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
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 $\Sigma$, a finite set $R$ of strings from $\Sigma^*$ (the set of all words over $\Sigma$) 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

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

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

Gloor’s Algorithm [5]

1. **Encode** all the strings $x_1, x_2, \ldots, x_n \in \mathbb{R}$ 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 $\mathbb{R}$. 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 $x_i \in \mathbb{R}, 1 \leq i \leq n$

4. **Return result** if our solution population is non-empty, return 'Yes' and the solution string(s). Otherwise, return 'No'
### Experiment and Results

<table>
<thead>
<tr>
<th>Step</th>
<th>Type I Error Levels</th>
<th>Type II Error Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Encoding sub-strings</td>
<td>NA</td>
<td>0.05, 0.005</td>
</tr>
<tr>
<td>2) Generate solution population</td>
<td>NA</td>
<td>0.05, 0.005, 0.0005, 0.00005</td>
</tr>
<tr>
<td>3) Match sub-strings to solution population</td>
<td>0.05, 0.005, 0.0005, 0.00005</td>
<td>NA</td>
</tr>
</tbody>
</table>

**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

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

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
Deaton’s Upper Bound [7]

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

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

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 $\epsilon$, critical level of input must be determined for a desired output error rate $\psi$. Interpret group of inputs higher than critical level $\delta$ as a positive state, lower than critical level as negative state.

DNA computing adaption

For every bio-op with error rate $\epsilon$, fix your output error rate $\psi$ to a desirable level by replicating the inputs $N$ times. Given $N$, find your critical level $\delta$ using

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

**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 $\delta$ of your inputs is same, operation produces negative result.
Targeting Operation: Multiplexing Discussion

<table>
<thead>
<tr>
<th>N</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\rho(N))</td>
<td>(2.7 \times 10^{-2})</td>
<td>(2.6 \times 10^{-3})</td>
<td>(2.5 \times 10^{-4})</td>
<td>(4 \times 10^{-6})</td>
<td>(1.6 \times 10^{-10})</td>
<td>(2.8 \times 10^{-19})</td>
</tr>
</tbody>
</table>

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 \(\epsilon\), fix your output error rate \(\psi\) to a desirable level by replicating the inputs \(N\) times. Given \(N\), find your critical level \(\delta\). **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 \(\delta\), 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.
Overview

Problem Setup And Analysis

Tuning the Errors

Conclusion

Sources

**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.
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 $\alpha$

Unmanageable for constant-volume algorithms, since quasi-exponential bio-mass growth
Concluding Thoughts

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


References II
