represent information that can then be processed by using standard methods of molecular biology. This biological experiment originated the emergence of a new interesting interdisciplinary area of research known as "DNA computing". Most of the research in this field has been devoted to study computational models and to answer two fundamental open questions [3]: (1) Can every algorithm/program be simulated by a DNA algorithm? (2) Is it possible, to design programmable DNA computers able to run any arbitrarily given program?

The research hereby presented focuses on the efficient and unconventional implementation of the DNA computing paradigm using non von Neuman computational models. The need for unconventional computing models stems from the fact that many DNA operations rely on a combinatorial process among molecules. To effectively model a combinatorial processing engine and the associated DNA operations, a nonlinear chaotic dynamical system is proposed as the core engine of the search process. Conventional approaches generate randomly a coded sequence of bits and then make use of bit stream matching algorithms to find the sequence of bits that provides the solution. The power of this method is limited by the complexity of the search-matching operation and the search space. This translates into untractable memory usage and long computation times. In our approach, we use chaotic systems to search the space, and instead of using deterministic bit stream matching algorithms, the proposed nonlinear chaotic computing model adapts itself as the space search progresses narrowing it at the same time to find autonomously the optimal solution!

Chaotic dynamical systems exhibit a long term unpredictable behavior due to the so-called sensitivity to initial conditions. Furthermore, the so-called chaotic attractors are characterized by the existence of a dense set of unstable periodic orbits [5]. In our computing model the huge amount of information generated by the flow of a chaotic system [4,6] is exploited to represent potential candidate solutions of a combinatorial problem in analogy to biological DNA synthesis. This information is then processed by using a simple feedback scheme that tracks the dynamics of the system and acts on its inputs in order to select the desired solution among the generated candidates. The various blocks composing the feedback are based on the same principles of the test tubes used in the Adleman's approach [1-3].

2. Background

2.1 DNA computing

A DNA strand is composed by a sequence of units called nucleotides which are distinguished by the chemical group, or base, attached to them. The four bases are abbreviated by the letters A, G, C and T. These letters integrate an alphabet $\Sigma = \{A, G, C, T\}$ and are used to form strings to encode the information [1-3]. A DNA molecule represents a string in this alphabet. A multiset of strings can then be used to represent candidate solutions of an optimization problem such as the Directed Hamiltonian Path Problem. In practice, the actual solution (or one of the solutions) has to be selected among the many candidates. This selection

0-7803-3949-5/97/$10.00 ©1997 IEEE

process is accomplished by using standard methods of molecular biology. These biochemical operations can be synthesized into an abstract model of molecular computing. In particular, the use of test tubes has been formalized in the so-called *restricted DNA model* as follows[3]: a tube is a multi-set of aggregates over an alphabet Σ. Given a tube, the following operations can be performed:

1. *Separate*. Given a tube T and a symbol $s \in \Sigma$, two tubes can be produced:
   +(T,s) containing all the aggregates of T containing s
   -(T,s) all the remaining aggregates of T not containing s.
2. *Merge*. Given tubes T_1 and T_2, produce the new tube $\cup(T_1,T_2)=T_1 \cup T_2$.
3. *Detect*. Given a tube T, say 'yes' if T contains at least one aggregate, otherwise say 'no'.

The above three operations allow the processing of the DNA strands and so they permit to realize well defined algorithms using them as the elementary instructions. A device capable of performing these algorithms is considered a molecular computer.

2.2 Chaotic systems

Some definitions are needed for the following presentation[4]. Let us define an nth-order *nonautonomous* dynamic system by the time-varying state equation:

$$\dot{x} = f(x,t) \qquad x(t_0) = x_0 \tag{1}$$

where $\dot{x} = dx/dt$, $x(t) \in \Re^n$ is the state at time t and $f:\Re^n \times \Re \rightarrow \Re^n$ is called the vector field. The solution of (1) with initial condition x_0 is called a *trajectory* and is denoted by $\phi_t(x_0)$. The mapping $\phi_t:\Re^n \rightarrow \Re^n$ is called the *flow* of the system. The dynamic system (1) is linear if and only if (iff) $f(x,t)$ is linear with respect to x, otherwise it is nonlinear. Finally, if f is a function of x only (namely it is independent of t) the system (1) is an *autonomous* system.

The steady-state behavior of nonlinear dynamic systems can be quite complex depending on the form of f and the value of its parameters. The main features characterizing chaos are:

1. the trajectories are bounded and are not periodic;
2. the steady state trajectories tend to a geometrical object called *attractor* whose dimension is not integer (it is a so-called fractal object);
3. the trajectories emanating from any two initial points, arbitrarily close one to another, diverge at a rate characteristic of the system until, for all the practical purposes, they are uncorrelated.

The last feature is the so-called sensitivity to initial conditions and it is the one that mostly makes chaos interesting; in fact, in practice the initial state of the system can never be specified exactly, but only to within some tolerance. Therefore, two or more states whose difference is less than that tolerance are practically not distinguishable. However, their long term values are different because of property (3). It follows that even if the dynamic system is completely deterministic it is not possible to predict its long term behavior.

The inherent randomness of chaotic systems along with some of the other features above recalled will be the strength points of our approach.

2.2.1 The Lorenz system

A classic example of chaotic systems is the *Lorenz system*, described by the following state model [6]:

$$\begin{aligned} \dot{x} &= -\sigma x + \sigma y \\ \dot{y} &= Rx - y - xz \\ \dot{z} &= -bz + xy \end{aligned} \tag{2}$$

where x, y and z are the state variables while σ, R and b are the system parameters. This is an autonomous system because it is independent from t. If $\sigma=10$, $R=28$ and $b=8/3$ then the system (2) will have a chaotic behavior. The steady state behavior is characterized by a double-lobe attractor in the three-dimensional state space whose projection on the x-y plane is shown in Fig. 1.

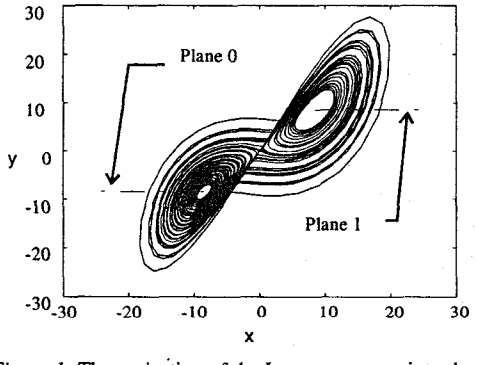

Figure 1: The projection of the Lorenz attractor into the x-y plane; the two section planes 0 and 1 cutting the trajectories in correspondence of two of the equilibrium points are reported.

2.2.2 Poincare' maps and symbolic dynamics

Let us consider an nth-order autonomous chaotic system, e.g. the Lorenz system (in this case n=3). Let x* be a point belonging to a trajectory and consider an (n-1)-dimensional hyper-plane transverse to the

trajectory at x*. The trajectory emanating from x* after a certain time will encounter again the section plane in another point and so on. The iterative function defining this sequence of points into the section plane is called the *Poincare' map* (P.M.) and constitutes a discrete-time representation of the continuous time chaotic system [4]. In Fig. 1 two possible Poincare' sections (perpendicular to the plane of the figure) labeled plane 0 and 1 are shown. In particular these are the two half-planes defined by the equations:

$$y = \pm\sqrt{b(R-1)}\,;\ |x| \geq \sqrt{b(R-1)}\,; \quad (3)$$

Let us associate two binary symbols 0 and 1 to the two planes 0 and 1, respectively. Consider an initial condition x^* on one of these two planes, and then keep track of the next intersections between the state trajectory emanating from x* and the two Poincare' sections. In this way a sequence of binary digits $\{b_n\}$ is defined by the initial condition x*. This sequence, associated to x*, is completely deterministic and it is called the *symbolic dynamics* of the system[6]. It can be represented by the binary fraction:

$$r = \sum_{n=1}^{\infty} b_n 2^{-n} \quad (4)$$

where r is called the *symbolic state* [6]. The function *r(x)* is called the *coding function* and can be obtained experimentally by measurements on the free-running system [6]. It can be quite complex depending on the vector field and/or the Poincare' sections. The one obtained for the Lorenz system with the above two Poincare' sections is smooth enough and its inverse, relative only to variable x, is shown in Fig. 2.

Finally, it has to be noted that certain sequences of bits do not appear if the system is allowed to evolve without external perturbations. The rules specifying allowed and disallowed sequences are called the *grammar* [6].

2.2.3 Symbolic dynamics and external perturbations

Let us now consider the symbolic dynamics (constrained to N bits):

$$\{b_1, b_2, b_3, \ldots, b_N\} \quad (5)$$

associated to a point x_1 on one of the P.M. and let the system run until the trajectory $\phi_t(x_1)$ intersects again one of the two P.M. in a new point x_2. It follows that the symbolic dynamics of x_2 will be:

$$\{b_2, b_3, \ldots, b_N, b_{N+1}\} \quad (6)$$

For the same reason, the next intersection x_3 will correspond to the symbolic dynamics:

$$\{b_3, \ldots, b_N, b_{N+1}, b_{N+2}\} \quad (7)$$

and so on. In conclusion the binary string corresponding to a point following another one in an unperturbed evolution is obtained by doing a circular shifting.

However, if the system is perturbed by an external signal u, while going from x_1 to x_2, then the trajectory will change and the next intersection will be on $x_2^* \neq x_2$ with new symbolic dynamics:

$$\{b_2, b_3, \ldots, b_k, b_{k+1}, \ldots\} \quad (8)$$

The length of the sequence is still N bits but only the first k bits are the same of the symbolic dynamics shown in (6). Of course the more intense the perturbation the lower will be k (it can even be k=2). Moreover, due to the sensitivity to initial conditions it follows that small perturbations are able to strongly change the symbolic dynamics.

3. Chaos for DNA computing

In this section a completely new approach to DNA computing is discussed. The richness of the chaotic dynamics and the sensitivity to initial conditions are exploited to generate information while a feedback scheme is used to process it.

3.1 Coding information into the chaos

From the above discussion it turns out that it is possible to associate a binary string to each point of the Poincare' maps (P.M.). Any binary string corresponding to any of these points describes the sequence of intersections of the trajectory with the two planes.

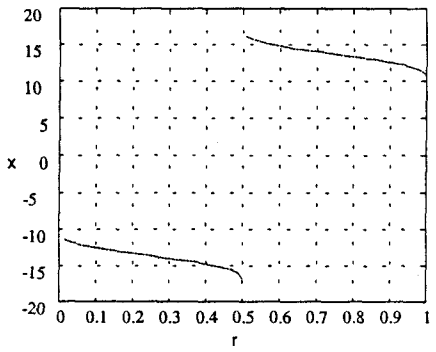

Figure 2: The inverse coding function for the Lorenz system with the plane 0 and 1.

In section 2.1 it has been recalled how the 4 bases of DNA can be used to code information into the DNA strands. Analogously strings of bits can be used to code information into binary words using a chaotic system[6,8]. For instance, let us suppose that the 26 letters of the alphabet have to be coded by using the

two PM and a Lorenz system. This means that any symbol (letter) needs 5 bits (e.g. the letter C can be coded into 00011). Therefore, a binary symbolic dynamics with N=5 bits can be considered. For this example the binary fraction is r=0.00011, i.e. the decimal fraction r=0.09375. By using the coding function of Fig. 2, it turns out that the letter C is coded into a point into the plane 0 since r=0.09375 < 0.5.

Finally, multiple points in the PM lead to the same future symbol sequence due to the limited number of bits considered for the symbolic dynamics. However, due to the smoothness of the coding function of Fig. 2 this means that all the points corresponding to the same symbolic state belong to the same contiguous common interval (and as such they can be considered indistinguishable).

3.2 Chaos and test tubes

In the previous paragraphs it has been seen that it is possible to generate information by coding it into the points of the PM. The generation of new information is due to the sensitivity to the initial condition and this phenomenon can be even strengthen by means of weak perturbations. All those PM points are analogous to the DNA strands contained in a test tube. Hence, the chaotic system can be viewed as a test tube containing the information to be processed.

However, the DNA algorithms permit to arrive to the solution of a problem by manipulating the content of the tubes by means of the three fundamental operations described in the restricted DNA model above discussed. Hence, analogous manipulations will be now defined for our chaotic tube.

Given the above duality between chaotic systems and tubes the two terms and the associated formalism will be now used interchangeably.

3.2.1 Feedback and the restricted DNA model

Let us consider the operation *separate* and a free-running chaotic system T. Note that in order to create the negative tube $-(T,s)$ it is necessary to discard all the points in the PM that correspond to symbolic dynamics containing the sub-sequence s. Moreover, if T generates a point x_1 whose symbolic dynamics contains s, then another point x_2 from the free running dynamics most likely still contains s (except for the particular case in which the first bit of s is also the first bit of the symbolic dynamics of x_1). Therefore T has to be perturbed so that:

1. the undesired sub-sequence s is removed;
2. a complete different information is generated.

This means that a *control unit*, that senses the state variables, has to perturb the state of T, when the trajectory encounters a PM, upon the following rule:

$$\text{if } s \text{ is a part of } r(x_1) \text{ then perturb T} \qquad (9a)$$

We will call this unit *negative-tube feedback block*. It easily follows that to obtain the positive tube $+(T,s)$ the antecedent of this rule has to be negated, leading to the rule:

$$\text{if } s \text{ is NOT a part of } r(x_1) \text{ then perturb T} \qquad (9b)$$

This defines the *positive-tube feedback block*. The (weak) perturbation of T has not been specified yet. In fact a well determined perturbation is not needed and it is even undesirable because the new information is partially a function of this perturbation. Another chaotic system uncorrelated to the state of T can then be used as a source of perturbations.

The block diagram of Fig. 3 depicts the realization of the positive tube.

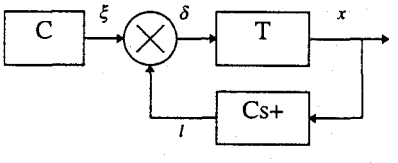

Figure 3: Block diagram for the realization of $+(T,s)$.

T is the chaotic system with state vector x. The state is sensed by the feedback block $Cs+$ that executes rule (9) when x encounters the PM. If the result is "perturb T" then the logic variable l is set to 1 (otherwise it is set to 0). This is multiplied by signal ξ coming from the independent chaotic system C so that the perturbation $\delta=l\cdot\xi$ is applied to T:

$$\dot{x} = f(x) + \delta \qquad (10)$$

The whole system realizes the *separate operation* $+(T,s)$. To accomplish the *merge* operation a logic composition of feedback blocks analogous to $Cs+$ is used. In fact, consider two tubes T_1 and T_2 obtained by adding suitable feedback blocks $Cf1$ and $Cf2$ to tube T containing all the DNA strands. If $\cup(T_1,T_2)=T_1\cup T_2$ has to be realized, then the merge is obtained by the parallel connection of the two feedback blocks $Cf1$ and $Cf2$ and composing the associated logic output by means of a simple logic AND gate as shown in Fig.4. This is easily verified by considering the rules corresponding to the independent feedback blocks and by application of the De Morgan theorems. The

merge operation can be generalized to an arbitrary number of tubes and corresponding feedback blocks. The last DNA operation (*detect*) is trivially obtained by checking the total logic output of the feedback network.

With this approach a DNA algorithm is systematically transposed in the corresponding feedback network. This one will check the generated information and perturb or not the system T according to the logical evolution of the DNA algorithm. When no perturbation is applied it means that the information generated fits all the requirements and so this represent a solution to the problem. Finally it is worth noting that if a merge operation combines the logic results of the feedback blocks by using an AND gate, then, conversely, successive *seggregations* of a set of DNA strands corresponds to combining the outputs of the feedback blocks by using an OR gate.

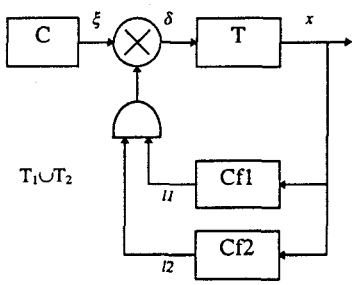

Figure 4. Merge between T_1 and T_2. through two control blocks

All of this is perfectly coherent with the Adleman approach in which, at the very end, the aim of all the biochemical manipulations (here translated into a suitable feedback network) is to filter the DNA string representing a solution among the many candidates.

4. Directed Hamiltonian Path Problem

The proposed approach is now applied to the solution the Direct Hamiltonian Path problem. A directed graph G is said to have a hamiltonian path if and only if there exists a sequence of compatible one-way edges that begins at v_{in} and ends at v_{out} and enters every other vertex exactly only once. The following non-deterministic algorithm was used by Adleman [1]:

1. Generate random paths through the graph
2. Keep only those paths that begin with v_{in} and end with v_{out}
3. If the graph has n vertices keep only those paths that enter exactly n vertices
4. Keep only those paths that enter all of the vertices of the graph at least once
5. If any paths remain, say YES; otherwise say NO

In order to apply the chaotic solver to the execution of this algorithm the candidate paths of the graph need to be coded into the Lorenz system. First, k-bit binary strings are assigned to each of the vertices of G (e.g. for 9 vertices 4 bits are needed). Because the graph has n vertices that have to be visited, then a *candidate solution* is a sequence of exactly n vertix codes. Hence, candidate solutions are strings of length $N=k*n$ bits which are inherently encoded into the N-bit symbolic dynamics of the chaotic tube T. Any time the chaotic system T intersects a PM a new random path passing through exactly n vertices of G is generated. This basically satisfies points (1) and (3) of the above non-deterministic algorithm.

Let us now consider point (2). Let c_{in} and c_{out} be the binary codes associated to vertex v_{in} and v_{out} respectively. A separate operation that eliminates the paths that do not have c_{in} and c_{out} as first and last codes is needed. This is obtained by means of two positive-tube feedback blocks Cin+ and Cout+. The first one senses the state x, evaluates the corresponding symbolic dynamics and then compares the first city's code with c_{in}. If they are equal its output is a logic 0 (this means that this particular constrain has been met and that it is not necessary to perturb T). Analogously, another positive-tube feedback block Cout+ is used to check the last city code.

To implement point (4) the remaining vertex codes c_2, c_3, ..., c_{n-2} are considered in the following operations: $+(T,c_2)$, $+(T,c_3)$,.., $+(T, c_{n-2})$. In this way the presence of all the vertices is checked. If there is at least one vertex missing then the corresponding feedback block will respond with a logic 1. The corresponding feedback block are C2+, C3+,.. in Fig. 5.

The outputs of these building blocks are combined by using a logic OR gate (as explained in Section 3 this corresponds to a successive seggregation). If the generated state corresponds to the solution of the problem, then no feedback block responds with a logic 1 and so no perturbation is applied. In all the other cases at least one block fires up and a perturbation is applied forcing T to cancel the current symbolic dynamics and to generate another candidate.

The solution we are addressing holds for both *complete symmetric* and *asymmetric* graphs. A *complete symmetric* graph has both edges (v_i, v_j) and (v_j, v_i) for every two distinct vertices v_i and v_j.; a *complete asymmetric* graph has only one.

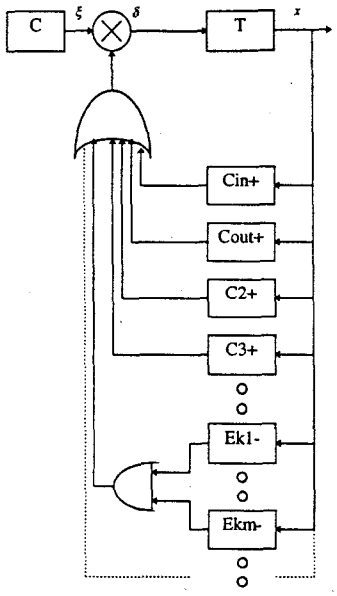

Figure 5: Block diagram for the directed hamiltonian path problem.

Therefore, additional feedback blocks are needed to check that all the edges involved in the current path are really existing in the given graph G. To this aim, a feedback logic similar to the one used for checking the presence of all the vertices is adopted. This corresponds to blocks *Ek1-*, .. , *Ekm-* shown in Fig. 5. Some results obtained from simulations using Matlab are shown in Table. 1.

Table 1: Simulation results for the Directed Hamiltonian Path problem.

Edges	Vertices	C	RT
4	3	59.7	11.94ms
6	3	57.7	11.54ms
9	4	333.6	66.72ms
11	4	81.8	16.36ms
16	5	217.5	43.5ms
30	6	15083	3.0166s

Several graphs with different number of edges and vertices were considered. C is the average number of candidates generated by the chaotic system prior to arriving at the solution. The average rate at which a new candidate is generated (considering a chaotic circuit oscillating with harmonic components in the acoustic range, e.g. 5KHz) is around every 0.2ms. Multiplying this by C provides us with an estimate of the running time if specialized hardware were in place. This has been reported in Table 1 as RT.

It can be noted that, although deeper investigation is needed, the more edges present in the graph, the faster is the convergence to the solution. This is a remarkable property if compared to conventional methods.

5. Conclusions

In this paper a new approach to DNA computing has been presented. It exploits the rich dynamics of chaos to simulate a test tube along with suitable feedback schemes for manipulating the contents of the tube. A systematic approach to obtain the feedback network corresponding to a desired DNA algorithm has been discussed. Finally, the new method has been applied to the Directed Hamiltonian Path Problem. Because of its features, the proposed setup represents a molecular computer in the sense introduced by Adleman [3]. Moreover, it can be implemented in real hardware. Preliminary results show that even at a moderate speed of operation the corresponding throughput is fairly remarkable.

6. References

[1] L.A. Adleman, "Molecular computation of solutions to combinatorial problems", *Science*, vol. 266, pp. 1021-1024, Nov. 11, 1994

[2] L. Kari, "DNA computing: the arrival of biological mathematics", to appear in *The Mathematical Intelligencer*

[3] L.A. Adleman, "On constructing a molecular computer", in *DNA Based Computers,* Eds. R.Lipton and E.Baum, DIMACS: series in Discrete Mathematics and Theoretical Computer Science, *American Mathematical Society*, 1996

[4] T.S. Parker and L.O. Chua, "Chaos: a tutorial for engineers", Proceedings of the IEEE, vol.75, no. 8, pp. 982-1008, 1987

[5] M.J. Ogorzalek, "Taming Chaos - Part I & II", IEEE Trans. On Circuits and Systems - Part I, vol. 40, no. 10, pp. 693-706, 1993

[6] S. Hayes and C. Grebogi, "Coding information in the natural complexity of chaos", SPIE 2038, Chaos in Communications, 1993

[7] R. Pool, "A boom in plans for DNA computing", *Science*, vol. 268, pp. 498-499, Apr. 28, 1995

[8] S. Hayes, C. Grebogi, E. Ott, "Communicating with chaos", Physical Rev Letters, vol.70, no.20, pp. 3031-3034, 1993