Bio-op Errors in DNA Computing
A Sensitivity Analysis

Daniel Bilar

University of New Orleans
Department of Computer Science
New Orleans, Louisiana, USA

May 27, 2009
ACIS SNPD '09
Catholic University of Daegu
Daegu, Republic of Korea

Talk Roadmap

Motivation DNA Computing

Parallelizable combinatorial problems such as Hamiltonian Path, DES code breaking and knapsack problems [1, 2, 3] can be solved
Error rates of biological operation range from 10^{-5} to 0.05 [4]

Sensitivity Analysis on DNA Algorithm

Simulate DNA algorithm for Shortest Common Superstring Problem
Perform sensitivity analysis for each step of algorithm
Goal is to make algorithm error resistant

Tuning the Errors

Good Encoding focus on *input data* error-resistance
Multiplexing focus on *operation* error-resistance
Constant Volume Transformation focus on *algorithm* as a whole error-resistance

Chosen Problem

Shortest Common Superstring Problem

NP-Complete Combinatorial Problem Given an alphabet Σ, a finite set R of strings from Σ^* (the set of all words over Σ) and a positive integer K, find a string $w \in \Sigma^*$ with length $|w| \leq K$ such that each string $x \in R$ is a substring of w.

Gloor's Algorithm [5]

1. **Encode** all the strings $x_1, x_2, \ldots, x_n \in R$ as DNA strands
2. **Generate all possible solutions** which are DNA strands w of length less than or equal to K
3. **Iteratively refine solution** Let x_j be a string of R. From our solution population, select only the ones which contain x_j as a sub-string. Let this be our new solution population. Repeat this step for each string $x_i \in R, 1 \leq i \leq n$
4. **Return result** if our solution population is non-empty, return 'Yes' and the solution string(s). Otherwise, return 'No'

Empirical Error Rates

Step	Bio-op	Type I Error	Type II Error
1) Encoding sub-strings	Synthesizing through sequential coupling	NA	Wrong letter is bonded (0.05)
2) Generate solution population	Synthesizing through sequential coupling	NA	Wrong letter is bonded (0.05)
3) Match sub-strings to solution population	Extraction using affinity purification	Correct match is not recognized as match (0.05)	Incorrect match is recognized as match (10^{-6})
4) Detect and output final solution	Sequencing using polymerase chain reaction and gel electrophoresis	Correct match is not recognized as match (0.05)	Incorrect match is recognized as match (10^{-5})

Table: Error rates of bio-operations [4][6]

Gloor's Algorithm [5]

1. **Encode** all the strings $x_1, x_2, \ldots, x_n \in R$ as DNA strands
2. **Generate all possible solutions** which are DNA strands w of length less than or equal to K
3. **Iteratively refine solution** Let x_j be a string of R. From our solution population, select only the ones which contain x_j as a sub-string. Let this be our new solution population. Repeat this step for each string $x_i \in R, 1 \leq i \leq n$
4. **Return result** if our solution population is non-empty, return 'Yes' and the solution string(s). Otherwise, return 'No'

Experiment and Results

Step	Type I Error Levels	Type II Error Levels
1) Encoding sub-strings	NA	$0.05, 0.005$
2) Generate solution population	NA	$0.05, 0.005, 0.0005, 0.00005$
3) Match sub-strings to solution population	$0.05, 0.005, 0.0005, 0.00005$	NA

Table: Bio-op error levels for factorial experiments

Setup

Algorithm implementation of all possible solutions of length $K \leq 6$ and chosen sub-string matches gg, t, cg, tg, tgg
Factorial experiment varied error rates for three bio-ops

Result

Hit rate most sensitive to the type II errors in step 1. In conjunction with lower type I error of step 3, pushed hit rate above the 90% mark
Lesson is encoding and extraction steps most important

Targeting Input Data: **Good Encoding** Overview

Target Input Data

False encoding of search strings most sensitive factor. Practical mechanism that produces error is *hybridization stringency* (number of complementary base pairs that have to match for DNA oligonucleotides to bond)

Deaton's Upper Bound [7]

Studied Hamiltonian Path Problem
Found upper bound of number of vertices that can be encoded in oligonucleotides of length n without producing mismatches

$$|C| \sum_{i=0}^{t} \binom{\frac{n}{2}}{i} (q-1) \leq q^{\frac{n}{2}}$$

where t is the number of errors that occur in hybridization, q is cardinality of the alphabet ($q = 4$ for DNA), and $|C|$ is the number of vertices.
Mismatch-free defined as every codeword being a distance greater than t from any other codeword
If the Hamming bound satisfied, no type II matching errors

Targeting Input Data: **Good Encoding** Discussion

Deaton's Upper Bound [7]

Upper bound of number of vertices that can be encoded in oligonucleotides of length n without producing mismatches

$$|C| \sum_{i=0}^{t} \binom{\frac{n}{2}}{i} (q-1) \leq q^{\frac{n}{2}}$$

where t is the number of errors that occur in hybridization, q is cardinality of the alphabet ($q = 4$ for DNA), and $|C|$ is the number of vertices.
Mismatch-free defined as every codeword being a distance greater than t from any other codeword
If the Hamming bound satisfied, no type II matching errors

Discussion

Biological pendant of the **Hamming error-correcting code**
Requires mismatch-free encoding, may not be possible for a given problem
Conclusion Added error flexibility has to be bought with carefully designed oligonucleotide encoding.

Targeting Operation: **Multiplexing** Overview

Target Operations

System rebound from error assuming a certain number of faulty inputs

von Neumann's Multiplexing [8]

Given input error rate and operation error rate ϵ, critical level of input must be determined for a desired output error rate ψ. Interpret group of inputs higher than critical level δ as a *positive* state, lower than critical level as *negative* state.

DNA computing adaption

For every bio-op with error rate ϵ, fix your output error rate ψ to a desirable level by replicating the inputs N times. Given N, find your critical level δ using

$$\rho(\mathrm{N}) = \frac{1}{\sqrt{2\pi \mathrm{k}}} \mathrm{e}^{-\frac{\mathrm{k}}{2}}, \text{with } \mathrm{k} = 0.62\sqrt{\mathrm{N}}$$

Interval zone $(\delta, 1-\delta)$ is one of uncertainty, where the error rate may or may not have been achieved. If at least the fraction $1-\delta$ of inputs remains the same, operation produces a positive result. If at most fraction δ of your inputs is same, operation produces negative result

Targeting Operation: **Multiplexing** Discussion

N	1000	2000	3000	5000	10000	20000
ρ(N)	$2.7 * 10^{-2}$	$2.6 * 10^{-3}$	$2.5 * 10^{-4}$	$4 * 10^{-6}$	$1.6 * 10^{-10}$	$2.8 * 10^{-19}$

Table: Given bio-op error rate $\epsilon = 0.005$, probability of uncertainty as a function of N

DNA computing adaption

For every bio-op with error rate ϵ, fix your output error rate ψ to a desirable level by replicating the inputs N times. Given N, find your critical level δ. **Interval zone** $(\delta, 1 - \delta)$ is one of uncertainty, where the error rate may or may not have been achieved. If at least fraction $1 - \delta$ of inputs remains the same, operation produces positive result. If at most fraction δ, operation produces negative result

Discussion

N **becomes very large** to decrease the probability of uncertainty
Multiplexing helps stabilize errors in algorithms with little data dependencies.
In some situations, multiplexing amplifies errors (divide-and-conquer algorithms)
Suggests **reformulation of algorithms** to suit m.o. of DNA computing

Targeting Algorithm: **Constant Volume** Overview

Target Algorithm

Previous two approaches concentrated on improving the operand and statistically improving error rate of operation
Broader view of **adapting algorithm** to the particularities of DNA computing

Boneh's Transform Approach [6]

Classify problems as Decreasing Volume' if number of strings decrease as the algorithm executes, 'Constant Volume' if number remains the same and 'Mixed' otherwise. DNA algorithms are 'Decreasing Volume', **transform into 'Constant Volume'**

Modification of bio-op steps 3 and 4 from Table 1

Step 3* Let s be the number of extraction steps, and let the initial solution population be 2^n strings. Double the solution population every $\frac{s}{n}$ steps using a PCR (a DNA amplification technique) operation.
Step 4* Pick m strands at random from the final solution population and check whether at least one of them is the desired solution. If not, report failure.

Targeting Algorithm: **Constant Volume** Discussion

Modification of bio-op steps 3 from Table 1

Step 3* Let s be the number of extraction steps, and let the initial solution population be 2^n strings. **Double solution population every $\frac{s}{n}$ steps** using a PCR (a DNA amplification technique) operation.

Keeping Constant Volume

Assume worst-case only one solution in 2^n population, let P_s be probability that solution survived extraction and is in final population.
Crucial step of bounding P_s Every $\frac{s}{n}$ steps, solution population is doubled. Hence, through growth process every s/n steps, chances increase that solution will survive all extractions

$$P_s = 2 - \alpha^{-\frac{s}{n}}, \text{with } \alpha \text{ being the type I error}$$

Discussion

Assumes PCR operation is error-free; accommodate by reducing α
Unmanageable for constant-volume algorithms, since quasi-exponential bio-mass growth

Concluding Thoughts

Figure: Yolshimhi hapsida! "Let's do our best"

Why bother with problem and DNA Computing?

Universally programmable DNA computers [9, 10]
Assumptions crucial Accept basic premise (e.g. DNA computing - operations inherently probabilistic)
Each **distinct computing environment may require particular algorithmic approach** (digital, DNA, hypercomputing, quantum [11])

Thank you

Thank you very much for your time and consideration of these ideas and for the opportunity to speak at SNPD 09 at the Catholic University of Daegu ☺

References I


L. Adleman, "Molecular Computation of Solutions to Combinatorial Problems," *Science*, no. 266, pp. 1021–1024, 1994.


L. Adleman, P. Rothemund, and et al, "On Applying Molecular Computation to the Data Encryption Standard," *Journal of Computational Biology*, vol. 6, no. 1, pp. 53–63, 1999.


E. B. Baum and D. Boneh, "Running Dynamic Programming Algorithms on a DNA Computer," in *DNA-Based Computers II: DIMACS*, vol. 44, pp. 77–87, 1999.


K. Langohr, "Sources of Error in DNA Computation," tech. rep., University of Western Ontario, 1997.


G. Gloor, L. Kari, and et al, "Towards a DNA Solution to the Shortest Common Superstring Problem," in *INTSYS '98: Proceedings of the IEEE International Joint Symposia on Intelligence and Systems*, p. 140, IEEE Computer Society, 1998.


D. Boneh and R. Lipton, "TR-491-95: Making DNA Computers Error Resistant," tech. rep., Princeton University (NJ), 1995.


R. Deaton and R. Murphy, "Good Encodings for DNA-based Solutions to Combinatorial Problems," in *DNA-Based Computers II: DIMACS*, vol. 44, pp. 247–258, 1999.


J. v. Neuman, "Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components," *Annals of Mathematics Studies*, no. 34, 1956.


X. Su and L. M. Smith, "Demonstration of a Universal Surface DNA Computer," *Nucleic Acids Research*, vol. 32, no. 10, pp. 3115–3123, 2004.

References II

Y. Benenson, B. Gil, and et al., "An autonomous Molecular Computer for Logical Control of Gene Expression," *Nature*, vol. 429, no. 6990, pp. 423–429, 2004.

M. J. Biercuk and H. Uys, "Optimized dynamical decoupling in a model quantum memory," *Nature*, vol. 458, pp. 996–1000, 2009.