Computing with DNA

James A. Foster
Laboratory for Applied Logic
Dept. of Computer Science
University of Idaho

May 8, 1997

Outline (part one)

- Chemistry of DNA
- Polymerase Chain Reaction
- Brute Force Computing
- Finding Hamiltonian Paths

May 8, 1997 (jaf)

Outline (part two)

- A Mathematical Model
- Solving SAT
- P-DNA is PSPACE
- Potential and Limitations

Chemistry of DNA

DNA molecules: *paired strands* of *nucleotides* (*bases*) attached to sugarphosphate *backbones*

Nucleotides (bases): **A**denine binds with **T**hymine, **G**uanine binds with **C**ytosine

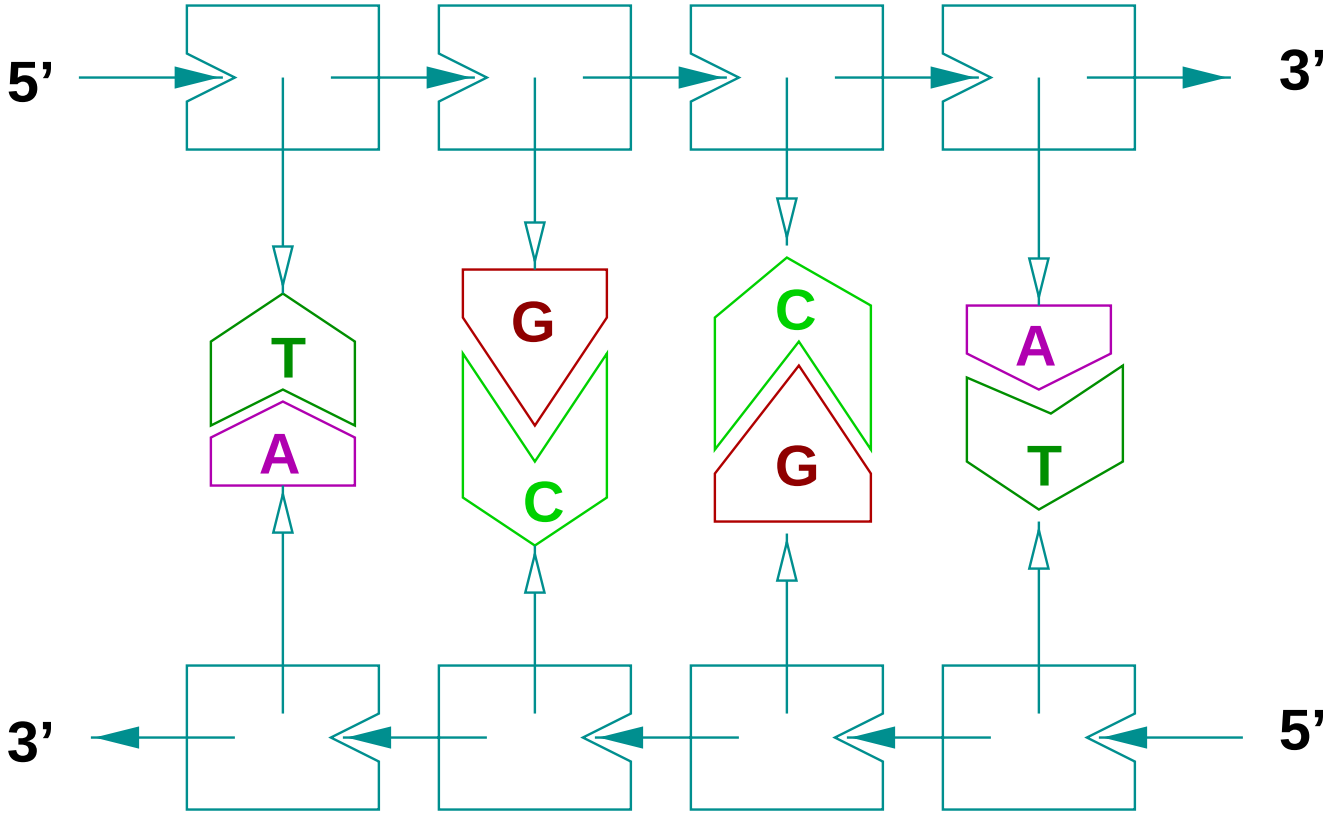
Backbone: 5 carbons, similar to linked list

- One molecule's $5'$ binds to next ones $3'$
- $1'$ binds to nucleotide

Paired Strands:

- Bases bond to complementary strand
- Sequences: listed $5'$ to $3'$

DNA Molecule Illustration



GCCA**TGCA**TTC | CGGT**ACGT**AAG

Note: $\overline{\texttt{GCCATGCATTC}}$ ($\overline{s}$ is complement of s) is CGGTACGTAAG

Polymerase Chain Reaction (PCR)

Given: collection of DNA and two *primers*, $\overline{s}, t$

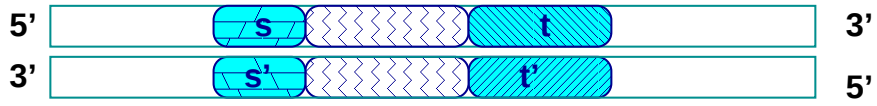
Action: *amplify* strands of the form svt for any sequence v

```
Input:  tube T of DNA, primers s̄ and t
Repeat until satisfied
   1) denature DNA with heat
   2) anneal DNA with primers
   3) elongate strands with DNA polymerase
```

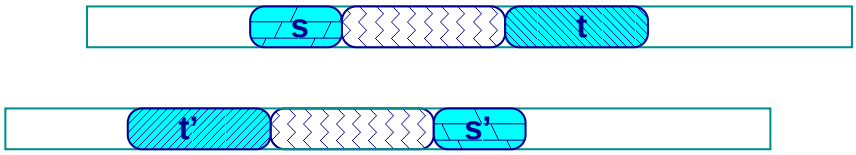
Note: copy number for target doubles each iteration

PCR Illustration

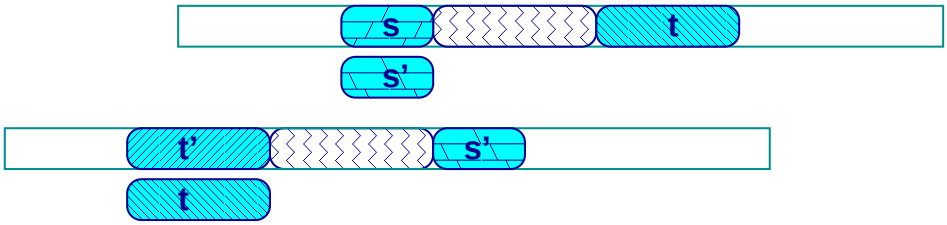
0) Given



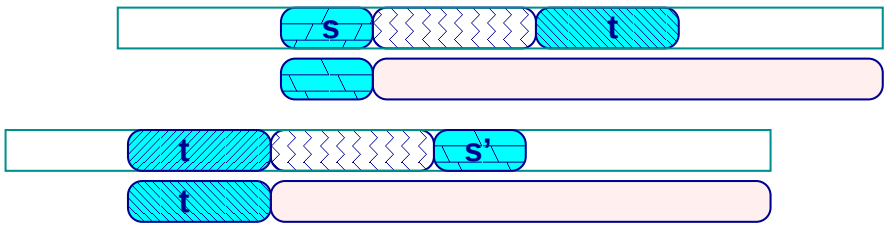
1) Denature (heat)



2) Anneal (Add primers)



3) Elongate (add polymerase)



Brute Force Computing

To solve problem P:

```
Represent input instance x as DNA
Represent possible solutions to P(x) as DNA
Make tube T
   with every possible solution to P(x)
Amplify positive results in T
Sample T to get answer
```

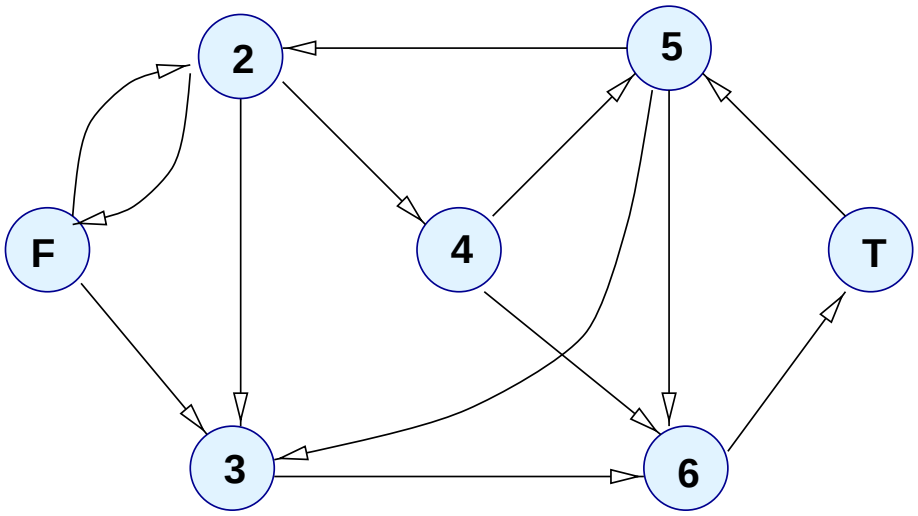
Recall: NP problems are easy to solve given a short "hint". This algorithm checks all possible "hints" in parallel, with a polynomial number of operations.

Example: Finding Hamiltonian Paths

The NP complete *directed Hamitonian Path (dHP)* problem:

Given: Directed graph G, nodes f, t

Question: Is there a directed path, visiting every node exactly once, from f to t in G?



A possible Hamitonian path: (F,2,4,6,3,5,T)

DNA Algorithm for dHP

Input: Graph G, nodes f and t

0) Represent nodes, edges, paths with DNA

1) Fill tubes with all possible paths

2) Select paths from f to t

3) Select paths of correct length

4) Select paths without duplicate vertexes

5) If anything remains

   Then return ‘‘yes’’

   Else ‘‘no’’

0: Representation

Node v: 20 random base pair sequence S_v

- Long enough not to bind to each other
- Short enough for PCR to work

Edge (u, v): build sequence S_{uv} with 10-base suffix from S_u, 10-base prefix from S_v (except use all of S_f and S_t)

Path $(a \cdot b \cdot c)$: catenate (a, b) and (b, c)

Example

Examples of Adleman's encoding

Nodes

S2: GTCACACTTC GGACTGACCT

S2': AGGTCAGTCC GAAGTGTGAC

S4: TGTGCTATGG GAACTCAGCG

S4': CGCTGAGTTC CCATAGCACA

S5: CACGTAAGAC GGAGGAAAAA

S5': TTTTTCCTCC GTCTTACGTG

Edges

(2,4): GGACTGACCT TGTGCTATGG

(4,5): GAACTCAGCG CACGTAAGAC

Paths (2.4.5):

GTCACACTTC GGACTGACCT TGTGCTATGG GAACTCAGCG CACGTAAGAC GGAGGAAAAA

AGGTCAGTCC GAAGTGTGAC CGCTGAGTTC CCATAGCACA TTTTTCCTCC GTCTTACGTG

1: Fill tube with all paths

Amplify tubes of S_x and $\overline{S_x}$ for each node x
Amplify tubes of S_{uv} and $\overline{S_{uv}}$ for each edge (uv)
Mix all tubes into tube T

Overlapping segments will bind and leave "sticky ends" to promote further binding

Example

Edge (2,4) GGACTGACCT TGTGCTATGG

Node 4' CGCTGAGTTC CCATAGCACA

With high probability, every possible path through G will be represented in T

2,3,4: Select candidate paths ($f \cdots t$)

Run PCR on tube T using S_F and $\overline{S_T}$ as primers, put products in T'

Separate strands with $20n + 10$ bases from T', put products in tube R

Note: by construction, nodes can be visited at most once (with high probability)

If any DNA is left in R, return "yes", else "no"

A Mathematical Model

Primitives: tubes of DNA (or similar)

Operations:

Remove $(T, T', \{\sigma_i\})$**:** Remove all strings in T of form $\tau\sigma_i\nu$, placing them in T'

Detect (T)**:** Decide if T has DNA in it

Mix $(\{T_i\}, T)$**:** Pour all T_is into T

Copy $(T, \{T_i\})$**:** Pour T into each T_i

We implicitly assume operations such as separation (by size), amplification, ligation, annealing, and denaturing where needed

Solving dHP

Representation: as before

Input: for each node v and edge (u, v),
T_v contains S_v (and $\overline{S_v}$)
T'_{uv} contains S_{uv} (and $\overline{S_{uv}}$)
Mix($\{T_i, T'_{uv}\}$,T)
Remove(T,T_0,$\{S_f\}$)
Remove(T_0,T',$\{S_t\}$)
Move length $20n + 10$ strings from T' to T''
if Detect(T'')
then return ``Yes''
else return ``No''

Complexity: linear in number of nodes (for Mix)

Note: S_v (and S_{uv}) are DNA strands representing node v (and edge (u, v)) in input graph

Listing Permutations

Problem: input n, list all permutations of n items

Representation: $p_1 i_1 p_2 i_x \ldots p_n i_n$ where p_j encodes "position j", $i_j \in \{1, 2, \ldots, n\}$

```
Input: T with all valid strings
for j = 1 to n
   Copy(T,{T_1, T_2, ..., T_n})
   for i = 1 to n
      for k = j + 1 to n
         Remove(T_i, J, {p_j ¬i, p_k i})
         // ¬i is any string other than i
      Mix({T_1, T_2, ..., T_n}, T)
   // first j is are distinct in each string
// T contains permutations of {1, 2, ..., n}
```

Requires $O(n^2)$ operations

Solving SAT

The Problem

Given: Boolean formula in CNF (p conjunctions, q literals) per clause, n variables

$$F(\vec{x}_n) = \bigwedge_{i=1}^{p} (\bigvee_{j=1}^{q} l_{i,j})$$

where $l_{i,j} = x_k$ or $\overline{x_k}$ for some variable x_k

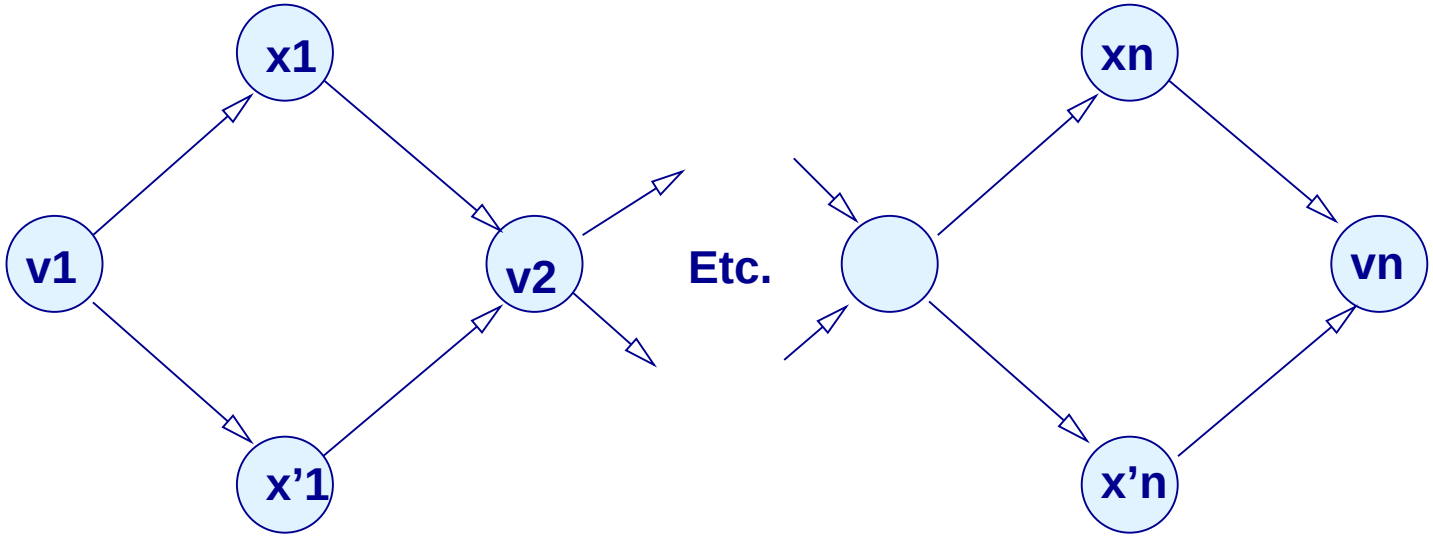
Question: Is there an $\vec{x}_n$ s. t. $F(\vec{x_n}) = T$?

Example:

$$\begin{aligned} F(\vec{x}_3) &= (x_1 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \\ G(\vec{x}_2) &= (x_1) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2}) \end{aligned}$$

F is satisfiable ($F(T,T,T) = T$), G is not

Representing Truth Assignments



Variables: x1, x2, ..., xn
Extra nodes: v1, v2, ..., vn
Paths: sequences of literals over these variables

DNA Algorithm for SAT

```
Input:  T_0 full of all truth assignment
Input:  Boolean formula F = ∧_{i=1}^{p} (∨_{j=1}^{q} l_{i,j})
for i = 1 to p
   for j = 1 to p
      if l_{i,j} is positive
      then Remove(T_{i-1}, T', {x_j})
      else Remove(T_{i-1}, T', {x̄_j})
      // Strands in T' will make clause j true
   Re-label T' as T_i
if Detect(T_p)
then return ``yes''
else return ``no''
```

Requires $O(n^2)$ operations

Computational Complexity

Let P-DNA be problems solvable with polynomial steps in this model

Thm (Beaver): P-DNA = PSPACE

Generalized Turing-complete models (splicing systems, DNA TMs) exist

But, P-DNA computations still require exponential volume, and perhaps lots of clock time

Disadvantages

"Steps" are manual (and slow)

Reaction time proportional to volume of reactants: real time can be (much) slower than number of steps

Required volume can be huge

Processes can introduce errors

Processes do not scale up well

Possible solutions to problems

Add active transport or catalyst to tubes

Build targeted solutions (forget brute force)

Compute on surfaces

Change molecules

Advantages

Massive parallelism

Attack special instances (e.g. Keller graphs for MC)

Very low energy consumption (10^{-19} J versus 10^{-9} J per basic operation) with no inherent lower bound

Way cool

Further Reading

L. Adleman, "Molecular Computation of Solutions to Combinatorial Problems", *Science*, 266:1021–1024, 1994

L. Adleman, "On Constructing a Molecular Computer", manuscript, *ftp://usc.edu/pub/csinfo/papers/adleman-/molecular_computer.ps*, 1995

D. Beaver, "A Universal Molecular Computer", Technical report, Penn. State U., 1995

J. Hartmanis, "On the Weight of Computations", *Bull. Euro. Assoc. for Theoretical Comp. Sci.*, 55:136–138, 1995

J. H. Reif, "Parallel Molecular Computation", in *Proc. 7th ACM Symp. on Parallel Alg. and Arch.*, pp. 213–223, 1995

Also see URLs from my homepage bookmarks