Introduction to Molecular Computing

Masami Hagiya
Professor, University of Tokyo
Graduate School of Information Science and Technology
Department of Computer Science
What is Molecular Computing?

• Biomolecule = information processing machine
  – Autonomous control of chemical reactions
    ⇒ Encoded in molecules themselves
    • Nanoscale, low energy
    • Massive parallelism
    • Physical and chemical functions of molecules

• Objectives of molecular computing
  – Scientific investigation of computational power of molecules and their reactions
  – Engineering realization of new computational paradigms based on molecular reactions

• References
Objectives of Molecular Computing

• Analysis of computational power of molecular reactions and Applications:
  – Molecular sensors using molecular computation
    ⇒ Application to biotechnology
  – Programmed self-assembly and molecular machines
    ⇒ Application to nanotechnology
  – Evolutionary computation by molecules
    ⇒ Application to molecular evolution

• New computational paradigms based on molecular reactions
## Related Fields

(Biology & Information Technology)

<table>
<thead>
<tr>
<th>Raw object</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Molecular Biology</td>
<td>Bioinformatics</td>
</tr>
<tr>
<td>Molecular Computing</td>
<td>Mathematical Biology</td>
</tr>
<tr>
<td>Molecular Evolutionary Engineering</td>
<td>Evolutionary Computation</td>
</tr>
<tr>
<td></td>
<td>Artificial Life</td>
</tr>
</tbody>
</table>

Evolution ⊆ Calculation
Related Fields
(Molecular Sciences)

• Molecular Electronics
  – Electronic circuit using molecular devices (existing computation paradigm)
  – Molecular computing as technology for constructing molecular circuits (nanotechnology)

• Nanotechnology
• Supramolecular chemistry
• Quantum computing
• Optical computing
• Molecular biology, biotechnology
• Molecular evolutionary engineering
Hagiya’s wet laboratory

- School of Engineering, Bldg. 9, Rm. 501
- Earthquake-proof reconstruction of Bldg. 9 completed at the end of March
- Experiments resumed in April
- General cleaning of the bldg on April 19
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity

• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membrane computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity
• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions
• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines
• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Analysis of Computational Power of Molecular Reactions

• Various computational models
  – Inter-molecular reactions vs. intra-molecular reactions
  – Liquid-phase vs. solid-phase
  – Test tube, membrane, cell
  – Heteronomous vs. autonomous

• Analysis of computational power
  (of various computational models)
  – Computability
  – Complexity --- time and space
  – Errors and yields --- probabilistic analysis
  – Analysis more faithful to actual molecular reactions
Analysis of Various Computational Models

• Adleman-Lipton
  – Random generation of solution candidates by hybridization of DNA
  – Extraction of solutions by data-parallel computation
  – Suyama --- Dynamic Programming
  – Sakamoto-Hagiya --- SAT Engine
  – Head-Yamamura --- Aqueous Computing

• Seeman-Winfree
  – Self-assembly of various forms of DNA molecules
  – Computation with self-assembly
Analysis of Various Computational Models

- Head
  - Language generation by gene splicing
- Ogihara-Ray
  - Parallel computation of Boolean circuits
- Hagiya-Sakamoto
  - State machine (Whiplash)
- Shapiro
  - Finite Automaton

\[ \rightarrow \text{molecular machines} \]
Adleman-Lipton Paradigm

- Adleman (Science 1994)
  - Solving Hamilton path problem with DNA
- Lipton, et al.
  - Solving SAT problem with DNA
- Massive parallelism using molecules
  - Combinatorial optimization as a main purpose
  - Random generation by DNA self-assembly
    - solution candidate $=\text{DNA molecule}$
  - Extraction of solution using biological experiment
- Currently recognized as a benchmark for biotechnology
  - Vision of the study: application to genomic analysis
Experiment by Adleman
(Science 1994)
Extraction and Detection of Solutions

- vertex 0
- vertex 6
- PCR
- length 20
- gel electrophoresis
- complementary sequence of vertex 1
- magnet
- execute each vertex
- extract solution molecules
- extract sequences including vertex 5
- length 140
- complementary sequence of vertex 5
- execute each vertex
- extract sequences including vertex 2
Adleman-Lipton Paradigm

generation of all solution candidates

0 0 0 0
1 1 1 1

inspection and extraction of solutions

$T_i$ : multiset of strings (test tube)

[Separate]
$T_2 = + (T_1, s)$ : extract sequences including $s$
$T_2 = - (T_1, s)$ : extract sequences without $s$

[Merge]
$T_3 = T_1 \cup T_2$ : merge $T_1$ and $T_2$

[Amplify]
$(T_2, T_3) = T_1$ : copy $T_1$

generation of all assignments
(Lipton 1995, satisfiability problem)

detection of solutions
Suyama’s DNA Computer

• “Counting” (Ogihara and Ray)
  – $O(2^{0.4n})$ molecules for $n$-variable 3-SAT
• “Dynamic programming” (Suyama)
• Iteration of generation and selection
  – Partial generation of solution candidates
  – Selection of solution candidates
• Although both are exponential orders, $O(2^{0.4n})$ is far less than $O(2^n)$
• Solid-phase
  – Affinity separation with magnetic beads
  – Suitable for automation ⇒ Robot !
Implementation of Basic Operations

- get \((T, +s),\) get \((T, -s)\)
- append \((T, s, e)\)
- amplify \((T, T_1, T_2, \ldots T_n)\)
DP Algorithm for 3CNF-SAT on DNA Computers

**Function dna3sat** \( (u_1, v_1, w_1, \ldots, u_n, v_n, w_n) \)

**Begin**

\[ T_2 = \{ X_1^T X_2^T, X_1^F X_2^T, X_1^T X_2^F, X_1^F X_2^F \} \]

for \( k = 3 \) to \( n \) do

amplify \( (T_{k-1}^T, T_w^T, T_w^F) \);

for \( j = 1 \) to \( m \) do

if \( w_j = x_k \) then

\[ T_w^F = \text{getuvsat}(T_w^F, u_j, v_j) \]

end

if \( w_j = \neg x_k \) then

\[ T_w^T = \text{getuvsat}(T_w^T, u_j, v_j) \]

end

end

\[ T^T = \text{append}(T_w^T, X_1^T, \overline{X_{k-1}^T} X_k^T); \quad T^F = \text{append}(T_w^F, X_k^F, \overline{X_{k-1}^T} X_k^F); \]

\[ T_k = \text{merge}(T^T, T^F); \]

end

**Return** detect \( (T_n) \);

**End**

---

**Function getuvsat** \( (T, u, v) \)

**Begin**

\[ T_u^T = \text{get}(T, + X_u^T); \quad T_u^F = \text{get}(T, - X_u^T); \]

\[ T_u^F = \text{get}(T_u^F, + X_u^F); \quad \text{/* can be omitted */} \]

\[ T_v^T = \text{get}(T_u^F, + X_v^T); \]

\[ T_v^T = \text{merge}(T_v^T, T_v^T); \]

**Return** \( T_v^T \);

**End**

---

**Number of operations**

\[(n - 2) \times (\text{amplify} + 2 \times \text{append} + \text{merge}) + m \times (3 \times \text{get} + \text{merge})\]
3CNF-SAT Solution on DP DNA Computer

**Problem:** 4 variables, 10 clauses

\[
\begin{align*}
(x_1 \lor x_2 \lor x_3) & \land (x_1 \lor \neg x_2 \lor x_3) \land \\
(\neg x_1 \lor x_2 \lor \neg x_3) & \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land \\
(x_1 \lor \neg x_3 \lor \neg x_4) & \land (\neg x_1 \lor x_2 \lor \neg x_4) \land \\
(\neg x_1 \lor x_3 \lor \neg x_4) & \land (x_2 \lor x_3 \lor x_4) \land \\
(x_2 \lor \neg x_3 \lor x_4) & \land (\neg x_2 \lor \neg x_3 \lor x_4)
\end{align*}
\]

**Solution:**

YES

\[\{X_1^T, X_2^T, X_3^F, X_4^F\}\]
DP Algorithm for 3CNF-SAT on DNA Computer

\( k \)'s loop: \( k \) ranges over variable indices

\( j \)'s loop: \( j \) ranges over clause indices

if \( x_k \) is the 3rd literal of the \( j \)-th clause then

remove those assignments which satisfy neither the 1st nor the 2nd literal

append \( X_k^F \) to the remaining assignments

(\( k = 3 \))

\[ x_3 \]

\[ X_1^T X_2^T \] \[ \rightarrow \]

\[ X_1^T X_2^T X_3^F \]

\[ X_1^F X_2^T \]

\[ (x_1 \lor \neg x_2 \lor x_3) \]

\[ X_1^T X_2^F X_3^F \]

\[ X_1^T X_2^F \]

\[ (x_1 \lor x_2 \lor x_3) \]

\[ X_1^F X_2^F \]
DP Algorithm for 3CNF-SAT on DNA Computer

$k$’s loop: $k$ ranges over variable indices

$j$’s loop: $j$ ranges over clause indices

if $x_k$ is the 3rd literal of the $j$-th clause then
remove those assignments which satisfy neither the 1st nor the 2nd literal
append $X_k^F$ to the remaining assignments

(Do similarly if $\neg x_k$ is the 3rd literal)

$k = 3$

$\neg x_3$  $X_1^T X_2^T$  $(\neg x_1 \lor \neg x_2 \lor \neg x_3)$

$X_1^F X_2^T$  $X_1^F X_2^T$  $X_1^F X_2^T X_3^T$

$X_1^T X_2^F$  $(x_1 \lor x_2 \lor x_3)$

$X_1^F X_2^F$  $X_1^F X_2^F X_3^T$
DP Algorithm for 3CNF-SAT on DNA Computer

$k$’s loop: $k$ ranges over variable indices

$j$’s loop: $j$ ranges over clause indices

if $x_k$ is the 3rd literal of the $j$-th clause then

remove those assignments which satisfy

neither the 1st nor the 2nd literal

append $X_k^F$ to the remaining assignments

(do similarly if $\neg x_k$ is the 3rd literal)

$k = 4$

$x_4$

$X_1^F X_2^T X_3^T \quad (\neg x_2 \lor \neg x_3 \lor x_4)$

$X_1^F X_2^F X_3^T \quad (x_2 \lor \neg x_3 \lor x_4)$

$X_1^T X_2^T X_3^F \quad X_1^T X_2^T X_3^F X_4^F$

$X_1^T X_2^F X_3^F \quad (x_2 \lor x_3 \lor x_4)$
10-variable and 43-clause instance of 3SAT

$$(\neg x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor \neg x_3 \lor \neg x_4)$$

$$\land (x_1 \lor x_4 \lor x_5) \land (x_2 \lor x_3 \lor \neg x_5) \land (\neg x_2 \lor \neg x_3 \lor \neg x_5)$$

$$\land (\neg x_1 \lor \neg x_3 \lor \neg x_5) \land (\neg x_2 \lor \neg x_4 \lor x_6) \land (\neg x_2 \lor x_3 \lor x_6)$$

$$\land (x_2 \lor \neg x_3 \lor x_6) \land (\neg x_1 \lor \neg x_5 \lor \neg x_6) \land (x_2 \lor \neg x_6 \lor x_7)$$

$$\land (x_1 \lor x_5 \lor x_7) \land (\neg x_1 \lor \neg x_5 \lor \neg x_7) \land (x_5 \lor \neg x_6 \lor \neg x_7)$$

$$\land (x_1 \lor \neg x_2 \lor \neg x_7) \land (x_1 \lor x_6 \lor \neg x_7) \land (\neg x_4 \lor x_6 \lor \neg x_7)$$

$$\land (x_1 \lor x_4 \lor x_8) \land (\neg x_1 \lor x_5 \lor x_8) \land (x_2 \lor \neg x_3 \lor x_8)$$

$$\land (x_1 \lor x_6 \lor x_8) \land (x_2 \lor x_5 \lor \neg x_8) \land (x_1 \lor x_4 \lor \neg x_8)$$

$$\land (\neg x_3 \lor \neg x_5 \lor \neg x_8) \land (\neg x_2 \lor x_4 \lor x_9) \land (x_4 \lor x_7 \lor x_9)$$

$$\land (x_1 \lor x_7 \lor x_9) \land (\neg x_4 \lor x_6 \lor \neg x_9) \land (\neg x_1 \lor x_3 \lor \neg x_9)$$

$$\land (\neg x_2 \lor x_3 \lor \neg x_9) \land (x_1 \lor \neg x_7 \lor \neg x_9) \land (\neg x_2 \lor x_4 \lor \neg x_9)$$

$$\land (\neg x_1 \lor x_5 \lor \neg x_9) \land (\neg x_4 \lor x_8 \lor x_{10}) \land (x_3 \lor \neg x_6 \lor x_{10})$$

$$\land (\neg x_2 \lor \neg x_7 \lor x_{10}) \land (x_3 \lor \neg x_4 \lor \neg x_{10}) \land (\neg x_5 \lor \neg x_6 \lor \neg x_{10})$$

$$\land (x_4 \lor x_5 \lor \neg x_{10}) \land (\neg x_1 \lor \neg x_3 \lor \neg x_{10}) \land (x_2 \lor x_8 \lor \neg x_{10})$$

$$\land (\neg x_1 \lor x_8 \lor \neg x_{10})$$
DNA Computer Robot based on MAGTRATION™ (Prototype No.1)
Programming DNA Computer

function dna 3sat \((u_1, v_1, w_1, \ldots, u_m, v_m, w_m)\) begin
\[T_2 = \{X_1^T X_2^T, X_1^T X_2^T, X_1^T X_2^T, X_1^T X_2^T, X_1^T X_2^T\}\];
for \(k = 3\) to \(n\) do
amplify\((T_{k-1}^T, T_w^T, T_w^E)\);
for \(j = 1\) to \(m\) do
if \(w_j = x_k\) then
\[T_w^E = \text{getuvsat}(T_w^E, u_j, v_j)\];
end
if \(w_j = \neg x_k\) then
\[T_w^T = \text{getuvsat}(T_w^T, u_j, v_j)\];
end
end
\[T^T = \text{append}(T_w^T, X_1^T X_2^T, X_1^T X_2^T, X_1^T X_2^T)\];
\[T^F = \text{append}(T_w^F, X_1^F, X_1^T X_2^T, X_1^F)\];
\[T_k = \text{merge}(T^T, T^F)\];
end
return detect\((T_n)\);
end

[Instrument]
[Reset Counter] 0
[Home Position] 0
[MJ-Open Lid]
... [Get1(0)]
[Get2(1)]
[Append(2)]
...
[Exit]

Do 2
SEND "LID OPEN"
Do 10
SEND "LID?"
Wait_msec 500
_CMP_GSTR "OPEN"
IF_Goto EQ 0 :open
Wait_msec 1000
Loop
Loop
: Time out
End
:open

protocol-level

script-level

Pascal/C-level
Hairpin Engine (SAT Engine)

- Selection by DNA hairpin structures
  - Digestion by restriction enzyme
  - Exclusive PCR
- 3-SAT
  - Single-stranded DNA comprised of literals each selected from a clause
  - Complementary literal = complementary sequence
  - Detection of inconsistency ⇒ hairpin
  - 6-variable 10-clause 3-SAT problem
- The essential part of SAT computation = hairpin formation
  - Number of steps is independent of the number of clauses/variables
  - Autonomous molecular computation
\((a \lor b \lor c) \land (\neg d \lor e \lor \neg f) \land \ldots \land (\neg c \lor \neg b \lor a) \land \ldots\)

digestion by restriction enzyme
exclusive PCR
Selection by Hairpin Structures

• Restriction enzyme digestion
  – Hairpins are cut at the restriction site inserted in each literal sequence

• Exclusive PCR
  – PCR is inefficient for hairpins
  – In exclusive PCR, solution is diluted in each cycle to keep the difference in amplification

• Number of steps is independent of the number of variables/clauses
Current Consensus on Adleman-Lipton Paradigm

• Far from outperforming electronic computers
  - Scale-up problem
• Important as a first proof that;
  - Molecules can really compute
• At least serves as a benchmark for biotechnology
• Application to genomic analysis (Suyama)
Seeman-Winfree’s Computation by DNA Self-Assembly

Various DNA Molecular Structures

Linear

Hairpin

3-Junction

DX (Double Crossover)

Various DNA Molecular Structures
The DNA representation of Wang tiles.
Winfree’s Tiling

An algorithmic pattern for self-assembly.

Sierpinski’s Triangle
Head’s Computation by Gene Splicing

• Theoretical model of gene splicing with restriction enzyme and ligase (Splicing Model)

Splicing Operation

x = GGGCTTAA

y = AAGATT

z = GGGTTTAA

w = AAGTTC

EcoRI
Language Generation by Splicing

• Splicing rule: \( r = u_1\#u_2#u_3\#u_4 \)
• \((x_1u_1u_2x_2, y_1u_3u_4y_2) \rightarrow_r (x_1u_1u_4y_2, y_1u_3u_2x_2)\)
• \(R\): Set of splicing rules
• \(A\): Set of strings (axiom)
• \(L\): Language generated from \(R\) & \(A\)
  – If \(x \in A\) Then \(x \in L\)
  – If \(x, y \in L\) and \(r \in R\) and \((x, y) \rightarrow_r (z, w)\)
    Then \(z, w \in L\)
• If \(R\) and \(A\) are finite, then \(L\) is regular
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity

• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Computability in Molecular Computing

• Computability of DNA self-assembly
  – Winfree’s results

• Computability of gene splicing
  – Various extensions of splicing model
Winfree’s Results on Computability of DNA Self-Assembly

- Language generated by linear molecules
  = regular
- Language generated by linear + hairpin + 3-junction molecules
  = context-free
- Language generated by linear + DX molecules
  = recursively enumerable
  = Turing computable
Winfree’s Model:
Example of Computation Process

\[ c = f(a, b) \]
\[ d = g(a, b) \]

Initial Configuration

Imitation of 1-dimensional Cellular Automaton
Extentions of Splicing Model

Generative ability of splicing model $< \text{Regular language}$

$(\text{splicing}) + \alpha ? = \text{Universal computational power}$

$+ \alpha : \text{circular molecules, multiple test tubes, time-dependent rules etc.}$
Circular Splicing System

$+\alpha:$

Allows to use circular strings (circular DNA)
Allows to distinguish between terminal and non-terminal symbols
(e.g. splicing of colon bacillus chromosome and F plasmid)

Splicing Rule: $u1<u2\#u3\u003cu4$
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity
• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions
• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines
• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Complexity of Molecular Computation

• Time
  – Number of experiment steps
  – Time required for each operation
    • Essential for the analysis of computational power of molecules

• Space (= degree of parallelism)
  – Number of molecules
    • Maximum
    • Total
  – Size of molecules (length)

• Trade-off analysis -- important
Complexity Analysis (Adleman-Lipton)

- **Reif (SPAA ’95)**
  - A nondeterministic Turing machine computation with input size $n$, space $s$ and time $2^{O(s)}$ can be executed in our PAM Model using $O(s)$ PA-Match steps and $O(s \log s)$ other PAM steps, employing aggregates of length $O(s)$.

- **Beaver (DNA1, 1995)**
  - Polynomial-step molecular computers compute PSPACE.

  - Exactly the problems in $P^{NP} = \Delta^p_2$ can be solved in polynomial time using Lipton’s model.
Rooß and Wagner (I&C, 1996)

- Exactly the problems in $P^{NP} = \Delta^P_2$ can be solved in polynomial time using Lipton’s model.
- $\text{BIO}\{\text{UN, BX, IN}\}, \{\text{EM}\})-P = P^{NP} = \Delta^P_2$
  - UN: union (merge) \( T_3 = T_1 \cup T_2 \)
  - BX: bit extraction (separate) \( T_2 = + (T_1, s) \) \( T_2 = - (T_1, s) \)
  - IN: initialization (random generation)
  - EM: emptiness test (detect)
  - -P: polynomial time
  - $P^{NP}$: polynomial time with NP-oracle
Complexity Analysis

• Rothemund and Winfree (STOC 2000)
  – For any $f(N)$ non-decreasing unbounded computable functions, the number of tiles required for the self-assembly of an $N \times N$ square is bounded infinitely often by $f(N)$.

• Winfree, Eng and Rozenberg (DNA6, 2000)
  – Linear assembly of string tiles can generate the output languages of finite-visit Turing Machines.
Errors and Yields of Reactions

• Yields
  – Equilibrium --- equilibrium constant \((K)\)
  – Time to reach equilibrium --- reaction rate \((k)\)
  – Example: \(\text{A} \leftrightarrow \text{B}\)
    \[ [\text{B}] = \frac{K}{1+K} \left( 1 - e^{-(k+k_-)t} \right) \]
    \[ K = \frac{k}{k_-} \]

• Errors
  – Example: mis-hybridization
  – Error probability is never 0

• Probabilistic analysis
Probabilistic Analysis

• Karp, Keynon and Waarts (SODA’96)
  – The number of extract operations required for achieving error-resilient bit evaluation is $\Theta(\lceil \log_\varepsilon \delta \rceil \times \lceil \log _\gamma \delta \rceil)$.

• Kurtz (DNA2, 1996)
  – Thermodynamical analysis of path formation in Adleman’s experiment
  – Time needed to form a Hamiltonian path --- $\Omega(n^2)$

• Winfree (1998, Ph.D. Thesis)
  – Thermodynamical analysis of DNA Tiling

• Rose, et al. (GECCO’99, etc.)
  – Computational incoherency
    (Thermodynamical analysis of mis-hybridization)
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity

• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Computational Aspects of Molecular Systems Design

• Molecular programming
• Design of molecules
  – Design of DNA = sequence design
  – Structure ⇒ sequence (inverse folding)
  – Patterns for self-assembly
  – Design of molecular machines
• Design of reactions
  – Adjustment of reaction conditions
  – Scheduling of experimental operations
  – Simulation tool
• Molecular machines
  – One of the current objectives of molecular programming
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity

• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Sequence Design

• Evaluation of sequence set
  – Avoiding mis-hybridization
    • Hamming distance
    • Energy calculation ⇒ mfold (Zuker), Vienna Package
  – Uniform $T_m$ (melting temperature)

• Searching for sequence set
  – Genetic algorithm
  – Coding theory --- Arita’s template method

• Inverse problem
  – Structure ⇒ sequence (inverse folding)
  – Vienna group
Template Method

• Arita and Kobayashi, 2002
  Same positioning of [AT] or [GC] in every sequence
  (= “template”)
  e.g. from 011010

  ACCTGA, TGCTCA, TCGACA, etc.

→ Melting temperature of every sequence
  will be the same
Stacking Energy

- ΔG kcal/mol (DNA/DNA) by Sugimoto et al.
Template Including Mismatches

• Proper selection of template ensures mismatch(es) even with shift / reverse

e.g. when 110100

110100 110100 110100 110100
110100 110100 110100

Includes at least 2 mismatches
even with any shifting or concatenation
Template Selection

• Selecting template $T$ which will include minimum of $(d)$ mismatches in each of the following patterns

- $T^R$
- $TT^R, T^R T$
- $TT, T^R T^R$

$T^R$: reversed sequence of $T$
When $T=110100$, $T^R=001011$
Examples of Templates

• Length 6 (2 mismatches)
  110100 (of $2^6$)

• Length 11 (4 mismatches)
  01110100100, 01011100010, 11000100101
  (of $2^{11}$)

• Length 23 (9 mismatches)
  01111010110011001010000, 10110011001010000011110,
  11100000101001100110101
  (of $2^{23}$)
Design of DNA Sequence

“Template + Error Correcting Code”

1 1 0 1 0 0 (template)
+ 0 1 0 0 1 1 (any code)

A T C A G G (DNA sequence)

Any Error Correcting Code can be used

1. BCH Code
2. Golay Code
3. Hamming Code etc.
Inverse Folding

- Vienna group
- Using McCaskill’s algorithm
- Sequence search by minimization of cost function

\[ \Xi(x) = E(x, \Omega) - G(x) = -RT \ln p \]
- \( \Omega \): Target structure
- \( x \): Sequence
- \( E(x, \Omega) \): Free energy of \( \Omega \) at \( x \)
- \( G(x) \): Ensemble free energy of sequence \( x \)
  (McCaskill)
- \( p \): Probability of \( \Omega \) at \( x \)
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity

• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Design of Molecular Reactions

• Condition of reactions
  – Temperature
  – Salt concentration
  – Time
• Operation scheduling
• Simulation
  – e-PCR
    • http://www.ncbi.nlm.nih.gov/genome/sts/epcr.cgi
  – VNA
VNA: Simulator for Virtual DNA

• Abstract, but sufficiently physical
  Bridging the gap between abstract models and actual reactions
  molecule — hybrid of virtual strands
  
  \[ \begin{align*}
  abcd & \\
  | & \\
  CDEF & 
  \end{align*} \]

• Reactions
  – hybridization
  – denaturation
  – restriction
  – ligation
  – self-hybridization
  – extension
VNA (cont)

• Objectives
  – Verifying feasibility of algorithms for DNA computation
  – Verifying validity of molecular biology experiments
    (e.g., PCR experiments)
  – Parameter fitting in molecular biology experiments

• Examples
  – Ogihara and Ray’s computation of Boolean circuits
  – Winfree’s construction of double-crossover units
  – PCR experiments

• Implementation
  – Java ⇒ executable as an applet
VNA (cont)

• Methods
  – Combinatorial enumeration
  – Continuous simulation (diff. eq.)

• Avoiding combinational explosion

• Contributions in simulation technology
  – Threshold
  – Stochastic

• Parameter fitting by GA
  – Optimizing amplification in PCR experiments
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity
• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions
• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines
• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Information Processing of Gene Expression Analysis Using DNA Computation

- Direct input of DNA molecule
- Massive parallelism

DNA encoding

Information processing inside the test tube *in vitro*

Output on DNA chips

Gene Expression

Direct input:
Intelligent DNA Chip

“(Dcn1 \land \neg Dcn2) \lor Dcn3”

“Dcn1 \lor Dcn3”

“\neg Dcn1 \lor (Dcn2 \land \neg Dcn3)”
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models · Computability · Complexity

• Computational aspects of molecular systems design
  – Design of molecules · Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membranous computing · Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
DNA Nanotechnology
DNA Self-Assembly

• DNA lattice
• DNA as a connector of molecules
  – Self-assembly of nanoparticles using DNA
  – Self-assembly of nanowires using DNA
• DNA tile
  – Structure formation by DNA itself
• Programmed self-assembly
DNA-Based Self-Assembly of Nanoparticles

Early Studies

• C. A. Mirkin et al.

• A. P. Alivisatos et al.
  Organization of ‘nanocrystal molecules’ using DNA. Nature 382, 609-611 (1996)
Winfree-Seeman’s DNA Tiles (double crossover molecules)

The DNA representation of Wang tiles.
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity
• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions
• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines
• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
Molecular Machines

• Machines as Actuators
  – Motor
  – Transporter

• Abstract Machines --- Finite State Machines (Automaton)
  – Have a finite number of states
  – Change their state autonomously or according to inputs
  – May produce outputs
  – Are the first step towards general-purpose computers
  – Have many kinds of applications
    • Switch
      • Memory (both holding contents and addressing)

• The difference between the two is still unclear
Molecular System Consisting of Finite State Machines

Molecular System

Information Processing

Output:
- movement
- transformation
- structural formation
- light
- electricity
- heat

Computation

Input:
- molecule
- temperature
- light
- salt concentration
- voltage
Molecular (DNA) State Machines

• Terminal-sequence machines
  – The terminal sequence encodes the state
  – Our whiplash machine
    • Gets longer as it changes the state
  – Shapiro’s automaton
    • Gets shorter as it changes the state

• Conformational machines
  – The state is encoded as a structure
  – Yurk’s molecular tweezers
  – Seeman’s PX-JX₂ Switch
  – Our hairpin-based machine…
Whiplash PCR (WPCR)

1) $\overline{B}$ $\overline{A}$ $B$ $A$ $C$

2) $\overline{B}$ $\overline{A}$ $B$ $A$ $C$

Komiya et al.
Whiplash PCR (WPCR)

3)

4)
Polymerization by DNA polymerase with dATP, dCTP, dGTP
Back-hybridization

Competing Alternative Hairpin Forms
Temperature optimization for WPCR

- 8 M urea 8% PAGE

Komiya, et al.

# Thermal schedule

- 94°C for 1 min.
- ↓
- x °C for 5 min.
- x = 59.8 ~ 92.2

in 1X Pfx buffer

- (the composition unknown)
- 1 mM MgSO₄
- 0.2 mM dATP, dCTP, dGTP
- 1.5 units Platinum Pfx DNA polymerase

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>59.8</th>
<th>65.9</th>
<th>74.0</th>
<th>82.1</th>
<th>89.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incubated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>62.2</td>
<td>69.9</td>
<td>78.0</td>
<td>86.1</td>
<td>92.2</td>
</tr>
</tbody>
</table>
Successful implementation of transitions

- 12% PAGE

Komiya, et al.
Shapiro’s DNA Automata

IIS-type restriction
Restriction cite Spacer

\[ \langle S', a' \rangle \]

\[ \langle S, a \rangle \rightarrow \langle S', a' \rangle \]

Transition molecule

The input sequence encoding the symbol \( a' \) contains \( \langle S', a' \rangle \) for each \( S' \). The transition molecule cuts the input at the site regulated by the spacer.
Shapiro’s DNA Automata

- *Nature* 2001
- 2 input symbols, 2 states
- *FokI*

\[ a = \text{CTGGGCT} \quad \quad b = \text{CGCAGC} \]

\[ 5' - p...22...\text{GGATGTAC} \quad \quad 3' - \text{GGT...22...CCTACATGCGAP} \]
\[ S_0, a \rightarrow S_0 \]

\[ 5' - p...22...\text{GGATGACGAC} \quad \quad 3' - \text{GGT...22...CCTACTGCTGCGAP} \]
\[ S_0, a \rightarrow S_1 \]
Yurke’s Molecular Tweezers
Introduction to Molecular Computing

Table of Contents:

• Analysis of computational power of molecular reactions
  – Computational models • Computability • Complexity

• Computational aspects of molecular systems design
  – Design of molecules • Design of molecular reactions

• Application of computational power of molecular reactions
  – Intelligent molecular sensing
  – Self-assembly
  – Molecular machines

• New computational paradigms based on molecular reactions
  – Membranous computing • Amorphous computing
  – Association with optical and quantum
  – Association with molecular electronics
New Computational Paradigms

• Membrane Computing
  – Paun

• Amorphous Computing
  – MIT Group
    • Abelson & Sussman
    • Knight

• And others…
  – Smart Dust
  – Programmable Matter
  – Quantum-Dot Cell Automaton
Cell Membrane Model

- Control of computation process using membrane
- Supercell system = universal computation model

\( G = \{ V, \mu, M_1, \ldots, M_4, (R_1, \rho_1), \ldots, (R_4, \rho_4), 4 \} \)

\( V = \{a, b, b', c, f\} \) \quad \text{alphabet}

\( \mu = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \) \quad \text{membrane structure}

\( M_i \) \quad \text{multiset of elements within membrane “i”}

\( (R_i, \rho_i) \) \quad \text{ordered set of rules within membrane “i”}
Computation of “n”^2 using cell membrane model
Amorphous Computing

- New computation paradigm for self-assembly
  - Microfabrication and cytoengineering
  - Various processors at low cost
- Computational particle
  - Small computational power and small memory
  - Random distribution, mobility
  - Asynchronous, local interaction
  - Wrong behavior, environmental influence
  - Identical program
  - No knowledge of their location nor orientation
  - Short distance (radius: r) communication with the neighboring particles
- Massive parallel computation system as a whole
- Simulation of the self-assembly of a circuit
What is Amorphous Computing?

• Background
  – Microfabrication and cytoengineering
  – Developing various processors at low cost
    (Not necessary to work precisely)
  – Study as a new computational paradigm

• Developing the model as an aggregate of “computational particles” that are randomly distributed and interact locally and asynchronously

• How can it be programmed effectively?
  – Relation to the formation of biological structure?
  – Is it possible to use biology for implementation, not just as a metaphor?
Characteristics of Computational Particles

- Have possibility of failure
- Will be influenced by the environment
- May make some movements
- May move around
- Have small computational power and small memory
- All particles are programmed identically (Capable of staying locally and generating random numbers)
- Have no knowledge about their location and orientation
- Make short distance (radius: r) communication with neighboring particles
- Massive parallel computational system as a whole
Pattern Formation Using Wave Propagation

• Start with first “anchor” particle, and convey the message (with information of the hop)
• Related to biological pattern formation
• “Impediment to growth” and “tropism” can be programmed using 2 anchor particles
• Program with Coore’s growing-point language (GPL), and compile to set into a particle
Quantum Dot Computer

• Bluffing?
• Different from quantum computers
• Quantum dot cell automaton (QCA)
  – Line up 4 quantum dots like dominoes
  – Electrons move inside dominoes (cells) by tunnel effect
  – The condition transmits by interaction of dominoes
• No need for wiring?
• Still, quantum dots must be arranged properly
Homework

① Explain methods of;
  - DNA/RNA secondary structure prediction
  - Minimum energy and partition function calculation using dynamic programming
  - Sequence design using secondary structure (use references)

② Explain DNA self-assembly and possibility of realization and applications of molecular machines
BASICS
DNA

• Sugar
  – Deoxyribose

• Phosphate

• Bases
  – Purine Bases --- 2 rings (hexagon and pentagon)
    • Adenine (A)
    • Guanine (G)
  – Pyrimidine Bases --- 1 ring (hexagon)
    • Thymine (T)
    • Cytosine (C)
Experimental Operations

• PCR (Polymerase Chain Reaction)
• Gel electrophoresis
• Affinity separation
• Restriction enzyme digestion
• Coupling with ligase
• Cloning and sequencing
PCR (Polymerase Chain Reaction)
Gel Electrophoresis

long molecule → short molecule

polyacrylamide gel electrophoresis
Dissociation of Single-Stranded DNA

- target molecule
- probe
- complementary
- biotin
- avidin bead
- extract
- magnet
- annealing
Secondary Structure of DNA (RNA) and Its Prediction
Secondary Structure of DNA (RNA)

• A set of base pair \(i,j\)

• \(k\)-loop --- a loop closed by \(k\) base pairs
  
  – 1- loop
    • Hairpin
  
  – 2- loop
    • Stack
    • Bulge
    • Interior
  
  – Multiple loop

• Energy is assigned to each loop
Assign energy to each of these structures (nearest neighbor model)
Dynamic Programming

- \( W(i, j) \) : Minimum energy between \( i \)-th and \( j \)-th bases
- \( V(i, j) \) : Minimum energy when \( i, j \) form a pair

\[
W(i, j) = \min(W(i+1, j), W(i, j-1), V(i, j), \min_{i \leq k < j}(W(i, k)+W(k+1, j)))
\]

\[
V(i, j) = \min(eh(i, j), es(i, j)+V(i+1, j-1), VBI(i, j), VM(i, j))
\]
- \( eh(i, j) \) : Hairpin energy
- \( es(i, j) \) : Stack energy
Dynamic Programming

- \( \text{VBI}(i, j) = \min(\text{ebi}(i, j, i', j') + V(i', j')) \)
  
  \[ i < i' < j' < j \]
  
  \[ i' - i + j - j' > 2 \]

  - \( \text{ebi}(i, j, i', j') \): Interior loop energy
  
  \( \rightarrow O(n^4) \)

- \( \text{VM}(i, j) = \min(W(i+1, k) + W(k+1, j-1)) \)
  
  \[ i < k < j - 1 \]

  - When multiple loop energy is 0
Interior Loop

- If interior loop energy $ebi(i, j, i', j')$ is proportional to the length of the loop, $(i' - i + j - j') \times c$

- $VBI(i, j) = \min_{l}(VBI(i, j, l))$

- $VBI(i, j, l) = \min(VBI(i+1, j, l-1) + c, VBI(i, j-1, l-1) + c, \ c \times l + V(i+1, j-l+1), \ c \times l + V(i+l-1, j-1))$

$\rightarrow O(n^3)$
Multiple Loop

• Approximate energy of a multiple loop:
  \[ a + b \times k' + c \times k \]
  
  \( k' \): Number of bases outside pairs
  \( k \): Number of pairs

  \[ \rightarrow O(n^3) \]

  \( k' = 5 \)
  \( k = 3 \)
McCaskill’s Algorithm

• Rather than calculating energy of each structure,
  Calculate energy distribution of all possible structures
  – Partition function
  – Probability of the formation of specific base pair

• Both can be calculated using dynamic programming
Basic Arrays

• $w[i,j]$  
  minimum energy between $i$ and $j$

• $w[k,j]$  
  $w[k,j]$ on condition that $k$ forms a pair
  -- reusable, only needed to memorize
  the case of $j$ and $j-1$

• $v[i,j]$  
  minimum energy between $i$ and $j$
  when $i$, $j$ form a pair

• Initialize all arrays to INF (infinite)
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        ww[i,j] = ww[i,j-1];
        if (i,j is a pair)
            ww[i,j] = min(ww[i,j], v[i,j]);
        for (temp=INF, k=i+1; k<=j; k++)
            temp = min(temp, w[i,k-1]+ww[k,j]);
        w[i,j] = min(temp, ww[i,j]);
    }
Arrays for Multiple Loop

- $v[i,j]$  
  minimum energy between $i$ and $j$ when $i, j$ form a pair

- $vm[i,j]$  
  minimum energy under assumption that the $i, j$ pair belongs to a multiple loop  
  -- includes at least one pair

- $vvm[k,j]$  
  $vm[k,j]$ on condition that $k$ forms a pair  
  -- reusable, only needed to memorize the case of $j$ and $j-1$
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--)
    {
        if (i,j is a pair) {
            v[i,j] = min(v[i,j], Hairpin energy);
            for (l=i+2; l<j-1; l++)
                for (k=l-1; k>i; k--)
                    if (k,l is a pair)
                        v[i,j] = min(v[i,j],
                                      v[k,l]+ 2-loop energy);
            for (temp=INF, k=i+2; k<=j-1; k++)
                temp = min(temp, vm[i+1,k-1]+vvm[k,j-1]);
            v[i,j] = min(v[i,j], temp+MLclosing+MLintern);
        }
        set vm and vvm;
    }
Multiple Loop

- Approximate energy of a multiple loop:
  \[ a + b \times k' + c \times k \]
  
  - \( k' \): Number of bases outside pairs
  - \( k \): Number of pairs

\[ \rightarrow O(n^3) \]

\[ k' = 5 \]
\[ k = 3 \]
Setting vm and vvm:

\[
\text{vvm}[i,j] = \text{vvm}[i,j-1] + \text{MLbase};
\]

if (i,j is a pair)
\[
\text{vvm}[i,j] = \min(\text{vvm}[i,j], v[i,j] + \text{MLintern});
\]
for (temp=\text{INF}, k=i+1; k<=j; k++) {
    \[
    \text{temp} = \min(\text{temp}, \text{vm}[i,k-1] + \text{vvm}[k,j]);
    \]
    \[
    \text{temp} = \min(\text{temp}, \text{MLbase}*(k-i)+\text{vvm}[k,j]);
    \]
} \[
\text{vm}[i,j] = \min(\text{temp}, \text{vvm}[i,j]);
\]
Partition Function

- With state energy $G$, the probability of state occurrence is proportional to Boltzmann factor $\exp(-G/kT)$

- Partition function $Z$ is a sum of the Boltzmann factors of all states

- Probability of state occurrence of energy $G$ is given by $\exp(-G/kT)/Z$
Calculation of Partition Function

- Instead of calculating the minimum energy while traversing secondary structures, calculate the sum of the Boltzmann factors while traversing secondary structures

<table>
<thead>
<tr>
<th>Minimum Energy</th>
<th>Partition Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>( \exp(-G/kT) )</td>
</tr>
<tr>
<td>initial value INF</td>
<td>initial value 0</td>
</tr>
<tr>
<td>min</td>
<td>+</td>
</tr>
<tr>
<td>+</td>
<td>*</td>
</tr>
</tbody>
</table>
Basic Arrays

- $w[i,j]$  
  partition function between $i$ and $j$
- $ww[k,j]$  
  $w[k,j]$ on condition that $k$ forms a pair  
  -- reusable, only needed to memorize the cases of $j$ and $j-1$
- $v[i,j]$  
  partition function between $i$ and $j$ when $i$, $j$ form a pair
- Initialize all arrays to $\text{INF (infinite)}$
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        ww[i,j] = ww[i,j-1];
        if (i,j is a pair)
            ww[i,j] = ww[i,j] + v[i,j];
        for (temp=0, k=i+1; k<=j; k++)
            temp = temp + w[i,k-1]*ww[k,j];
        w[i,j] = temp + ww[i,j];
    }
Arrays for Multiple Loop

• $v[i,j]$
  partition function between $i$ and $j$
  when $i, j$ form a pair

• $vm[i,j]$
  partition function under the assumption
  that the $i, j$ pair belongs to a multiple loop
  -- includes at least one pair

• $vvm[k,j]$
  $vm[k,j]$ on condition that $k$ forms a pair
  -- reusable, only needed to memorize
  the cases of $j$ and $j-1$
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        if (i,j is a pair) {
            \( v[i,j] = v[i,j] + \text{Hairpin partition function}; \)
            for (l=i+2; l<j-1; l++)
                for (k=l-1; k>i; k--)
                    if (k,l is a pair)
                        \( v[i,j] = v[i,j] + v[k,l] \times \text{2-loop partition function}; \)
            for (temp=0, k=i+2; k<=j-1; k++)
                temp = temp + \( v_{m}[i+1,k-1] \times v_{vm}[k,j-1]; \)
            \( v[i,j] = v[i,j] + \text{temp} \times \text{expMLclosing} \times \text{expMLintern}; \)
        }
        set \( v_{m} \) and \( v_{vm}; \)
    }
Setting \(vm\) and \(vvm\):

\[
\begin{align*}
\text{vvm}[i,j] &= \text{vvm}[i,j-1]\times \exp\text{MLbase}; \\
\text{if (i,j is a pair)} \\
&\quad \text{vvm}[i,j] = \text{vvm}[i,j]+v[i,j]\times \exp\text{MLintern}; \\
\text{for (temp=0, k=i+1; k<=j; k++)} \\
&\quad \text{temp} = \text{temp}+\text{vm}[i,k-1]\times \text{vvm}[k,j]; \\
&\quad \text{temp} = \text{temp}+\exp\text{MLbase}^{(k-i)}\times \text{vvm}[k,j]; \\
&\quad \text{vm}[i,j] = \text{temp}+\text{vvm}[i,j];
\end{align*}
\]