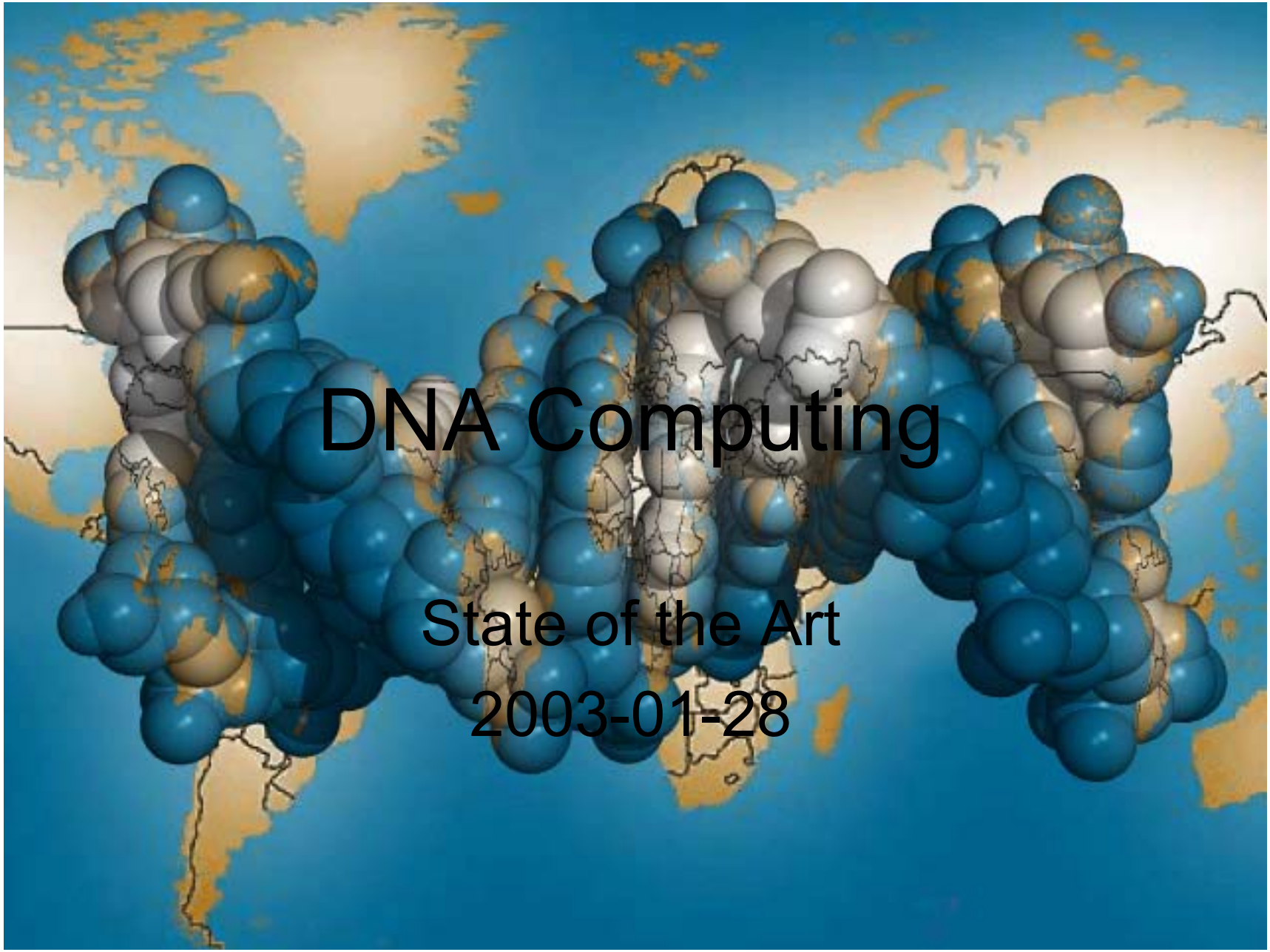


DNA Computing

State of the Art

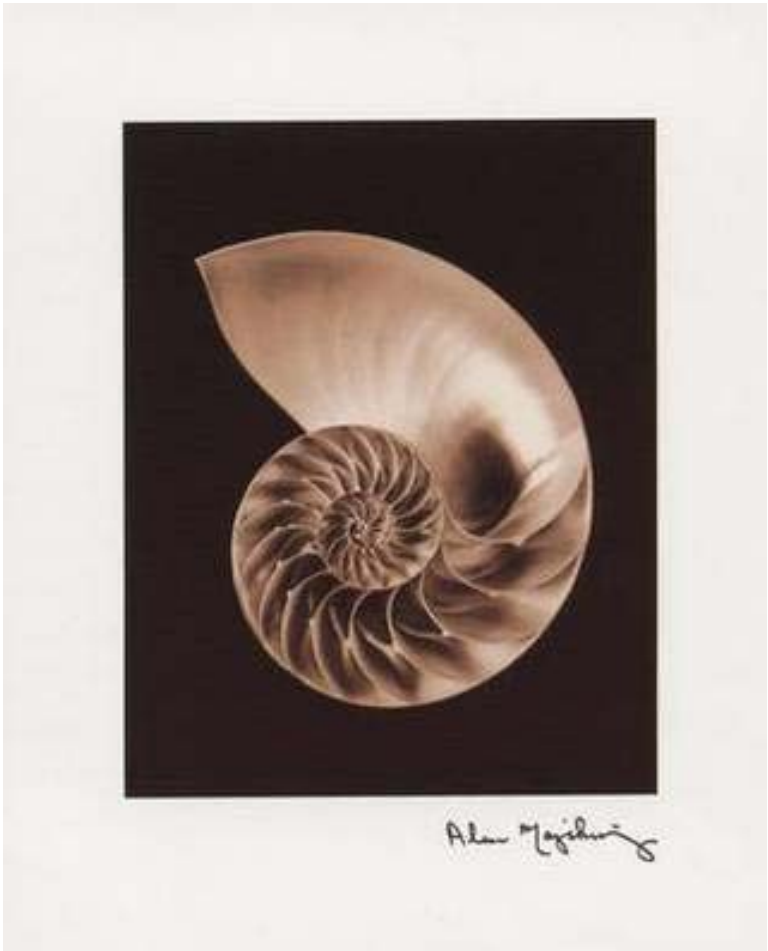
2003-01-28

CPSC 601.73 www.cpsc.ucalgary.ca/~omair/cpsc60173/Presentation

Interesting Facts

- DNA molecule is 1.7 meters long
- Stretch out all the DNA in your cells and you could reach the moon 6000 times!

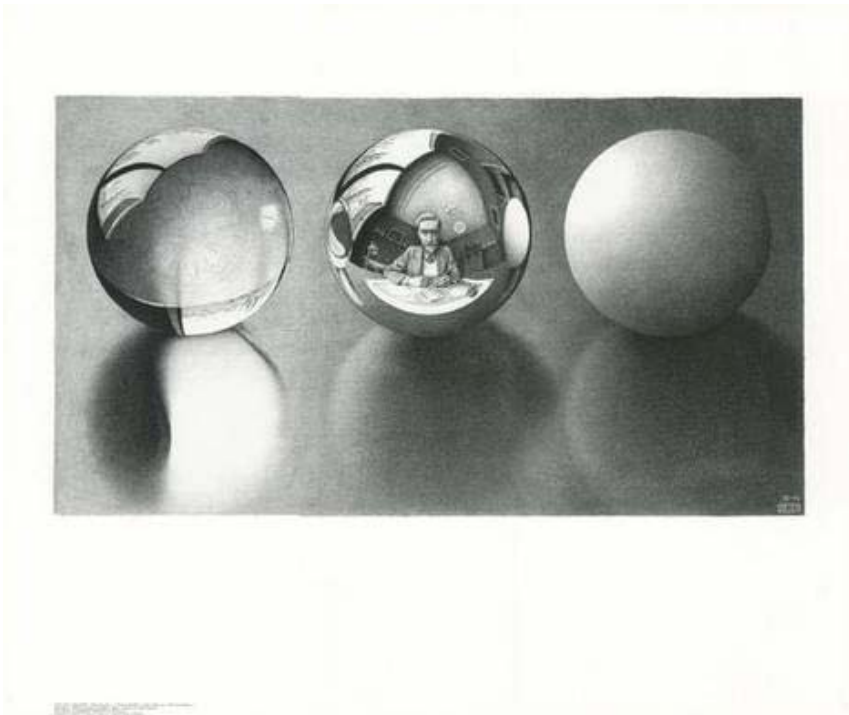
Interesting Facts



- DNA is the basic medium of information storage for all living cells. It has contained and transmitted the data of life for billions of years

Interesting Facts

- Roughly 10 trillion DNA molecules could fit into a space the size of a marble. Since all these molecules can process data simultaneously, you could theoretically have 10 trillion calculations going on in a small space at once



Leonard M. Adleman

- In 1994, Leonard Adleman took a giant step towards a different kind of chemical or artificial biochemical computer. He used fragments of DNA to compute the solution to a complex graph theory problem. Adleman's method utilizes sequences of DNA's molecular subunits to represent vertices of a network or `"graph". Thus, combinations of these sequences formed randomly by the massively parallel action of biochemical reactions in test tubes described random paths through the graph. Using the tools of biochemistry, Adleman was able to extract the correct answer to the graph theory problem out of the many random paths represented by the product DNA strands.

Richard J. Lipton

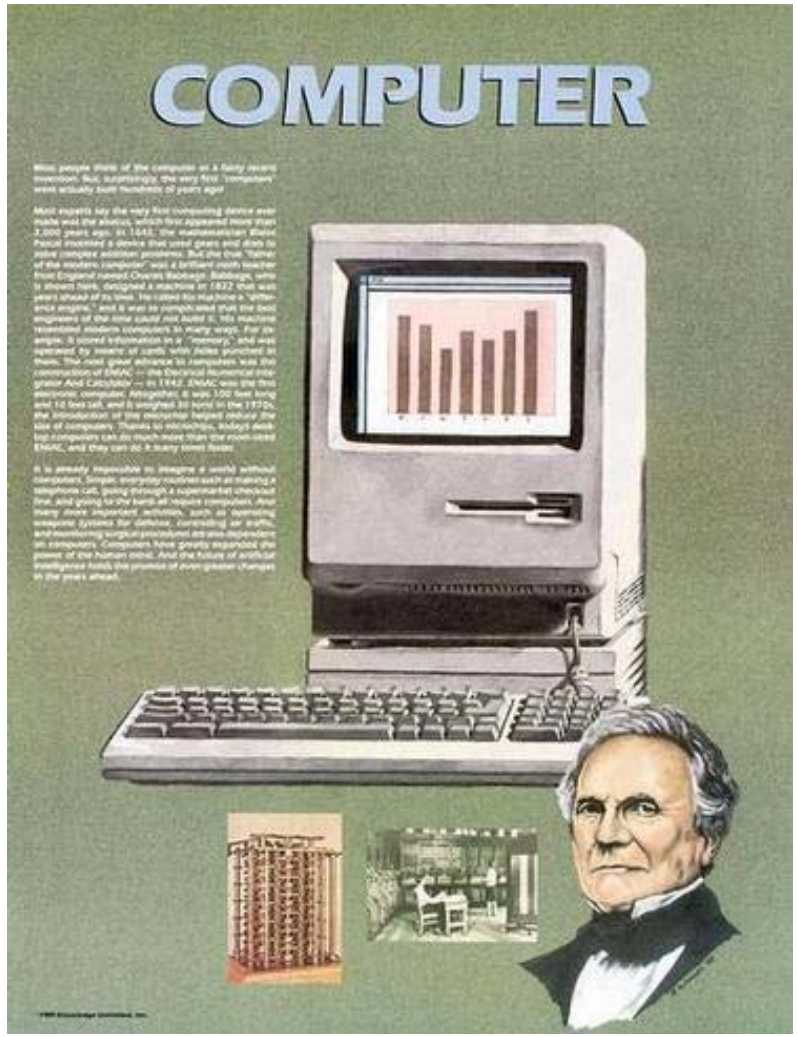
- Generalized to Satisfiability Problems
  - Problems that are NP Complete
  - A hard NP problem is one in which the time required for algorithms to find a solution increases exponentially with the number of variables involved. (In an easy NP problem, the algorithm running time increases in proportion to the number of variables.)

NP Complete Problems

- A hard NP problem can eat up a lot of computer cycles if carried out by brute force. For example, the Hamilton path problem —commonly known as the traveling salesman problem — is a hard NP problem. If there are N cities in a Hamilton path problem, there are N!/2 possible paths, where N! is N factorial, which is the multiplication of every integer from 1 to N — for example, 4!= 1 x 2 x 3 x 4.

NP Complete Problems

- As the number of cities grows, the number of possible path combinations soars. For example, if there are nine cities, there are 180,000 possible paths. Eleven cities would have 19.8 million paths, 13 cities would have about 3 billion paths, and 17 cities would have about 200 trillion paths. For larger and larger numbers of cities, brute force attempts to calculate all paths would quickly overwhelm even a supercomputer.



Example: Genetic Checkmate

- Princeton University (Dec 3, 1999)
- Using DNA Computing techniques solved a simple Knight Problem

The Knight Problem

- Generally what configurations of knights can one place on an n x n chess board such that no knight is attacking any other knight on the board?

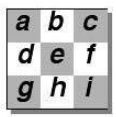
The Knight Problem

- Made use of an 10-bit RNA library
- Applied the problem to a 3x3 chessboard as a 9-bit instance of the problem.
- A true or “1” value represented by presence of a knight at position a-i
- Whereas false or “0” represents the absence of a knight at that position.

The Knight Problem

- we represent the problem as follows ($\wedge$, ''and''; $\vee$, ''or''):
  - $((\neg h \wedge \neg f) \vee \neg a) \wedge ((\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge$
  - $((\neg c \wedge \neg i) \vee \neg d) \wedge ((\neg a \wedge \neg g) \vee \neg f) \wedge ((\neg b \wedge \neg f) \vee \neg g) \wedge$
  - $((\neg a \wedge \neg c) \vee \neg h) \wedge ((\neg d \wedge \neg b) \vee \neg i)$.
- In this particular example, this simplifies to
  - $((\neg h \wedge \neg f) \vee \neg a) \wedge ((\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge$
  - $((\neg c \wedge \neg i) \vee \neg d) \wedge ((\neg a \wedge \neg g) \vee \neg f)$.
- This reduces the number of operations that one has to perform

The 'Knight Problem': *3* × *3* Board



$((\neg h \wedge \neg f) \vee \neg a) \wedge (\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge ((\neg c \wedge \neg i) \vee \neg d) \wedge (\neg a \wedge \neg g) \vee \neg f) \wedge$

$(\neg b \wedge \neg f) \vee \neg g) \wedge ((\neg a \wedge \neg c) \vee \neg h) \wedge (\neg d \neg b) \vee \neg i)$

The Knight Problem

- We are already familiar with the operations we are capable of performing on DNA namely:
- *merge*(N_1, N_2) = N
- N = *duplicate*(N_1)
- *detect*(N)
- separate/extract
  - $N \leftarrow +(N, w)$
  - $N \leftarrow -(N, w)$
- *length seperate*
  - $N \leftarrow (N, \leq n)$
- *position separate*
  - $N \leftarrow B(N_1, w)$
  - $N \leftarrow E(N_1, w)$

The Knight Problem

- In addition this particular problem was solved using RNA rather then DNA
- RNA was chosen because it is better suited for a *destruct* algorithm
- using Ribonuclease (RNase) H digestion, a destructive algorithm that would hydrolyze RNA strands that did not fit the constraints of a chosen problem

The Knight Problem

- RNA's 2' hydroxyl group makes it more prone to hydrolysis
- RNA strands can be marked for destruction by introducing complementary DNA oligonucleotides
- RNase H serves as a "universal RNA restriction enzyme".
- Another way to describe:
- So, if we want to destroy a string containing 0 on position a, we add a complementary DNA sequence that sticks to the targeted RNA sequence, and afterwards we introduce this RNase H enzyme to "clean" the solution.

- RNase H digestions destroyed the RNA strand of RNA DNA hybrids marked by hybridization of the pool RNA to the complementary bit oligonucleotides shown in Table 2.

Table 2. Nucleotide sequences for each DNA bit oligonucleotide used in the computation

	Hybridizes to bit set to 0		Hybridizes to bit set to 1
a_0	AGAATTGAGTAAGAG	a_1	TAAGTAATGTGAGGA
b_0	gttatTAAGAGGTTGATATG	b_1	gttatGGATATAAAGGAAGT
c_0	TGTGAAGTGGAGGAT	c_1	GGATGTTTGTTATAA
d_0	gtaaaGAGGGAAGATTTTAA	d_1	gtaaaTGAAGAGGGTTATGT
e_0	GGTGTGGATAAATAG	e_1	TATGGAAAGTAAGGT
f_0	GGAATTGTTTGAAGC	f_1	GTAGGGAGAATGTAC
g_0	attgaTTGAATTTGAGAGTT	g_1	attgaGAATATAAGATTATG
h_0	TGAAGTAAAGGTTAG	h_1	GAAGTATGTGATTAT
i_0	GTGGGATAAGGAATG	i_1	TAGGTAGTTGGTGGA

Twenty-base oligonucleotides are complementary to both the 15-nt bit sequence (uppercase) and adjacent 5-nt spacer (lowercase) to increase the melting temperatures of weaker hybridizing oligonucleotides.

The Knight Problem – RNA Library

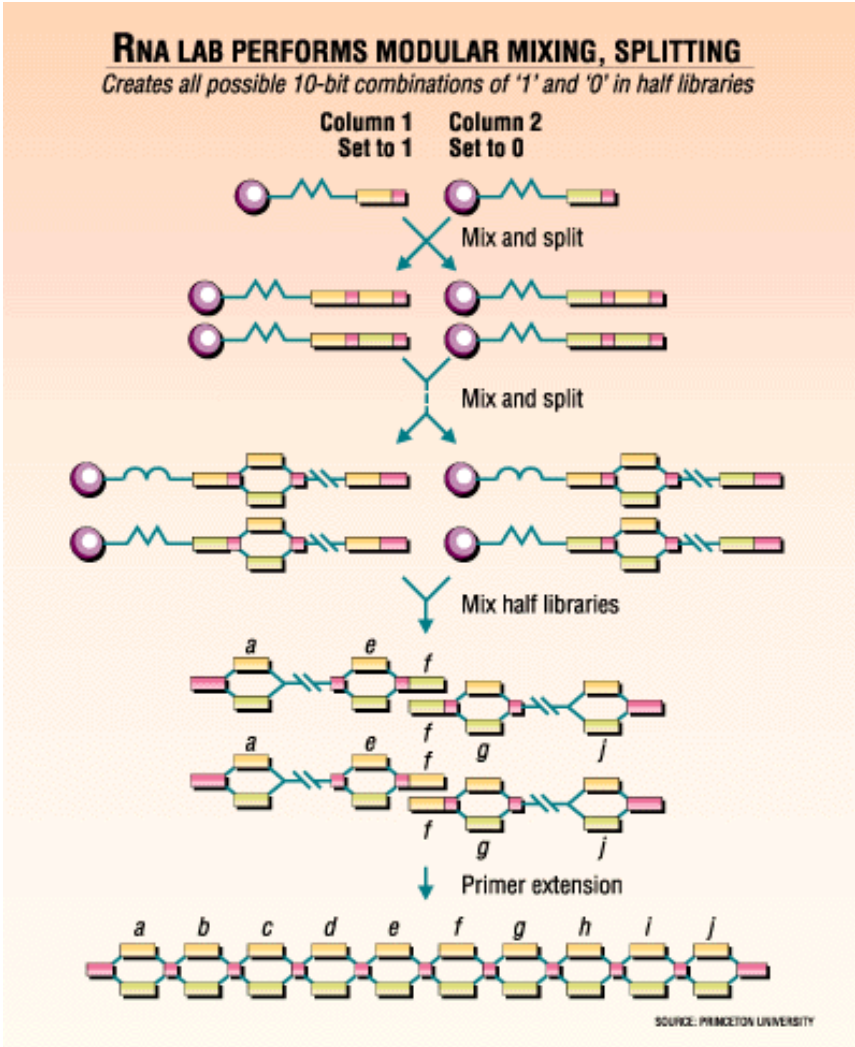
- Prepared as 2 halves using mix and split phosphoramidite chemistry with sequences shown on Table 1.

Table 1. Nucleotide sequences for each bit and spacer of the combinatorial library

Bit	Knight absent code, bit set to 0	Knight present code, bit set to 1	Spacer code
a	CTCTTACTCAATTCT	TCCTCACATTACTTA	TCTAC
b	CATATCAACATCTTA	ACTTCCTTTATATCC	ATAAC
c	ATCCTCCACTTCACA	TTATAACAAACATCC	CTTAA
d	TTAAAATCTTCCCTC	ACATAACCCTCTTCA	TTTAC
e	CTATTTATCCACACC	ACCTTACTTTCCATA	TACAA
f	GCTTCAAACAATTCC	GTACATTCTCCCTAC	TCCTT
g	AACTCTCAAATTCAA	CATAATCTTATATTC	TCAAT
h	CTAACCTTTACTTCA	ATAATCACATACTTC	TCCAA
i	CATTCCTTATCCCAC	TCCACCAACTACCTA	ACACA
j	CACCCTTTCTCCTCT	TTTTAAATTTCACAA	SUFFIX

The Knight Problem – RNA Library

- Prepared as 2 halves using mix and split phosphoramidite chemistry with sequences



The Knight Problem – RNA Library

- To build the library three criteria were used
  1. Each bit encoding must be fundamentally different. Hence sequences were chosen to maximize the Hamming Distance between different library strands. (no more than 5 matches over 20-nt windows, both within and between all 2^10 possible strands).

Table 1. Nucleotide sequences for each bit and spacer of the combinatorial library

Bit	Knight absent code, bit set to 0	Knight present code, bit set to 1	Spacer code
a	CTCTTACTCAATTCT	TCCTCACATTACTTA	TCTAC
b	CATATCAACATCTTA	ACTTCCTTTATATCC	ATAAC
c	ATCCTCCACTTCACA	TTATAACAAACATCC	CTTAA
d	TTAAAATCTTCCCTC	ACATAACCCTCTTCA	TTTAC
e	CTATTTATCCACACC	ACCTTACTTTCCATA	TACAA
f	GCTTCAAACAATTCC	GTACATTCTCCCTAC	TCCTT
g	AACTCTCAAATTCAA	CATAATCTTATATTC	TCAAT
h	CTAACCTTTACTTCA	ATAATCACATACTTC	TCCAA
i	CATTCCTTATCCCAC	TCCACCAACTACCTA	ACACA
j	CACCCTTTCTCCTCT	TTTTAAATTTCACAA	SUFFIX

The Knight Problem – RNA Library

- To build the library three criteria were used
  2. Strands biased to avoid secondary structure, each bit would be equally accessible to the enzymes and oligonucleotides. Accomplished using 3 letter alphabet, A,C, and U for bits and spacers. Eliminating potential G-C and G-U pairs.

Table 1. Nucleotide sequences for each bit and spacer of the combinatorial library

Bit	Knight absent code, bit set to 0	Knight present code, bit set to 1	Spacer code
a	CTCTTACTCAATTCT	TCCTCACATTACTTA	TCTAC
b	CATATCAACATCTTA	ACTTCCTTTATATCC	ATAAC
c	ATCCTCCACTTCACA	TTATAACAAACATCC	CTTAA
d	TTAAAATCTTCCCTC	ACATAACCCTCTTCA	TTTAC
e	CTATTTATCCACACC	ACCTTACTTTCCATA	TACAA
f	GCTTCAAACAATTCC	GTACATTCTCCCTAC	TCCTT
g	AACTCTCAAATTCAA	CATAATCTTATATTC	TCAAT
h	CTAACCTTTACTTCA	ATAATCACATACTTC	TCCAA
i	CATTCCTTATCCCAC	TCCACCAACTACCTA	ACACA
j	CACCCTTTCTCCTCT	TTTTAAATTTCACAA	SUFFIX

The Knight Problem – RNA Library

- To build the library three criteria were used
  3. The strands would avoid hybridization to themselves or any other library strands by more then seven consecutive base pairs. Otherwise this would interfere to operate on RNA strands by making regions inaccessible to reagents.

Table 1. Nucleotide sequences for each bit and spacer of the combinatorial library

Bit	Knight absent code, bit set to 0	Knight present code, bit set to 1	Spacer code
a	CTCTTACTCAATTCT	TCCTCACATTACTTA	TCTAC
b	CATATCAACATCTTA	ACTTCCTTTATATCC	ATAAC
c	ATCCTCCACTTCACA	TTATAACAAACATCC	CTTAA
d	TTAAAATCTTCCCTC	ACATAACCCTCTTCA	TTTAC
e	CTATTTATCCACACC	ACCTTACTTTCCATA	TACAA
f	GCTTCAAACAATTCC	GTACATTCTCCCTAC	TCCTT
g	AACTCTCAAATTCAA	CATAATCTTATATTC	TCAAT
h	CTAACCTTTACTTCA	ATAATCACATACTTC	TCCAA
i	CATTCCTTATCCCAC	TCCACCAACTACCTA	ACACA
j	CACCCTTTCTCCTCT	TTTTAAATTTCACAA	SUFFIX

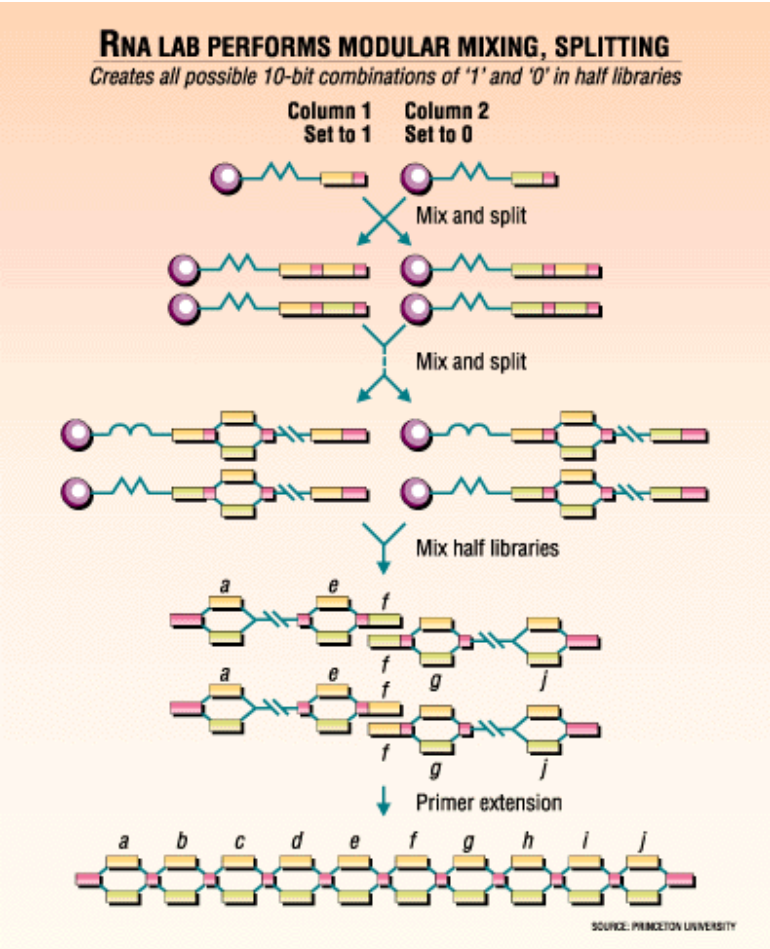
The Knight Problem – RNA Library

- To satisfy all combo, PERMUTE (a computer program) was used to generate random nucleotide combos until all three criteria were met.
- Prefix and Suffix provide PCR primer binding sites. Table 1.0 shows sequences.
- Library was physically built using a mix and split strategy.

Table 1. Nucleotide sequences for each bit and spacer of the combinatorial library

Bit	Knight absent code, bit set to 0	Knight present code, bit set to 1	Spacer code
a	CTCTTACTCAATTCT	TCCTCACATTACTTA	TCTAC
b	CATATCAACATCTTA	ACTTCCTTTATATCC	ATAAC
c	ATCCTCCACTTCACA	TTATAACAAACATCC	CTTAA
d	TTAAAATCTTCCCTC	ACATAACCCTCTTCA	TTTAC
e	CTATTTATCCACACC	ACCTTACTTTCCATA	TACAA
f	GCTTCAAACAATTCC	GTACATTCTCCCTAC	TCCTT
g	AACTCTCAAATTCAA	CATAATCTTATATTC	TCAAT
h	CTAACCTTTACTTCA	ATAATCACATACTTC	TCCAA
i	CATTCCTTATCCCAC	TCCACCAACTACCTA	ACACA
j	CACCCTTTCTCCTCT	TTTTAAATTTCACAA	SUFFIX

The Knight Problem – RNA Library



Mix and Split

- Bit n set to 0 and spacer n synthesized on one column.
- Bit n set to 1 and spacer n synthesized on the other column.
- These were mixed together
- Next variable position similarly created.
- In the end 2^10 (1024) library strands were created (all possible solutions)
- Prefix and suffix used to verify degeneracy of the DNA pool.
- RNase H digestions destroyed the RNA strand of RNA DNA hybrids marked by hybridization of the pool RNA to the complementary bit oligonucleotides shown in Table 2.
- PCR product cloned directly using PCR

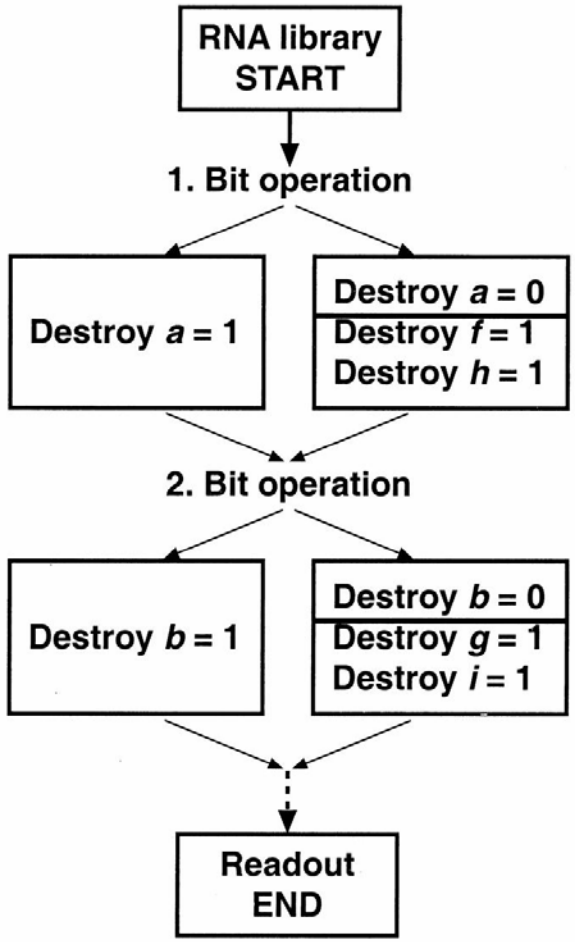
The Knight Problem - Algorithm

- Initially we have all solutions (1024)
- Each string has form x1,…,xn
- Where xi = 1 or 0
- x1 =a, x2=b, …, x9=i (this is our mapping)
- destroy strings that fail to satisfy the first clause
- $((\neg h \wedge \neg f) \vee \neg a)$ as follows:
  - i) Execute OR clause and divide the library into two halves. In one test tube, we select those strands that contain a 1 at position ‘a’ by annealing DNA bit oligonucleotide a0 to the library. Hence digesting those strands with position ‘a’ set to 0 (thereby setting bit at position ‘a’ to 1).

    Simultaneously, destroy any 1’s at those bit positions that must be set to 0 to full the clause (bits ‘f’ and ‘h’ in this case).

    In the other test tube we anneal DNA bit oligonucleotide a1 to perform a mirror operation setting bit position ‘a’ to 0.
  - ii) Undigested molecules are recovered and reverse transcribed. Contents of both tubes are mixed and amplified.

    Library is split again to execute the next or clause..

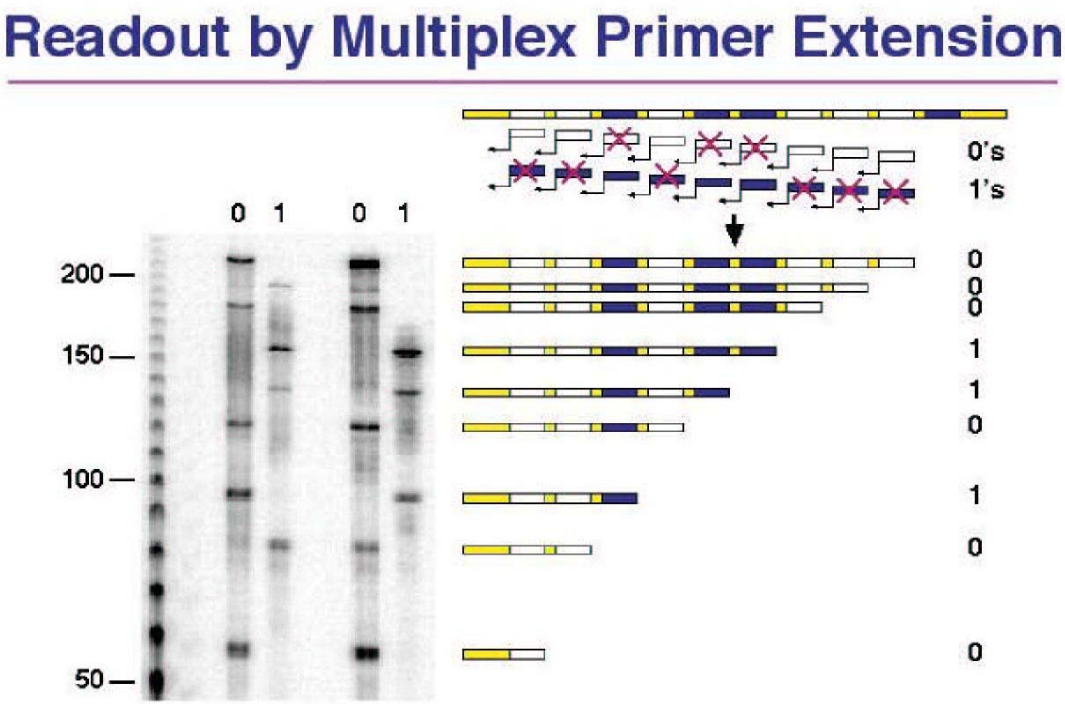


The Knight Problem - Algorithm

- Bit Shuffling
  - Recombination likely to occur during PCR amp of heterogeneous target sequences. Especially since several stretches are shared.
  - 25 cycles of PCR
  - 20 clones randomly chosen
  - 40% the result of bit shuffling
  - Bit shuffling was a serious problem
  - 15 cycles of PCR done
  - 20 random clones chosen
  - No bit shuffling found
  - Therefore High proportion of incompletely extended strands annealing to heterogeneous target sequences. Fixed by reducing PCR cycles.

The Knight Problem

- Readout Methods
  - Multiplex Linear PCR
  - Creates a bar code for each strand.

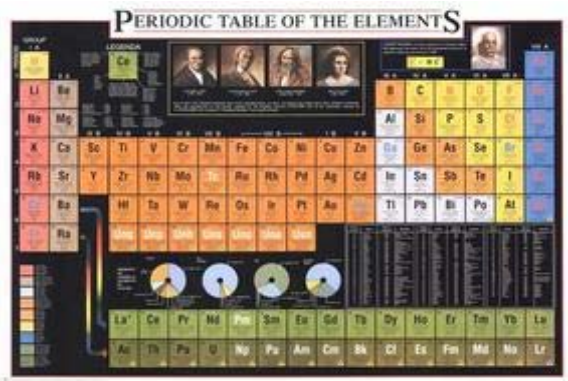


Knight Problem

- RNA Solutions
  - 43 Clones Randomly chosen
  - 42 Represented a Solution
  - 127 Knights on 43 Boards and only on knight had an unacceptable position.
  - 97.7% success rate effectively.

State of the Art

- Interdisciplinary field, includes molecular biology, chemistry, computer science, mathematics



State of the Art

- End to Disease?
  - Most research on DNA processors is being done by biotech companies hoping to cash in on recent breakthroughs on human genome
  - Microprocessor chips – contain fragments of DNA in place of electrical circuitry
  - Contain array of specific genetic info
  - These arrays called microarrays
  - Can compare chip to real human DNA to see how human DNA changes when it becomes cancerous or is afflicted with a virus

State of the Art

- Micro Factories
  - Motorized tweezers 1000x smaller then the head of a pin
  - Molecular sized motors could assemble complex structures such as electronic circuits.
  - Natural protein motors in living cells cause muscle contractions
  - Took double helix DNA structure, which usually floats with its arms open
  - Arms open and close by adding or subtracting another DNA strand
  - Pair of dye molecules to witness motion
  - Significant is that we can induce motion at the molecular level

State of the Art

- Breaking DES Using Molecular Computer (Data Encryption Standard)
  - Method of encrypting 64-bit messages with a 56-bit key
  - Used extensively in US
  - Using special purpose electronic computer and differential cryptanalysis DES can be found in several days.
  - However, would require 2^43 examples of encrypted and decrypted messages (plain text/cipher text pairs) and would slow down by a factor of 256 if key was increased to 64bits.
  - Could be solved using Adleman's original technique. Would take 4 months but need only a single plain-text/cipher-text pair or an example of cipher text with several plain text candidates to be successful.
  - + they could do it using less then a gram of DNA.

State of the Art

- Olympus Optical Co. – First practical DNA Computer
  - Tokyo (July 3rd, 2002)
  - Olympus Optical Co. Ltd.
  - First commercially practical DNA computer
  - Specializes in gene analysis
  - Akira Toyama, an assistant prof at Tokyo University
  - Standard gene analysis approach very time consuming (3 days)
  - Now done in 6hrs
  - Joint project called NovousGene Inc. spec in genome informatics
  - Two sections –
    - Molecular Calculation component
      - » DNA combination of molecules
      - » Implements chemical reactions
      - » Searches
      - » Pulls out right DNA results
    - Electronic Calculation component
      - » Executes processing programs
      - » Analysis these results
  - Available for commercial use by researchers by 2003 sometime

State of the Art

- Israel's First DNA computer
  - Trillion could fit in a test tube
  - Billions of ops/sec 99.8% accuracy
  - First programmable autonomous computing machine
  - Input, output, software, and hardware all made of biomolecules
  - DNA comp inside cells to monitor cell vitals.

Making DNA Computers Error Resistant

- DNA computers are Not ERROR Free!
- DNA calculations fall into 3 basic classes
  1. Decreasing Volume (# strands are reduced with each step)
  2. Constant Volume (# strands constant throughout all steps)
  3. Mixed Algorithms

Making DNA Computers Error Resistant

- Aldeman and Lipton are even more special. Each strand is “good” or “bad”
- Good strands encode a solution
- Bad strands do not
- If a good strand is damaged or lost the algorithm fails
- If a bad strand is not removed and many are left at end then the algorithm fails

Making DNA Computers Error Resistant

Two sources of errors

1) Every operation can cause an error (extraction)
   - extraction is not perfect usually 95% strands match the desired pattern
   - In addition, strands that do not match will sometimes be removed anyways. Rates typically 1 part in 10^6
2) DNA has ½ life, and decays at a finite rate. If an algorithm takes months good solutions will dissolve away.

Making DNA Computers Error Resistant

- First main result - Map Adleman's and Lipton's algorithms into a new algorithms that are constant volume. These new algorithms are highly resistant to errors.
- Also they will run in same number of steps approximately, the time penalty is small.

Making DNA Computers Error Resistant

Assume the following following:

- **2^n** strands of DNA at start
- only one is good
- Hence worst case
- The algorithm consists of **s** extraction steps
- Good strand always matches, bad ones may or may not
- A **Type I error** (or false negative) error occurs when a strand is not correctly extracted
- Let **p** be the probability that this happens (Type I error)

Making DNA Computers Error Resistant

- A Type II error (or false positive) error occurs when a strand should NOT be extracted but is anyways.
- Let **q** be the probability that this happens (Type II error)
- Typical values of **p = 0.95** and **q = 10^-6**

Making DNA Computers Error Resistant

- If **x** is a bad strand, let **M(x)** be the number of extractions for which it does NOT match the pattern. **M(x) is at least 1**
- Assume that computation decreases its volume at a uniform rate.
- Also, assume that every step reduces volume at uniform rate
- Thus we have **2^n** strands and s steps therefore volume goes down by a factor of 2 every **s/n** steps. If s=900 and n=60 then every 15 steps DNA volume halves.

Making DNA Computers Error Resistant

Modifications to the algorithms come here (twofold)

1) every s/n steps double the amount of DNA (PCR)
   - addition in cost to time is small
   - even if PCR process is slow, only occurs s/n steps (therefore small percentage of total time).

Making DNA Computers Error Resistant

2) modify “Final Detect Step”.

- Before we assume that ideally one strand remains
- Replace detect step
- Instead we remove a strand and sequence it
- Check if it’s a solution if so then we are done
- If not try again with another strand
- Try **m** times, if we fail then no solution is found

Making DNA Computers Error Resistant

Now the algorithms can fail in 2 ways

1) no good strands exist (there is no solution)
2) There might be good strands but there are so many bad strands that the probability that the final step gets a solution is small.

Making DNA Computers Error Resistant

- Leads to the following: Let **Ps** be the Survival Probability, the probability that a good strand is left in tube

- Let **Pr** be the Selection Probability, the probability that at least **δ** (small delta) strands are good

Making DNA Computers Error Resistant

- **Pr** and **Ps** together determine success of new algorithm
- We use them to calculate the upper bound of algorithm failure
- failure at most is:
    - **1 – Ps + Ps(1 – Pr) + PsPr(1 – δ)^m**
- remember **m** is repeated detects and **δ** number of good strands
- THUS, the probability that the algorithm works is:
    - **PsPr(1 – δ)^m**

Making DNA Computers Error Resistant

- If we can bind **Ps** and **Pr** we get more control over our success

Making DNA Computers Error Resistant

- First consider Ps:
  - without PCR the probability that a good strand survives is p^**s** (s extractions performed)
  - but every s/n steps survivors are doubled (both good and bad).
  - This is an example of **branching process**. Famous for modeling nuclear reactions and spread of diseases.
  - Let **r** be the probability that a single good strand survives to the next PCR step. i.e. **r = p^(s/n)**. If it survives then we have 2 good ones. Therefore greater chance for future extractions.
  - Following Feller (the guy behind the branching process). What we want to look at is the extinction probability **ζ** (small zeta). For this particular branding process it is the smallest positive solution to the equation.
  - **1 – r + rζ^2 = ζ**
  - simplify this down to : **ζ = 1/r -1.** The probability Ps, that some good strand survives is **1 – ζ.** Therefore:
  - **Ps = 1 – ζ = 2 – 1/r = 2 – p-s/n**

Making DNA Computers Error Resistant

– Table 1.0 shows some example results. Table shows that some good strands survive the entire series of steps.

p	probability good survives
.97	.42
.98	.65
.99	.84

Table 1: $s = 900$ and $n = 60$

Making DNA Computers Error Resistant

- Now must show that **Pr** is large. i.e. that it is likely that the ratio of good strands to bad is not too small.
- We can ignore PCR since we interested in ratios only. And the PCR does not effect ratios whatsoever

Making DNA Computers Error Resistant

- Considering **Pr**:
  - Recall, **bad strand x** does NOT match pattern of an extract for **M(x)** extractions. Therefore bad x will survive **M(x) Type II errors**. Thus at most bad strands survive:
  - $\sum_{x} q^{M(x)}$
  - Remember q = probability that Type II error occurs.
  - We simplify this sum to by defining **Mk** as the number of bad strands x that have **M(x)=k**.
  - The above is then equal to:
  - **M1q + M2q^2 + M3q^3 + …**
  - Missing terms insignificant since **q = 10^-6** (very small) and there are at most 2^70 strands
  - Key point here is that **M1q + M2q^2 + M3q^3** determines how long final detect step takes. Let this quantity be **l**. Then detect step would require **O(l)** steps.
  - We would want to minimize this. Without further assumptions **l** can be large.

Making DNA Computers Error Resistant

– Points in our favor here:

- Practical problems – few bad strands meet many solution constraints. Typically M(x) is very large for bad strands. Even if **M1=10^8** and **M2=10^14** and **M3=10^20 l** is only 300.
- We can restructure computation to reduce l. If **l** is large just repeat whole computation then all M(x) values are doubled.
- Transform problem to reduce the number of "almost solutions" (reduce **l**) related with Probabilistic Checkable Proofs.

Making DNA Computers Error Resistant

- The above works for algorithms where there is a reduction in volume (and we take advantage of it to bring it up). This wouldn't work for constant volume algorithms. The physical materials used would double after every PCR. Quickly we would find that there is too much physical material to handle.
- Here another technique needed!

Making DNA Computers Error Resistant

- Double Encoding of data
- Here the idea is to use only ½ the length of the available DNA strand. Here the binary bit is encoded twice. Increasing the chances of it being found when we detect and extract.

Conclusion

- All of the original papers (mostly in PDF format) can be found on my website at
- www.ucalgary.ca/~omair/cpsc60173/Presentation
- Also, a small summary will be provided of each article and link posted.