Computing with DNA

Eric B. Baum
NEC Research Institute
4 Independence Way
Princeton NJ
eric@research.nj.nec.com
http://www.neci.nj.nec.com:80/homepages/eric/eric.html

Unfortunately, drawings omitted from this posted version.

Introduction

DNA designed by Evolution to handle information.

The "purpose" of life is to process the info in DNA. Many tricks evolved for this. Enzymes, repair, etc.

Recently Molecular Biologists have uncovered many of these, and learned how to apply them.

(Including some tricks not known to nature? e.g Meade and Kayyem.)

Use tricks to do computing on a molecular scale

- Parallelism of order 10^{20} ?
- Memory of order $\geq 10^{20}$ Bytes?

Brief History

(1) Adleman, (Science,11/11/94 pp 1021-1024)

•Proposed DNA based computing.

•Physically solved a small graph problem (HPP).

(2) Lipton, (Talk last December, Science 4/28/95)

• Gave a different, more practical algorithm.

(3) EBB (last December, Science 4/28/95)

•Proposed DNA based Associative Memory.

(4) DIMACS Workshop 4/4/95

8 talks, including

Boneh: Decrypting DES,

Smith: DNA Turing machine.

• Proceedings to Appear, published by AMS, Edited by E.B. and R. Lipton.

(5) Reif, DNA Turing machine, Formalized models, SPAA95

(6) I'm aware of at least 20 unpublished papers including:

Perhaps 8 proposals of DNA Turing Machines.
Several addressing potential problems, e.g. errors, physical limitations.
Algorithms for new problems.
Ideas about biology.

(7) Call for Papers for 2nd Annual Conference on DNA Computing...
http://www.neci.nj.nec.com:80/homepages/eric/eric.html
June 10-12 Princeton University.

Outline

(1) Review Lipton's proposal for SAT.

(2) Useful DNA Code

(3) Dynamic Programming

SATisfiability

Set of literals $\{x_1, x_2, ..., x_n, \bar{x}_1, \bar{x}_2, ..., \bar{x}_n\}$

SATisfiability: given a set of clauses in x_i, is there any assignment satisfying them?

$[x_1 \cup x_3 \cup \bar{x}_4]$; $[\bar{x}_2 \cup \bar{x}_4]$; $[x_1 \cup x_2]$

Canonical NP-complete problem.

Solution is either No, or some vector (1,1,0,0,...,1) which satisfies.

Strategy of Lipton's DNA solution

(1) Encode all 2^n vectors in DNA.

(2) Use Biological operations to extract those satisfying first clause.

(3) Of these, extract those satisfying 2nd clause. Etc.

(4) If any DNA left, sequence codes solution.

Encoding of Vectors

$\forall i = 1, ..., n$, pick 2 DNA sequences A_i, B_i.
A_1 might be ATATATATGG.
B_2 might be CCCGATCAGC

Create all sequences having A_i or B_i for all i, but not both. Interpret as Boolean vectors.
$A_1, B_2, A_3 \Leftrightarrow (1, 0, 1)$
Every Boolean vector $\Leftrightarrow$ A Sequence

(1) Can create all such sequences in testtube.
E.g. by mixing subsequences for edges and splinting sequences.

CGATCGATCG|ATCAAATTTC
CTAGC|TAGTT

(2) Biological operations can be used to extract from testtube molecules coding for “1” in the i-th position.

To do:

(1) create sequences $\bar{A}_i$. Affix to magnetic bead.
(2) Introduce to testtube. Will stick to sequences containing A_i.
(3) Extract magnetically.

Say have $[x_1 \cup x_3 \cup \bar{x}_4]$; $[\bar{x}_2 \cup \bar{x}_4]$; $[x_1 \cup x_2]$

(1) Extract from t molecules containing $A_1 \to t_1$.
(2) Extract from t molecules containing $A_3 \to t_2$.
(3) Extract from t molecules containing $B_4 \to t_3$.
(4) Pour t_1, t_2, t_3 together $\to t$. Go to next clause.

• Lipton's procedure solves SAT problem in O(N) time, if $n < 60$.

• Improved algorithms solve Circuit Sat (Boneh et al.)

• What problems can be solved efficiently with appeal to a fast oracle for small, 60 input, circuits?

Associative Memory

Store words $\vec{x}^i \in \{1,0\}^n$ where $i = 1, ..., M$.
Given some of bits (possibly also some errors) find best match.
So, say store:

0,0,0,0,0,0,0,0,0,0
1,1,0,1,0,0,1,0,1,1
0,0,0,1,1,0,0,1,0,0

Given input cue,say:

,1,,1,*,*,*,*,*,*

would like to retrieve and read second stored word.
Human memory said to have this characteristic...

Straightforward to do with DNA:
•To store word, place appropriate word in test tube.
•To read content addressably:
(1) Attach magnetic beads to sequences matching components in cue,
(2) Extract.
(3) Sequence.

Capabilities

- Capacity- currently attainable complexities of 10^{13} - 10^{14} molecules in a test tube. Might get 10^{20} in 1000 liters.

- Read time: Days?

- Can in principle copy whole memory fairly rapidly...

Key questions

(1) **Can all this really scale up?**

Potential problems include:

(a) Errors of many types

(b) DNA Hydrolisis

(c) Annealing slowdown in large systems

Computer Scientists are good at schemes for computing in noisy environments.

So, a variety of proposals for dealing with such problems ongoing...

(2) **What Killer Ap's?**

Chasing a moving horizon, if takes 5 years to get working, and have to give up big factor, may not beat tomorrow's supercomputers by much...

DNA Sequences Useful for Computation

Lipton/Adleman code with "random" DNA subsequences.
These will have overlap on subsequences, causing gumming or error in retrieval.

Want set of subsequences X_i, spacer S_0, s.t.
No subsequence of length k of X_i appears in $S_0 X_j S_0$ and no length k subsequence of $S_0 X_i S_0$ has complement in $S_0 X_j S_0$.

Recall complement of $AGTCC$ is $GGACT$

Not clear what k acceptable, certainly smaller the better.
Some claims k as low as 5 already a problem.

Need minimum 120 sequences for Lipton...
But more sophisticated algorithms (e.g. DES) require 1000's of sequences.

Without loss of generality, consider k long sequences.
If have sequence S, can not have $\bar{S}$ so divide space into equivalence classes.

One set: $S_0 = AAA...A$
$\{X_i\}$ contains one rep from each equiv class not beginning nor ending with A nor T.
$4^{k-1}/2$ For $k = 5$ not enough...

Theorem: The maximum size set of strings satisfying conditions, for $k \geq 5$, $k = 2m$ or $k = 2m + 1$, is

$$3 \times 4^{k-2} + 5 \times 4^{k-4} - 2 \times 4^{m-1}$$

k=	4	5	6	7	8	9
N=	40	208	816	3232	13440	54144

Running Dynamic Programming Algorithms

Graph Connectivity

Breadth First Search solves in $O(|E|)$ time.

Define graph by set of fns

$$f_i : V \rightarrow V \ \forall i = 1 \ldots m$$

Given vertex v, neighbors $\{f_1(v), f_2(v), \ldots, f_m(v)\}$.
Let $C(f_i)$ be a circuit computing f_i.

DNA computer solves in time

$$d = \max\{\text{distance}(u, v) | u, v \in V\}$$

Sketch:

(1) Start with solution containing strands representing vertex S.

(2) Apply d times:

(a) Let D be current solution. Use PCR to make m copies.

(b) Compute circuits f_i. By appropriate extractions, have m solutions D_m representing neighbors of S.

(c) Pour all of these into D'.

(d) Test if target vertex in D'. If yes, done. If no, $D = D'$.

This algorithm works for graphs balanced in the sense that the number of paths from S to V is roughly independent of V. Define graph ϵ-balanced if ratio of number of paths least/most $> \epsilon$.

W_r = no. vertices V which have path of length r from S.
$W = \max\{W_r\}$.
Then algorithm works with high prob if initial solution contains $P = \frac{W}{\epsilon}\frac{d}{m}\log^2 W$ molecules representing S.

Note- can't implement more general types of DP yet.
Shows limitation of DNA- no communication between the different processors. So, e.g., can't remove redundant copies.

Example: **Knapsack Problem**:
Given positive integers $b_1, \ldots b_n$ and integer B, does there exist a subset $S \in \{1, \ldots n\}$ s.t. $\Sigma_{i \in S} b_i = B$.

Given solution T of strands representing numbers, we give algorithm for transforming to solution T' representing all numbers in T plus X, for X an m digit number.

Then start with 0 and apply addition on numbers $b_1 \ldots b_n$.
So first step, tube represents 0.
Then, contains strands for 0 and b_1.
Then, contains strands for $0, b_1, b_2$ and $b_1 + b_2$
...
Check if final solution contains B.

Note, at each step, constant number of biological operations does sum on all numbers represented in tube.

Restriction enzymes:

cut 2-strand DNA at a particular point.

May leave a sticky end.

Say use EcoRI enzyme, have strings:

$$\updownarrow S_i GAATTC \;\; \rightarrow \updownarrow S_i G \uparrow AATT$$

Append

All strings in tube end in GAATTC.

Break with EcoRI

Splint on sticky end a common string, ending in GAATTC.

This allows you to append to all strings in a tube a common sequence.

Circuit SAT: Given circuit, is there a solution?
Algorithm using 2^n DNA strands, linear time.

Start with test tube coding all 2^n n-bit sequences $x_1 ... x_n$.
Inductively produce tube containing all DNA strands $x_1 ... x_n y_1 ... y_k$ where y_k are values of first k gates.
Say next gate $y_{k+1} = y_j \cap y_i$
Extract strings marked $y_j = 1$.
Extract from these strings marked $y_i = 1$.
Mark these $y_{k+1} = 1$, others mark $y_{k+1} = 0$.

For example, can use this type trick to add to all numbers in a test tube.

Let T contain sequences representing numbers α_i.
Let X be integer with m decimal digits.
Want Solution T' rep'ing all numbers $\{\alpha_i, \alpha_i + X\}$

(1) Let d be least significant digit of X.
(2) Separate T into 10 sol'ns T_i s.t. T_i contains numbers with lsd i.
(3) In each T_i append Oligo rep'ing $i + d$ mod 10.
(4) Pour $T_0, ..., T_{9-d}$ in T^0. Pour $T_{10-d} ... T_9$ in T^1
(5) Iterate appropriately for each of the m digits of X, accounting for carries.
(6) At end of this process, strands look like:

$$3' - \alpha_i - S - \alpha_i + X - 5'$$

where S is some predetermined separator.
(7) Use restriction enzyme to cut all strands at S.

Conclusion

(1) 5 years from now, may see truly huge problems solved using DNA... but too early to say for sure...

(2) Call for Papers for 2nd Annual Conference on DNA Computing...
http://www.neci.nj.nec.com:80/homepages/eric/eric.html
June 10-12 Princeton University.