Tutorial on Molecular Computing

M. Sakthi Balan

ECOM Research Lab
Education & Research
Infosys Techologies Limited
Bangalore

Contents

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Contents

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Contents

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Contents

1. **Introduction**
   - Computing
   - DNA strands
   - Peptides and Antibodies
2. **DNA Computing**
   - What is DNA Computing
   - First papers on DNA Computing
   - Theoretical Models
3. **Peptide Computing**
   - As a Computational Model
   - Solving SAT Problem
   - Hamiltonian Path Problem
   - Theoretical Models
4. **Remarks and Open Problems**

Infosys

Outline

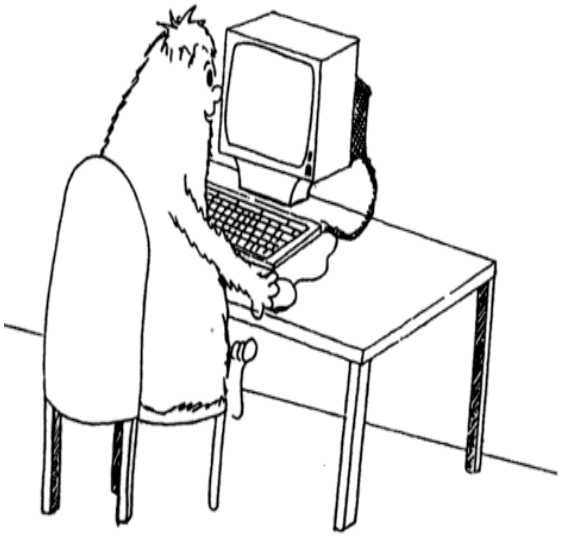
1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

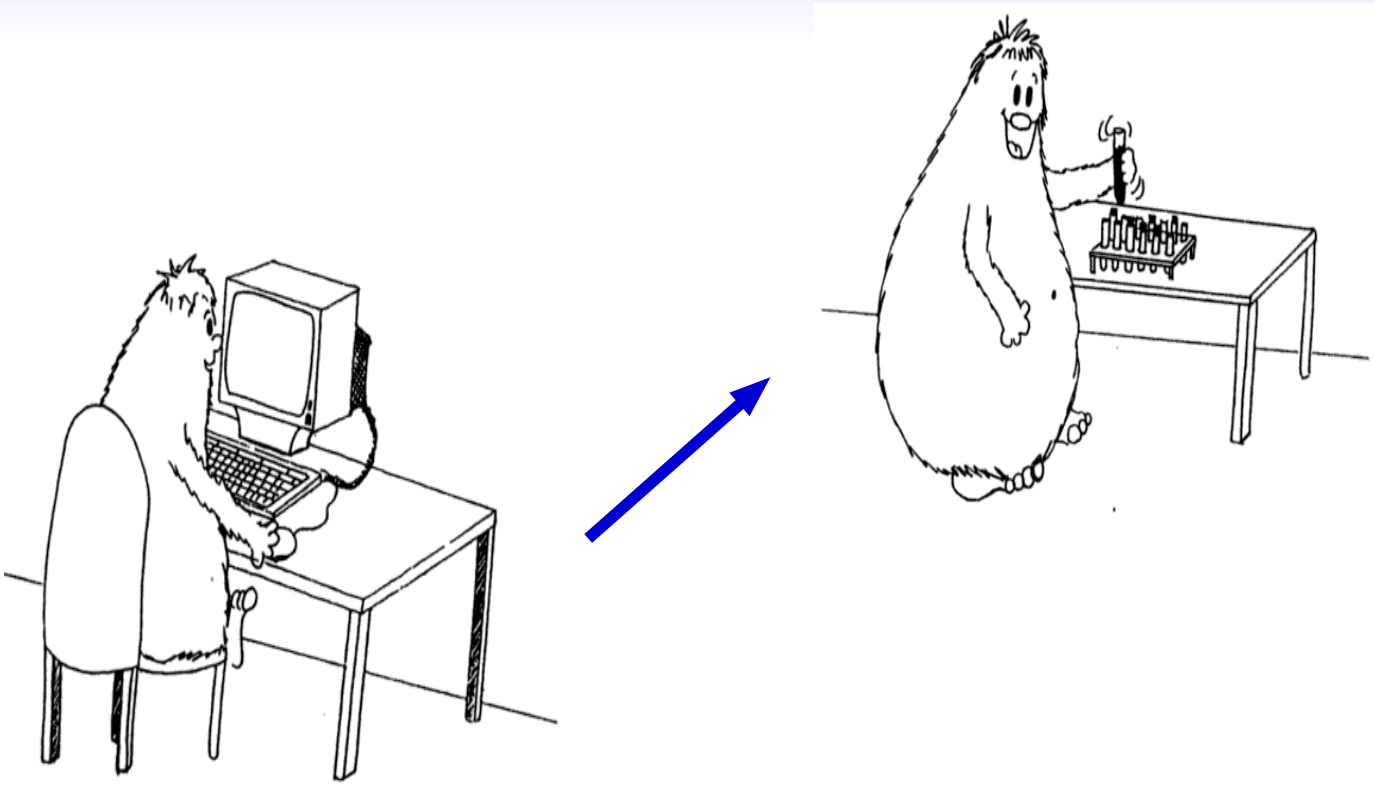
2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

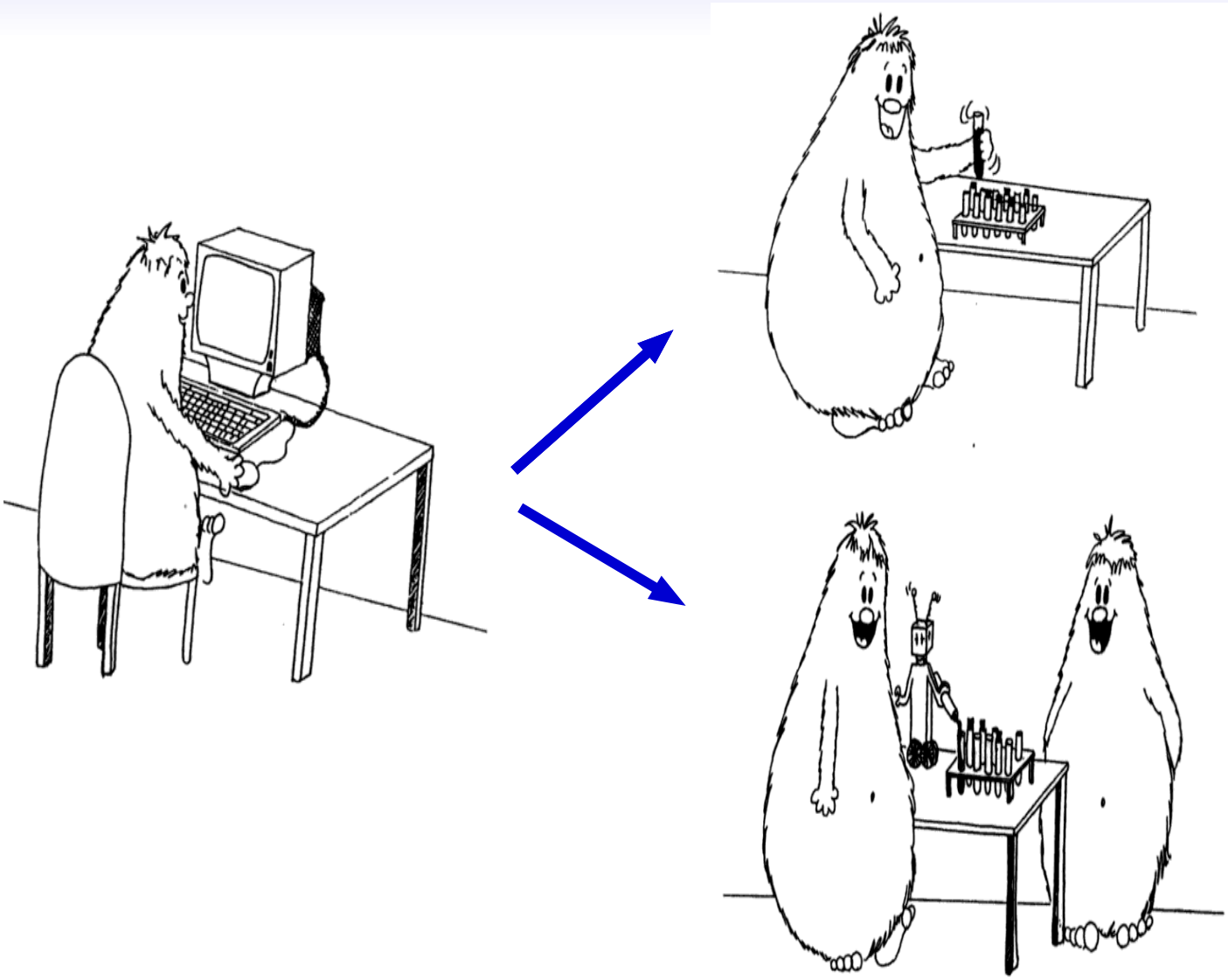
3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys







Computing...

What is computing...

Turing model:

- Finite number of symbols.
- Finite number of state of the system.
- Transition of the system.
- Head reading finite number of symbols at a time.
- An infinite tape to (re)write.

Silicon Computers

Facts

- Computes “fast”,
- Stores huge amount of data,
- Retrieves data fast,
- Communicates fast.

Limits

- Problems that have no efficient algorithms - Is NP=P? is still open.
- Physical limits in terms of speed.
- Deterministic and sequential machines.
- Lack of machines that can do intellectual work on our behalf.
- Security, fault-tolerance – attacks, faults are pre-defined.

From *Computing with Cells and Atoms* by Cristian S. Calude and Gh. Păun

...It seems that progress in electronic hardware (and the corresponding software engineering) is not enough; for instance, the miniaturization is approaching the quantum boundary, where physical processes obey laws based on probabilities and non-determinism, something almost completely absent in the operation of classical computers. So, new breakthrough is needed...

Infosys

Natural Computing

Bio

- DNA hybridization
- Immune reaction

Others

- Quantum mechanical phenomena
- Reaction-diffusion process

Infosys

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

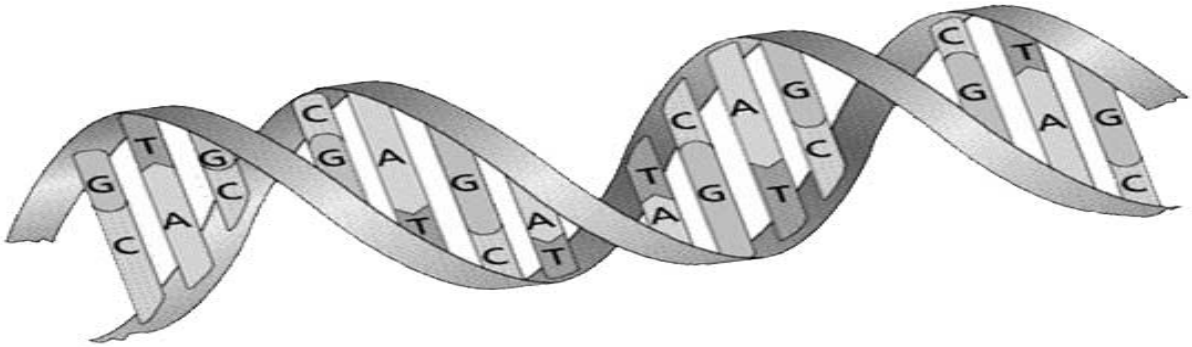
2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

DNA Strands



- DNA consists of polymer chains – DNA strands.
- Chain consists of nucleotides that differ only in their bases.
- There are four bases: *A* (adenine), *G* (guanine), *C* (cytosine) and *T* (Thymine).
- Double-helical structure is formed through bonding of two strands.
- *A* always bonds with *T* and *G* with *C*.

Infosys

Operations on DNA

- The length of a DNA strand can be measured using gel electrophoresis.
- A known DNA strand can be fished in a solution containing many DNA strands using a filtering method.
- A double stranded DNA can be denatured into two single strands by heating.
- Two DNA strands can be hybridized into a single double stranded DNA molecule (DNA strands bind together respecting the Watson-Crick complementary property).

Infosys

Operations on DNA

- A class of enzymes called polymerases can lengthen a partially double stranded to make it as a complete double stranded molecule.
- Enzymes called restriction endonucleases cut DNA strands at the specific site where a specific sequence of nucleotides are present.
- Enzymes ligases can paste two DNA strands with overhanging ends provided those ends are Watson-Crick complementary of each other. This process is called ligation.
- Specific set of DNA sequences can be multiplied using a polymerase chain reaction.
- The exact sequence in the DNA strand can be found out by polymerase action on the strand. This process is called sequencing.

Infosys

DNA Strands

```
TATAGCCGCTCGATTACGGC
          GCTAATGCCG CGGCGCGTAT
```

Two sticky ends

DNA Strands

```
TATAGCCGCTCGATTACGGC
                GCTAATGCCG CGGCGCGTAT
```

Two sticky ends

```
TATAGCCGCTCGATTACGGC
                GCTAATGCCG
```

One sticky end

DNA Strands

```
TATAGCCGCTCGATTACGGC
                GCTAATGCCG CGGCGCGTAT
```

Two sticky ends

```
TATAGCCGCTCGATTACGGC
                GCTAATGCCG
```

One sticky end

```
TATAGCCGCTCGATTACGGC  GCCGCGCATATACGATGTAT
                GCTAATGCCG CGGCGCGTAT
```

Two sticky ends

DNA Strands

```
TATAGCCGCTCGATTACGGC
                GCTAATGCCG CGGCGCGTAT
```

Two sticky ends

```
TATAGCCGCTCGATTACGGC
                GCTAATGCCG
```

One sticky end

```
TATAGCCGCTCGATTACGGC  GCCGCGCATATACGATGTAT
                GCTAATGCCG CGGCGCGTAT
```

Two sticky ends

```
GCCGCGCATATACGATGTAT
```

Single strand

About complementarity

- Watson-Crick complementarity is given by nature.
- Without complementarity it will be many-many relations which will be quite uninteresting.
- Note that on the condition that the bases are complementary in nature the two strands bind – hence in the perspective of computation we can view it as *in vitro* the hybridization takes place on some condition *D* being satisfied.
  - this gives the notion of computing.
  - this also resembles the transition function of a Turing machine.
  - this can be exploited at least for *in vitro* experiments.

Infosys

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Peptides and Antibodies

- Peptides – short proteins – sequence over 20 basic amino acids.
- Interactions between peptides and antibodies – Immune reactions.
- Antibodies recognize specific sequence in peptides – epitopes.
- Affinity power of antibodies presents an option to remove and attach antibodies – resembles a rewriting system

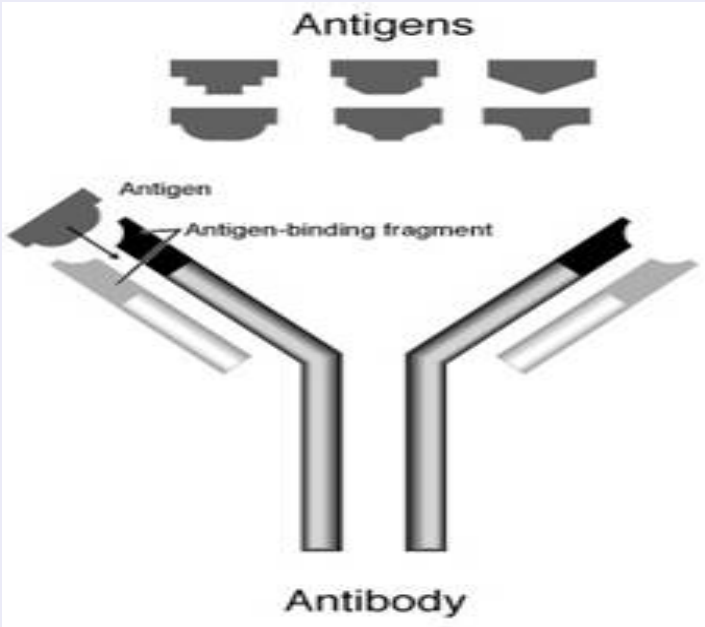
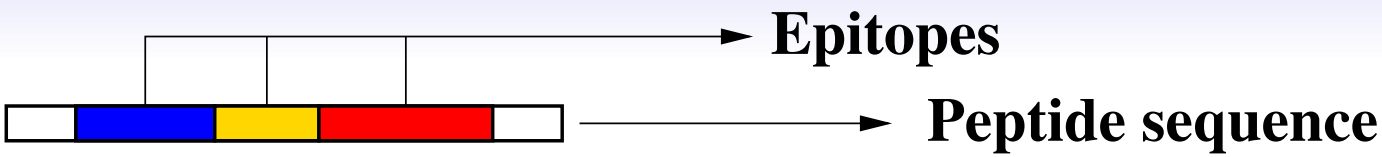
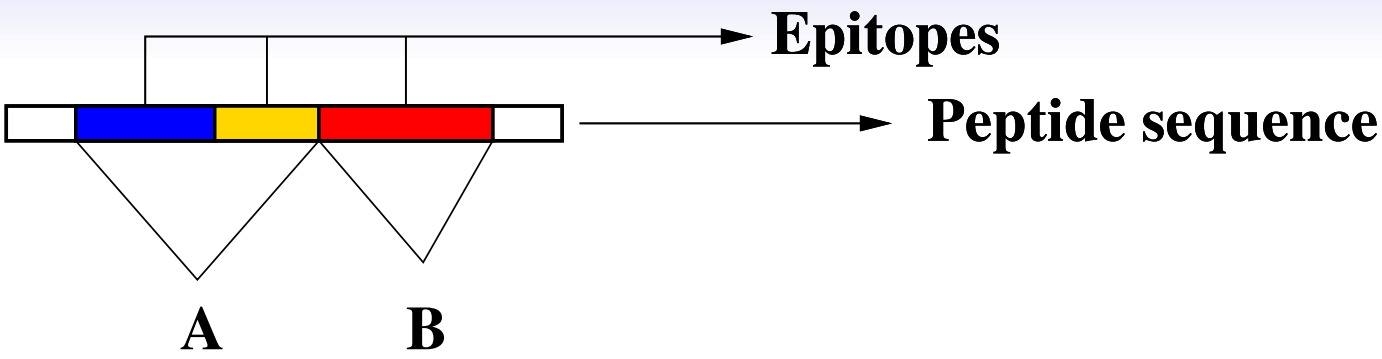
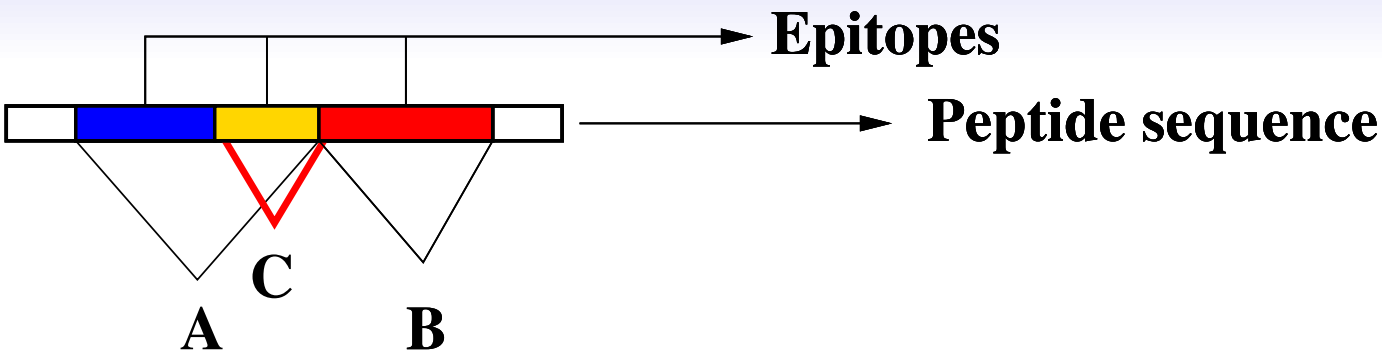


Figure: Antibody-antigen binding

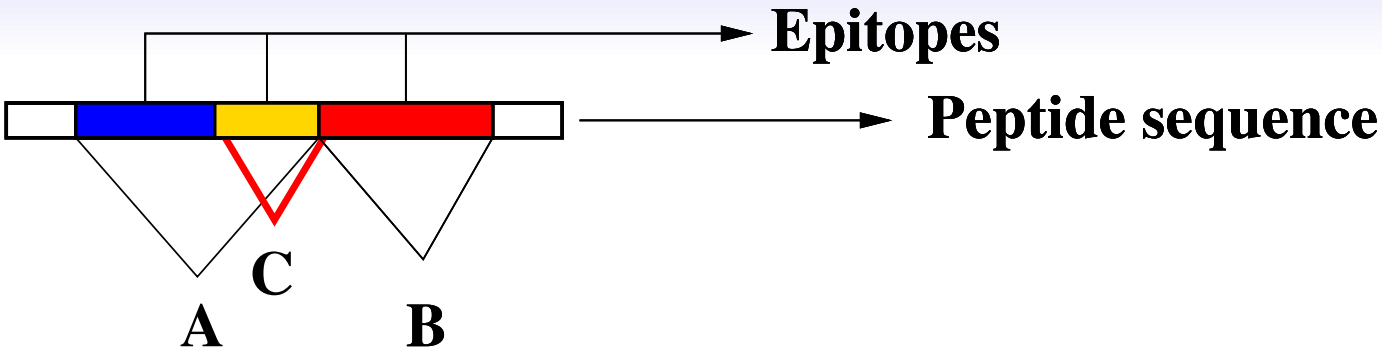

Epitopes
Peptide sequence



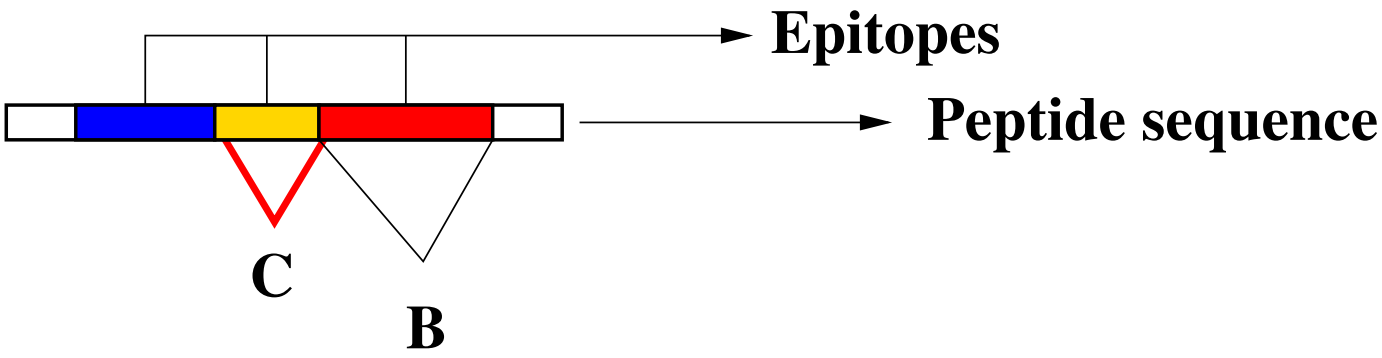
Peptide sequence with antibodies



Peptide sequence with antibodies



Peptide sequence with antibodies



Peptide sequence with antibodies

Peptides and Antibodies

- Epitopes for different antibodies or same antibody can overlap.
- There is a power called affinity associated with the binding of antibodies to its epitopes.
- One antibody can have more than one epitope to bind with.
- There can be many antibodies that bind to a single or overlapping epitopes.

Infosys

Molecular Computing

- Interactions between molecules as a computing model.
  - DNA hybridization,
  - DNA splicing,
  - Binding of antibodies to epitopes and so on.
- Massively parallel and non-deterministic.
  - Multiple copies of molecules – multiset.
- Has the potential to solve hard problems easily.
  - Brute force in a massively parallel way.
- Energy efficient.

Infosys

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

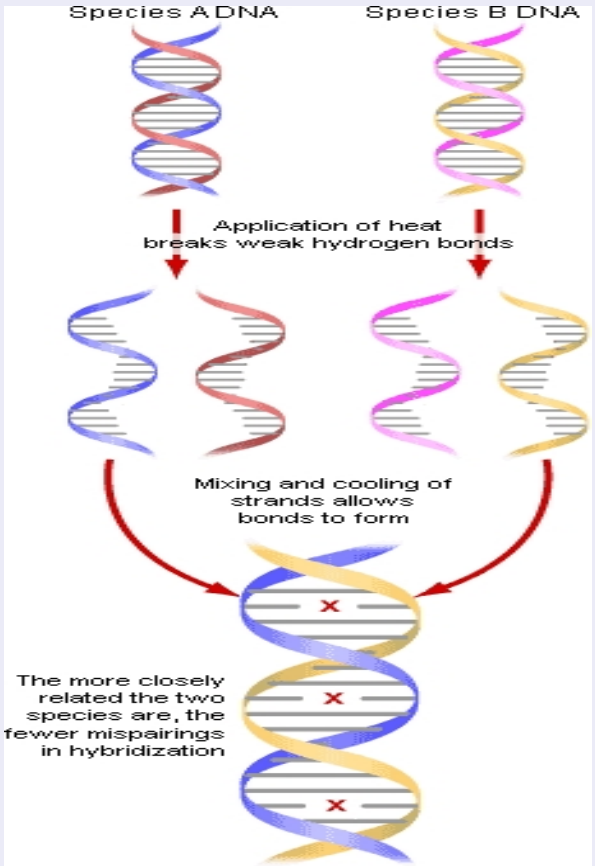
4 Remarks and Open Problems

Infosys

DNA Computing

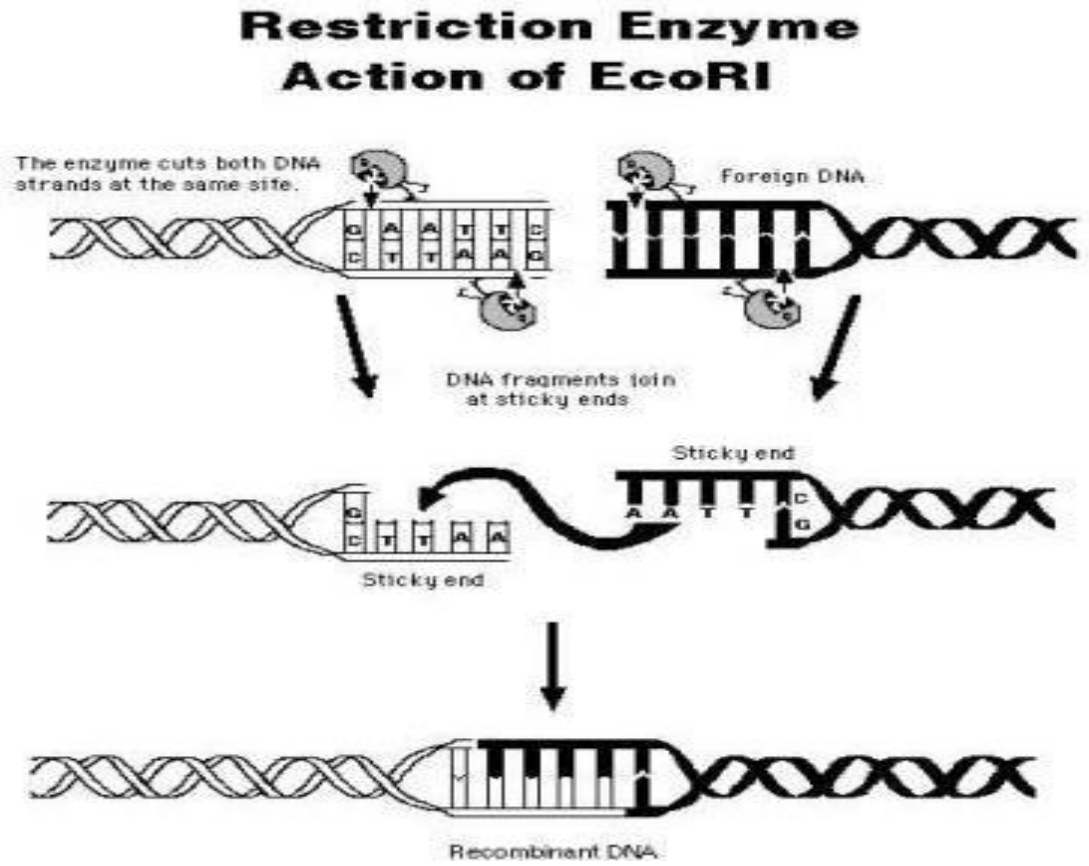
- Uses DNA strands and the interactions between strands as operations.
- Interactions are DNA hybridization, splicing and so on.
- Note that we will be having multiple copies of each strands so many things happen at the same time.
- Hence it is massively parallel and highly non-deterministic.

DNA hybridization



- DNA splicing resembles a rewriting process.
- Cutting of two DNA strands at specific sites and making fragments of DNA.
- If the sticky ends of the fragments of DNA are complementary in nature then they recombine to form new DNA strands.
- Note that there will be several copies of DNA strands floating around – Parallel and non-deterministic.

Recombinant DNA



DNA Computing

- Prepare specific DNA strands according to the problem.
- Allow them to hybridize and form various strands that denotes all possible solutions of the problem.
- Explore those DNA strands and eliminate the strands that will not lead to solution.
- At the end of final elimination step if we find a strand then that will be the solution for the problem.
- Adleman's experiment in 1994 (Science) to find Hamiltonian path of the given instance of a graph.

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Tom Heads' work

- Tom Head in 1987 gave a mathematical model for the recombinant behavior of DNAs by defining Splicing systems.
  - His work combined Formal language theory and Molecular biology.
  - He treated a DNA strand as a linear string or word over 4 alphabet $[A/T]$, $[C/G]$, $[G/C]$ and $[T/A]$.
  - His definition of splicing system resembles the recombinant behavior of DNA.
  - His study was aimed at the analysis of the generative capacity of these type of systems.

Adleman's' Experiment

- Adleman in 1994 took an instance of a graph and using DNA strands and DNA hybridization solved the Hamiltonian path problem in the wetlab.
- Basic idea in this work was elimination method.
- Using inherent parallelism in the model explore all the possible paths and eliminate those not satisfying the conditions for a Hamiltonian path.

Adleman's' Experiment

Problem statement

The Hamiltonian path problem (HPP) is stated as follows:

Input: A directed graph G with n vertices, among which v_{in} and v_{out} are designated vertices.

Output: *Yes* if any path remains, *No* otherwise.

Adleman's' Experiment

Non-deterministic algorithm:

1. Random paths in G are generated in large quantities.
2. Paths that do not begin with v_{in} and end in v_{out} are eliminated.
3. All paths that do not involve exactly n vertices are rejected.
4. Keep only those paths where all n vertices are represented.

Infosys

Adleman's' Experiment

Preprocessing steps:

- Specific DNA strands are selected for each vertices.
- For each vertex i associate a random 20-mer sequence, say s_i.
- For each edge e_{ij} from vertex i to vertex j choose a 20-mer sequence s_{ij} where
  - If the vertex is not a start or end vertex then the prefix consists of 10-mer suffix of s_i and 10-mer prefix of s_j.
  - If the vertex is a start or end vertex then the sequence will be whole of s_i.

Infosys

ATATCGGCGAGCTAATGCCG CGGCGCGTATATGCTACATA

Figure: DNA strands for vertices

GCTAATGCCGCGGCGCGTAT

Figure: DNA strand for edge

TATAGCCGCTCGATTACGGC GCCGCGCATATACGATGTAT
GCTAATGCCG CGGCGCGTAT

Figure: After DNA hybridization

Adleman's' Experiment

Generating paths

- For each vertex i, $\bar{s}_i$ and for each edge e_{ij} , the sequence s_{ij} are mixed and put in a soup.
  - The DNA sequence $\bar{s}_i$ serves as a splint to bring nucleotides associated with compatible edges together for ligation.
  - At the end of this step we have generated all possible paths and much more in the graph.

Adleman's' Experiment

Generating paths

- Next step is elimination:
  - Using filtering technique keep those sequences that start with vertices v_{in} and v_{out}.
  - Check the length of the sequences to see if it has n vertices in it (it has to be of length $20 \times n$ (In this case it has to be length 140-mer, since there are 7 vertices).
  - Do a filtering method again to check if all n vertices are in the path..
- If there are any DNA strands left in the soup then the answer is there is a Hamiltonian path in the graph or else there is no Hamiltonian path.

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Splicing system

- Splicing system is also called as H-system.
- The generative capacity of H-system is studied with respect to the Chomskian hierarchy of languages.
- A splicing operation over two words is defined as follows:

  Let the splicing rule be given by $(u_1; u_2; u_3; u_4)$ where u_is are strings over a finite alphabet.

  The result of splicing x and y are z and w if and only if $x = x_1 u_1 u_2 x_2, y = y_1 u_3 u_4 y_2$ and $z = x_1 u_1 u_4 y_2, w = y_1 u_3 u_2 x_2$.
- The splicing system or H-system is a generative system that uses this splicing operation as a basic tool.

Definition

An extended H system is a quadruple $\gamma = (V, T, A, R)$ where V is an alphabet, $T \subseteq V$, $A \subseteq V^*$, and $R \subseteq V^*\#V^*\$V^*\#V^*$; $\#, \$$ are special symbols not in V.

V is the alphabet of, T is the terminal alphabet, A is the set of axioms, and R is the set of splicing rules; the symbols in T are called terminals and those in $V - T$ are called nonterminals.

For $x, y, z, w \in V^*$ and $r = u1\#u2\$u3\#u4$ in R we define $(x, y) \vdash_r (z, w)$ if and only if $x = x_1 u_1 u_2 x_2$, $y = y_1 u_3 u_4 y_2$ and $z = x_1 u_1 u_4 y_2, w = y_1 u_3 u_2 x_2$ for some $x_1, x_2, y_1, y_2 \in V^*$

Definition

For an H system $\gamma = (V, T, A, R)$ and for any language $L \subseteq V^*$, we write

$$\sigma(L) = \{z \in V^* \mid (x, y) \vdash_r (z, w) \text{ or } (x, y) \vdash_r (w, z), \text{ for some } x, y \in L;$$

and we define

$$\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L),$$

where

$$\sigma^0(L) = L, sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)),$$

for $i \geq 0$.
The language generated by the H system is defined by

$$L(\gamma) = \sigma^*(A) \cap T$$

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Peptide Computing

- Proposed by H. Hug and R. Schuler [Hug,Schuler 2001].
- Solve some difficult combinatorial problems.
  - Satisfiability problem.
  - Hamiltonian path problem [M.S. Balan et al 2002].
- Universally complete [M.S. Balan et al 2002].

Peptide Computing

- Peptides are sequence over 20 basic amino acids.
- Interactions between peptides and antibodies is the basic operations.
- Antibodies recognizes a subsequence of a peptide, called epitopes, by binding to it.
- Binding of antibodies to epitopes has associated power called affinity.
- Higher priority to the antibody with larger affinity power.
- Affinity power of antibodies presents an option to remove and attach antibodies – resembles a rewriting system.

Infosys

Computing Model

- Form peptide sequences for each possible solution of the given problem – preprocessing step.
- Choose antibodies according to the current instance of the problem.
- Eliminate those sequences that will not lead to solutions.
- End of experiment – if the fluorescence is detected then a solution exist.
- Decipher the sequence to find the solution.

Infosys

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Satisfiability Problem

Problem

Let F be a formula over n variables. Does there exists an assignment of truth values to every variable in F such that F becomes *true*.

Satisfiability Problem

- Let F be formula in conjunctive normal form
- There are n variables
- Find an assignment that makes F *true*
- There are 2^n possible assignments

Infosys

Satisfiability Problem

Formula

$$F = (v_1 \vee \neg v_2) \wedge \neg v_2 \wedge (v_1 \vee v_2)$$

Possible Assignments

X_i	$X_i(v_1)$	$X_i(v_2)$
X_1	*false*	*false*
X_2	*false*	*true*
X_3	*true*	*false*
X_4	*true*	*true*

Infosys

Satisfiability Problem

Formula

$$F = (v_1 \vee \neg v_2) \wedge \neg v_2 \wedge (v_1 \vee v_2)$$

Possible Assignments

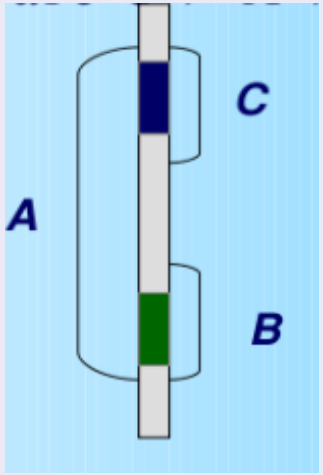
X_i	$X_i(v_1)$	$X_i(v_2)$
X_1	*false*	*false*
X_2	*false*	*true*
X_3	*true*	*false*
X_4	*true*	*true*

Infosys

Satisfiability Problem

- For each assignment prepare a peptide and different antibodies binding to overlapping epitopes
- Binding affinities are $C > A > B$

Single peptide



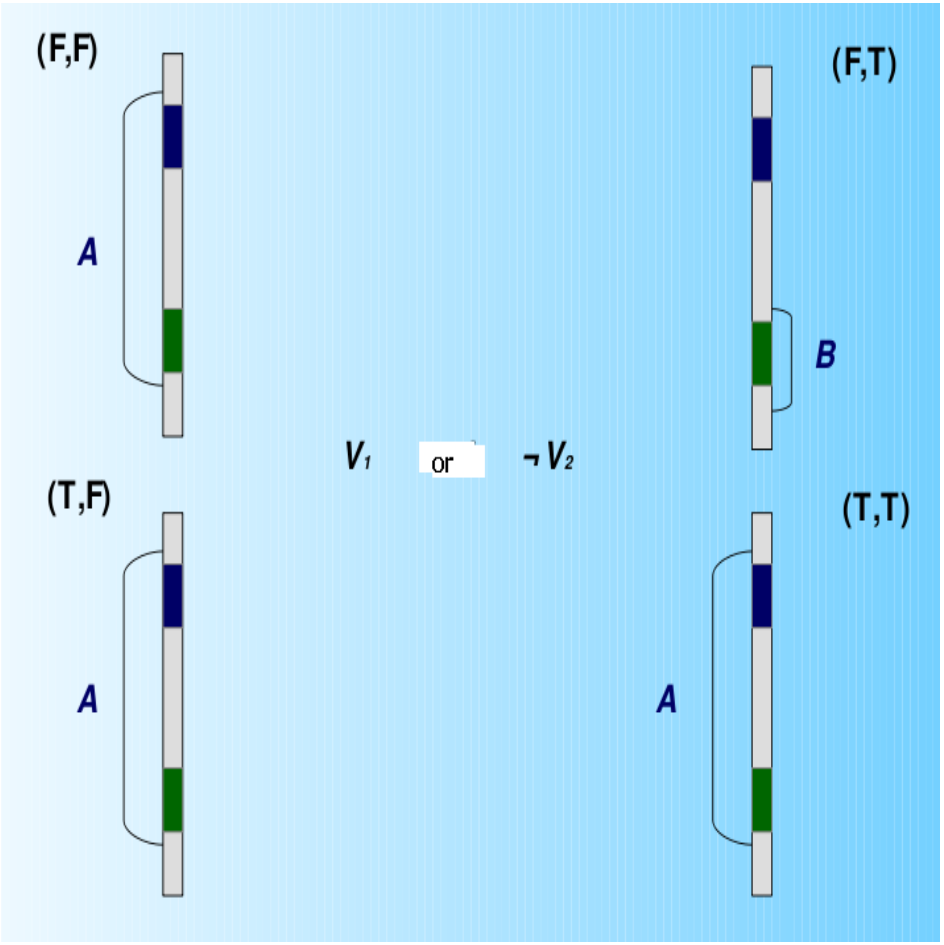
Satisfiability Problem

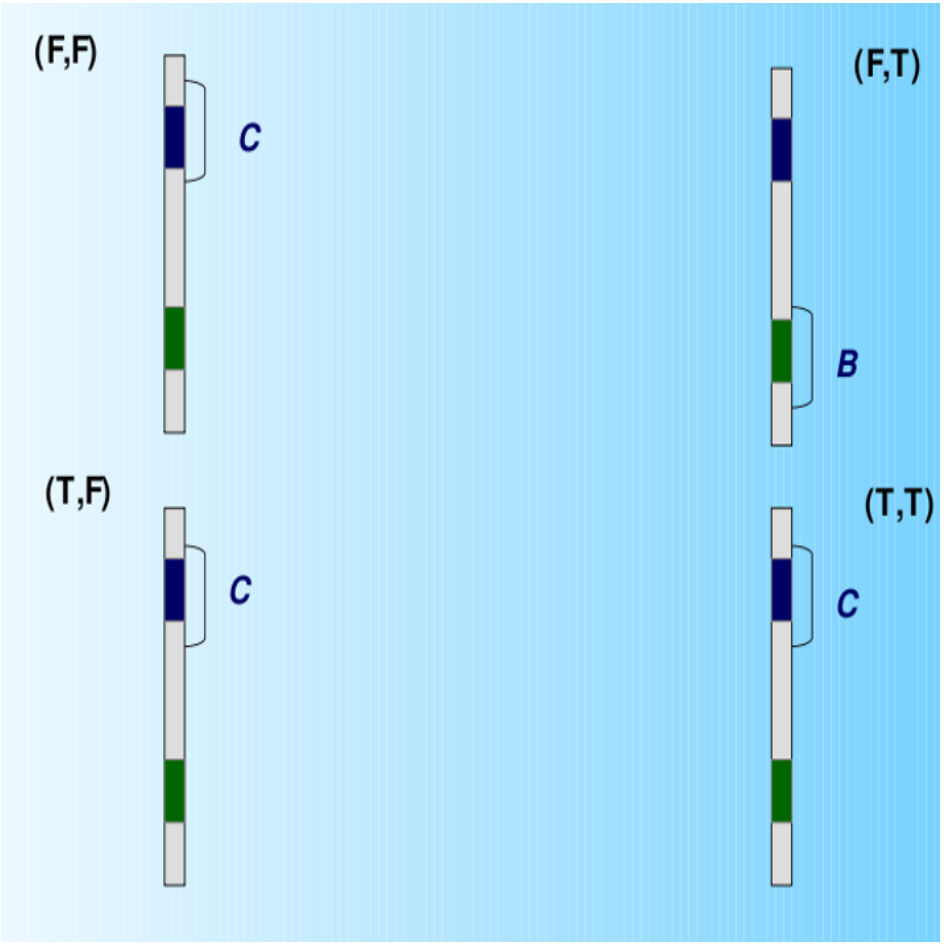
- Prepare partial solutions $G_1, G_2, \cdots, G_k$ where G_i contains antibody A if C_i is *true* under corresponding assignment X
- $G_1 = \{A_1, A_3, A_4\}, G_2 = \{A_1, A_3\}, G_3 = \{A_2, A_3, A_4\}$

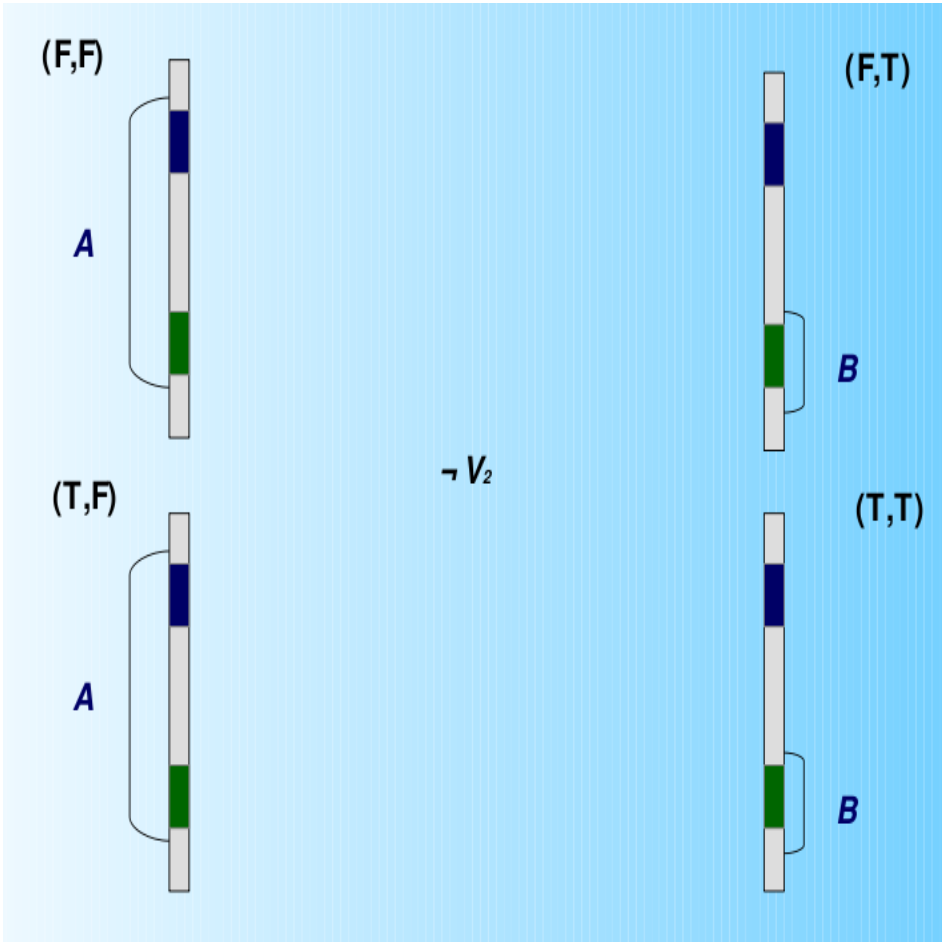
Infosys

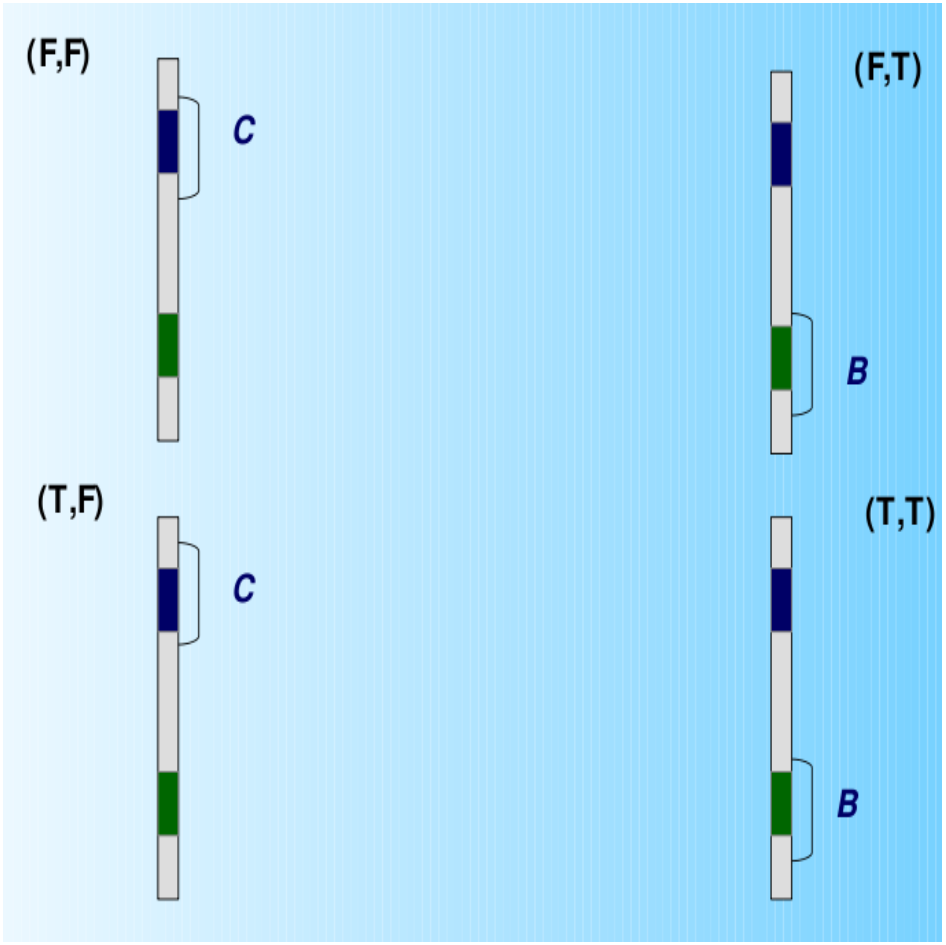
Algorithm

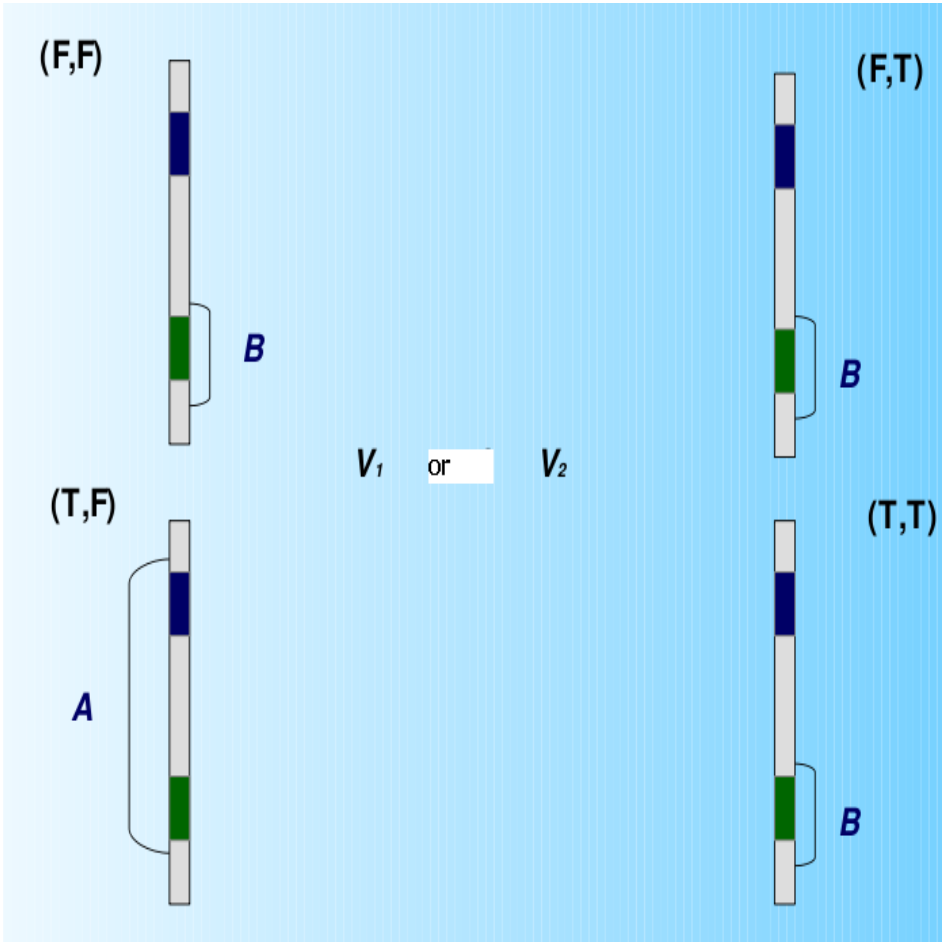
1. Let $m = k$. While $m > 0$, repeat the following steps:
   1. The antibody set G_m is added. The antibodies A of G_m bind to their epitopes.
   2. Antibodies B are added. Antibodies B bind to all free binding sites for B.
   3. Antibodies C are added. Antibody C binds to all of its epitopes, since it has the highest binding affinity. Note that all antibodies A of set G_m are removed, whereas antibodies B remain bound to their epitopes.
   4. Antibodies C are removed by adding epitopes for C in excess.
   5. All remaining antibodies are attached to their epitopes by adding linker.
   6. Let $m = m - 1$.
2. Add labelled antibody A or B.
3. Detect whether there are labelled antibodies present.

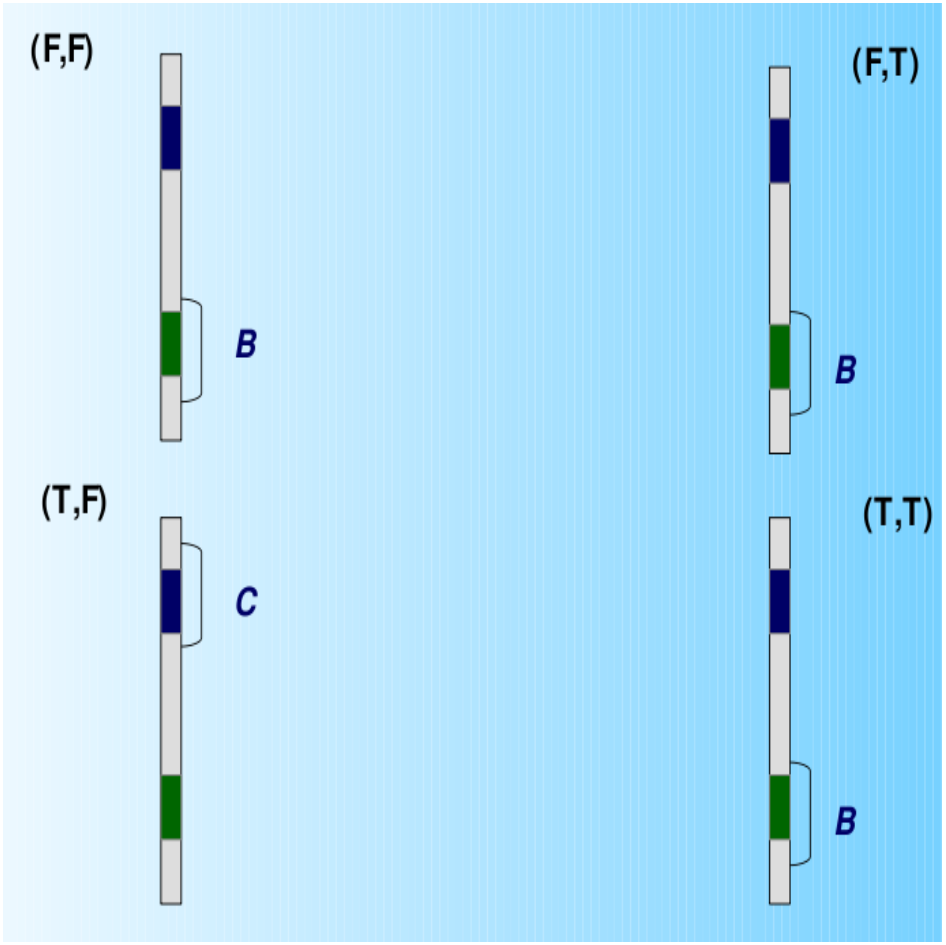

(F,F)
A
(F,T)
B
V1 or ¬V2
(T,F)
A
(T,T)
A

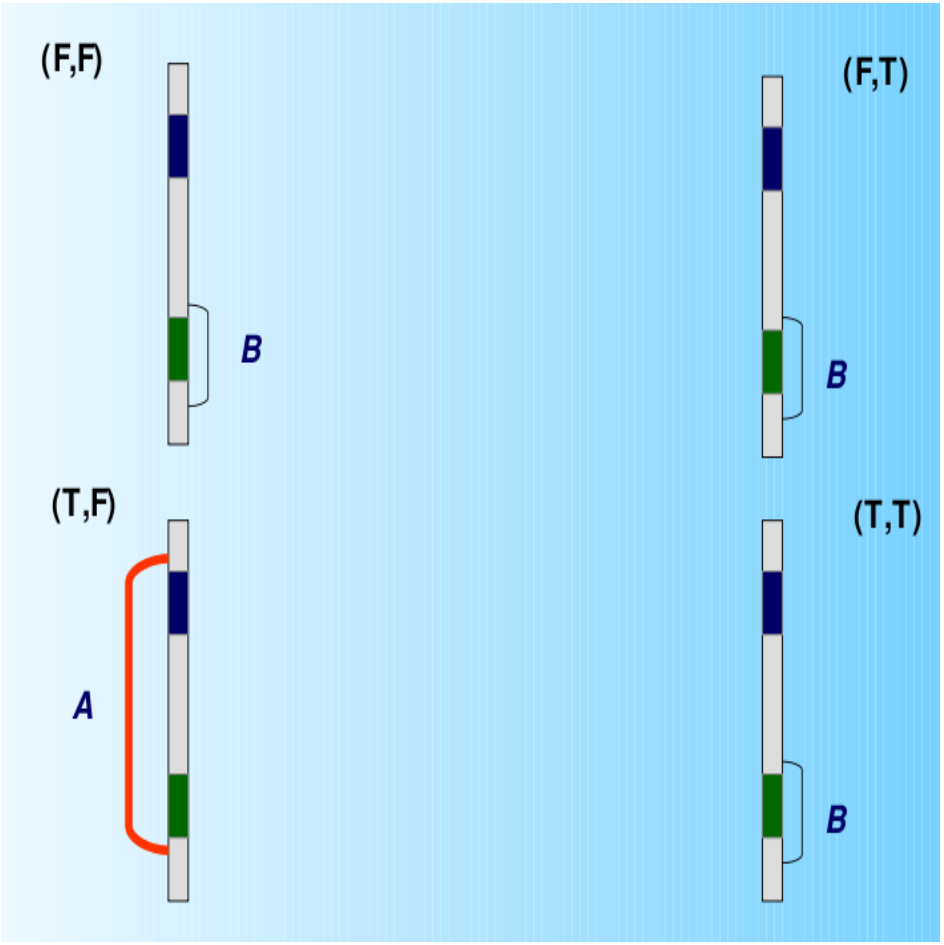

(F,F)
C
(F,T)
B
(T,F)
C
(T,T)
C


(F,F)
A
(F,T)
B
¬ V_2
(T,F)
A
(T,T)
B


(F,F)
C
(F,T)
B
(T,F)
C
(T,T)
B


(F,F)
B
(F,T)
B
V1
or
V2
(T,F)
A
(T,T)
B


(F,F)
B
(F,T)
B
(T,F)
C
(T,T)
B


(F,F)
B
(F,T)
B
(T,F)
A
(T,T)
B

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Hamiltonian Path Problem

- $G = (V, E)$ is a directed graph;
- $V = \{v_1, v_2, \cdots, v_n\}$ is the vertex set;
- $E = \{e_{ij} \mid v_i \text{ is adjacent to } v_j\}$ is the edge set
  v_1 – source vertex and v_n – end vertex.
- Problem: Test whether there exists a Hamiltonian path between v_1 and v_n.

Infosys

Peptides Formation

- Each vertex v_i has a corresponding epitope ep_i,
- Each peptide has ep_1 on one extreme and ep_n on the other extreme,
- Each peptide has a doubly duplicated permutation of $ep_2, \cdots, ep_{n-1}$ in between ep_1 and ep_n.

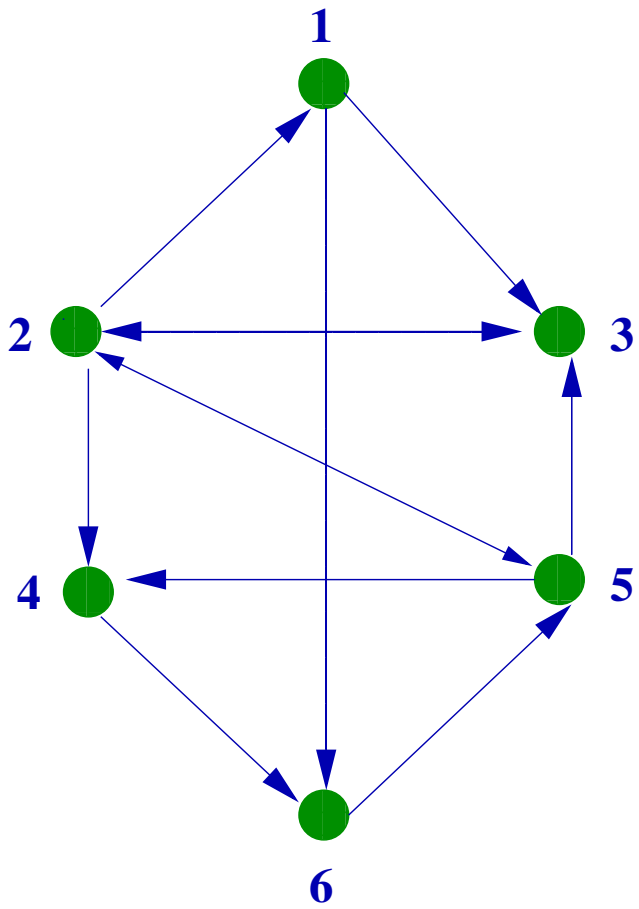
Infosys

Antibody Formation

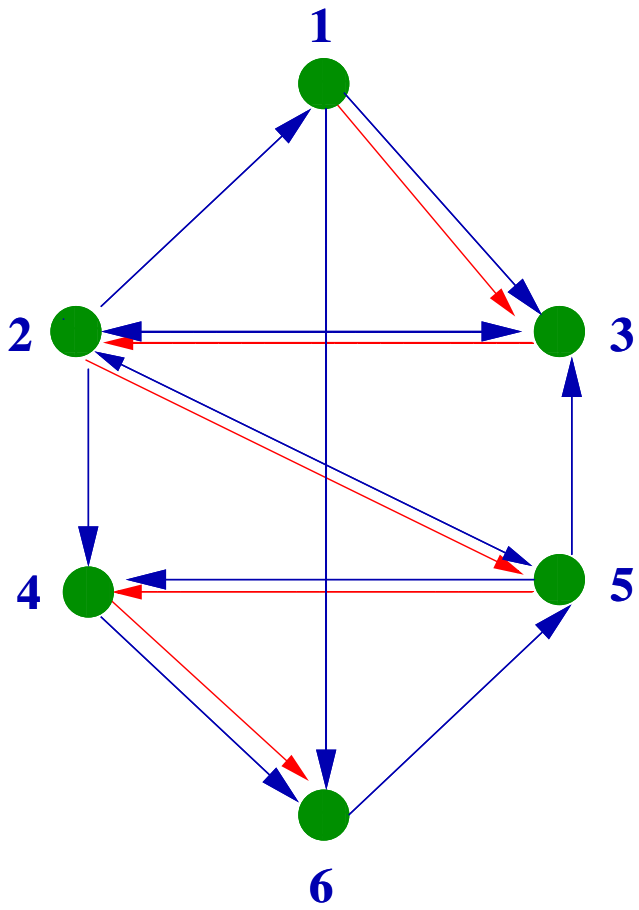
- Form antibodies $A_{ij} - site = ep_i ep_j$ if v_j is adjacent to v_i,
- Form antibodies $B_{ij} - site = ep_i ep_j$ if v_j is not adjacent. to v_i,
- Form antibody C – site is the whole of peptide,
- $Affinity(B_{ij}) > Affinity(C)$
- $Affinity(C) > Affinity(A_{ij})$

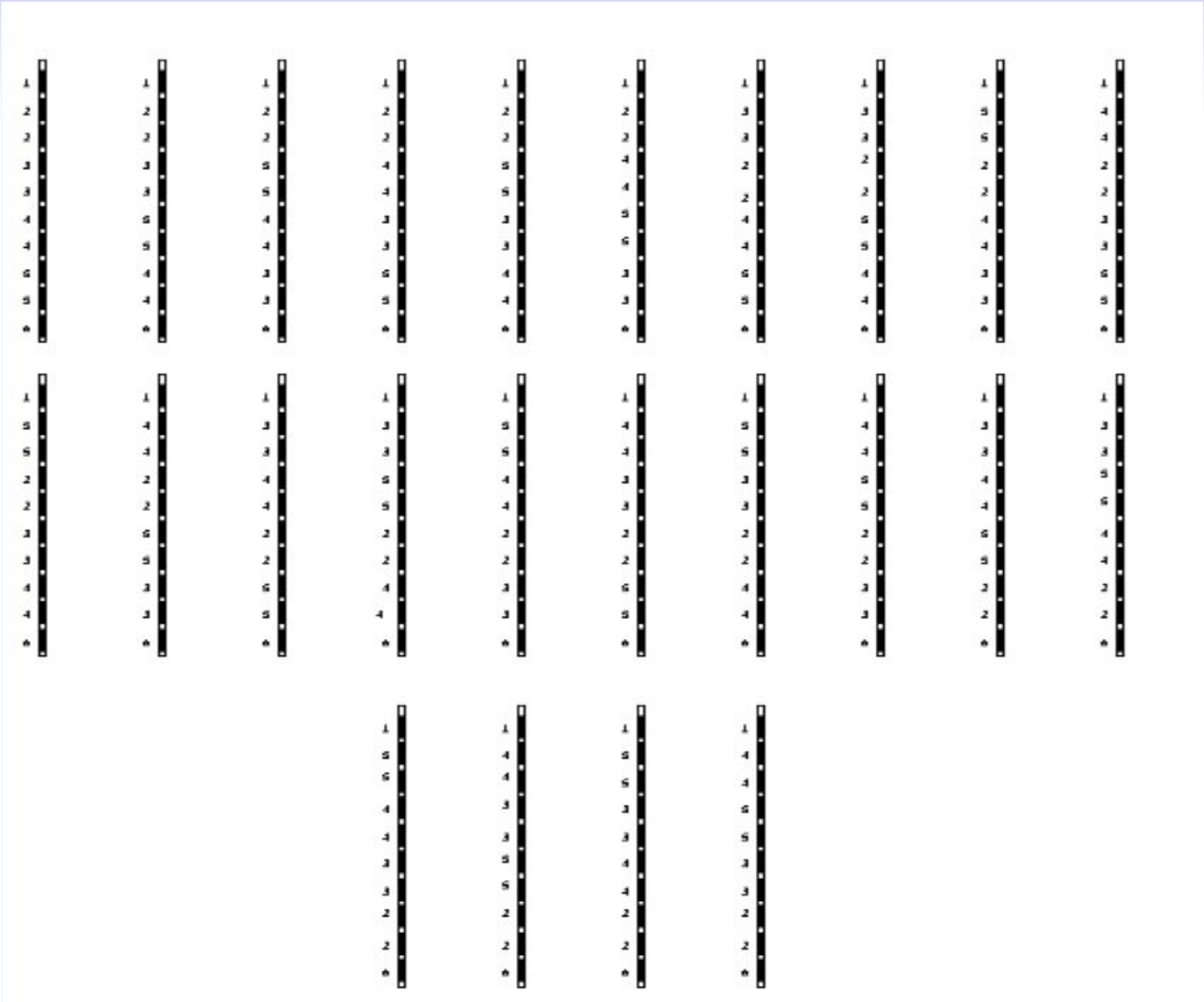
Infosys

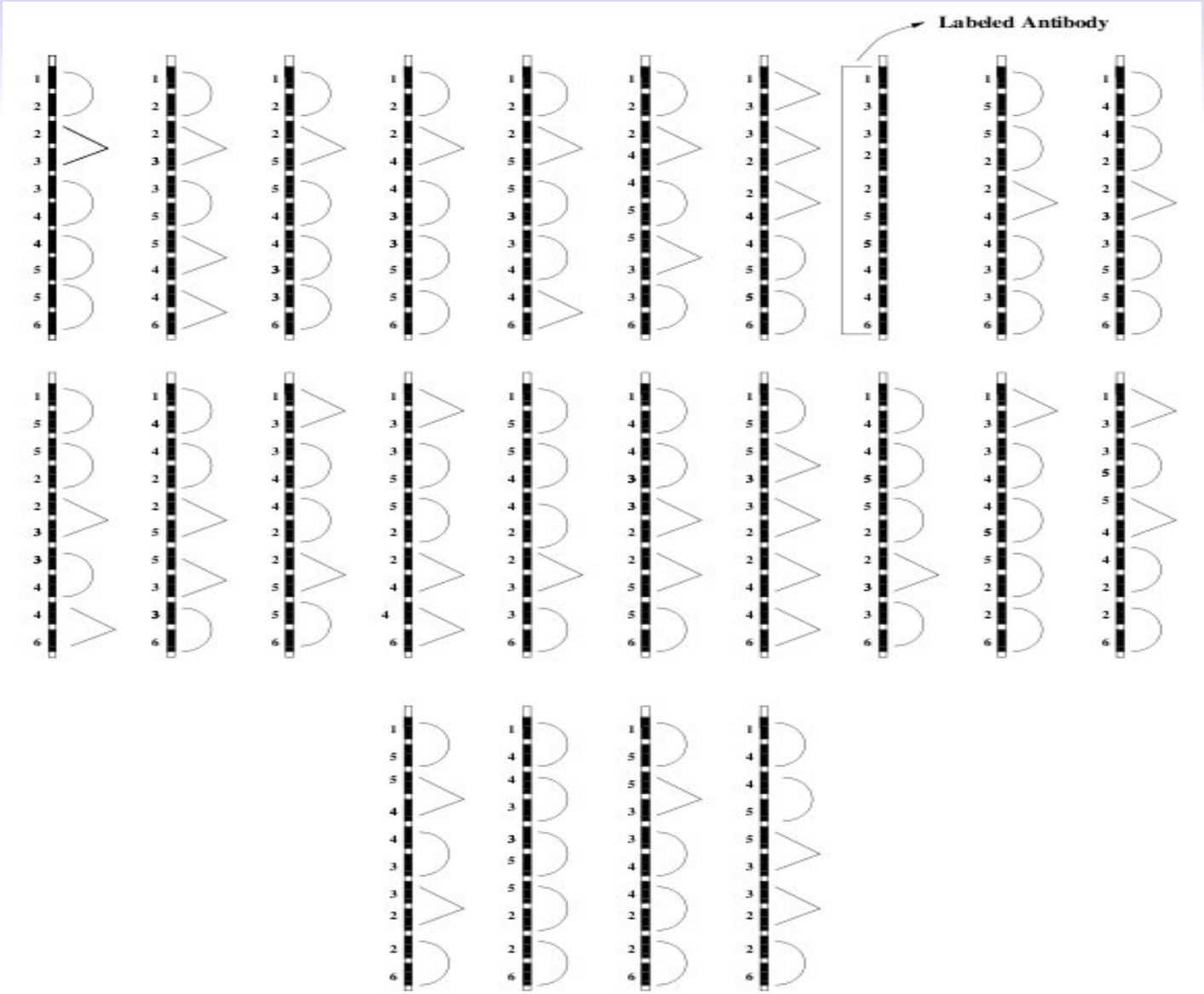
Graph G

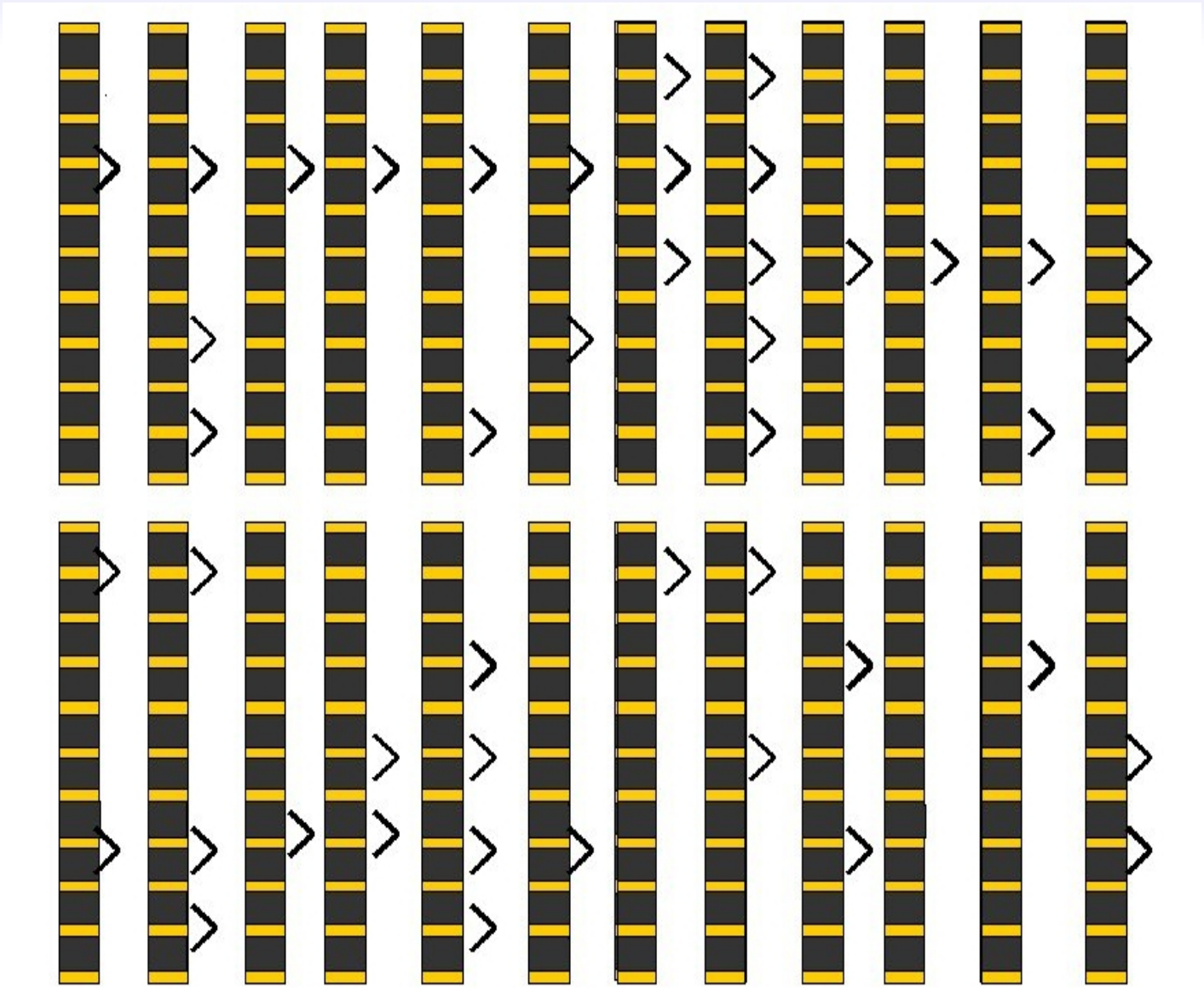


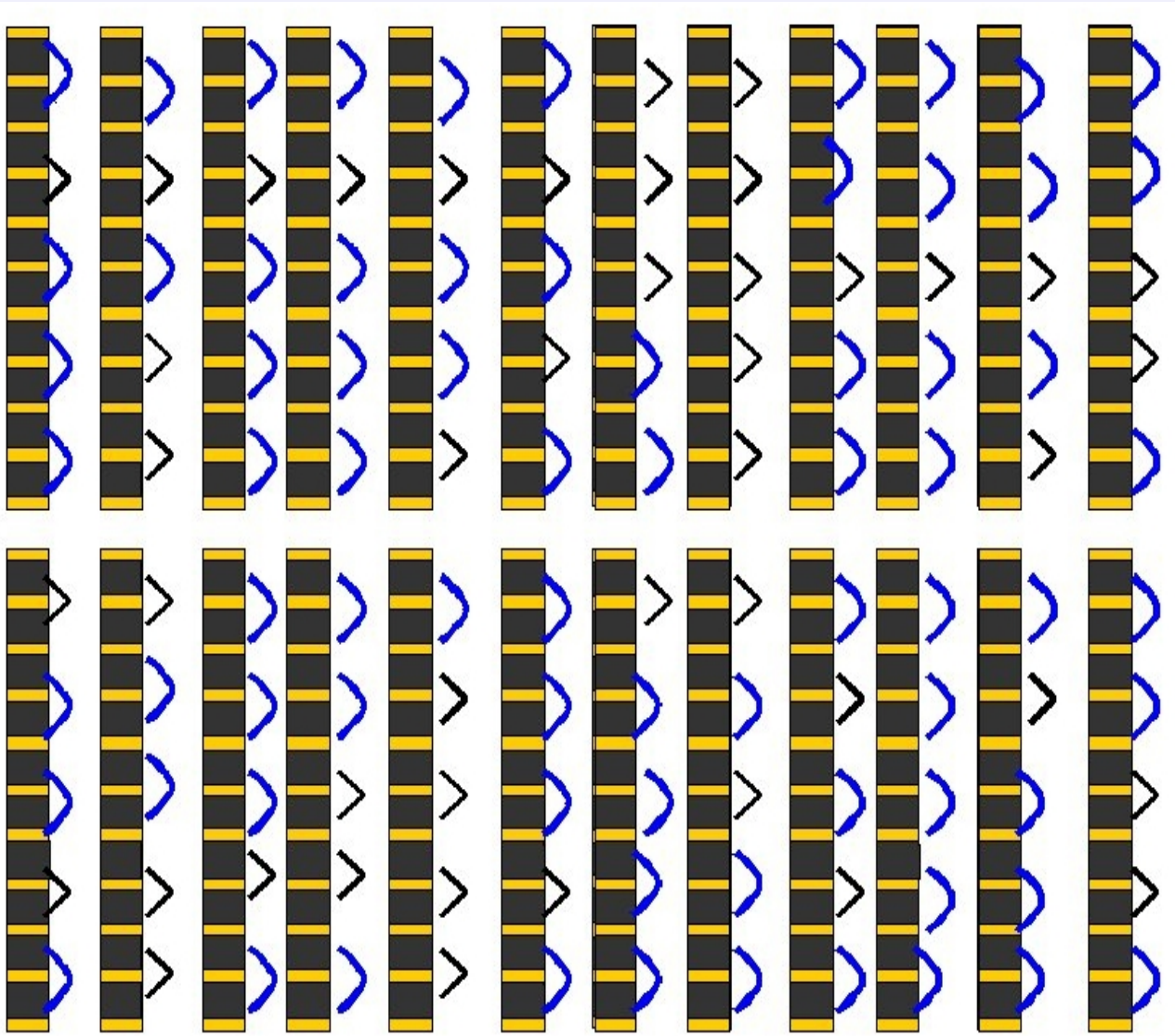
Graph G

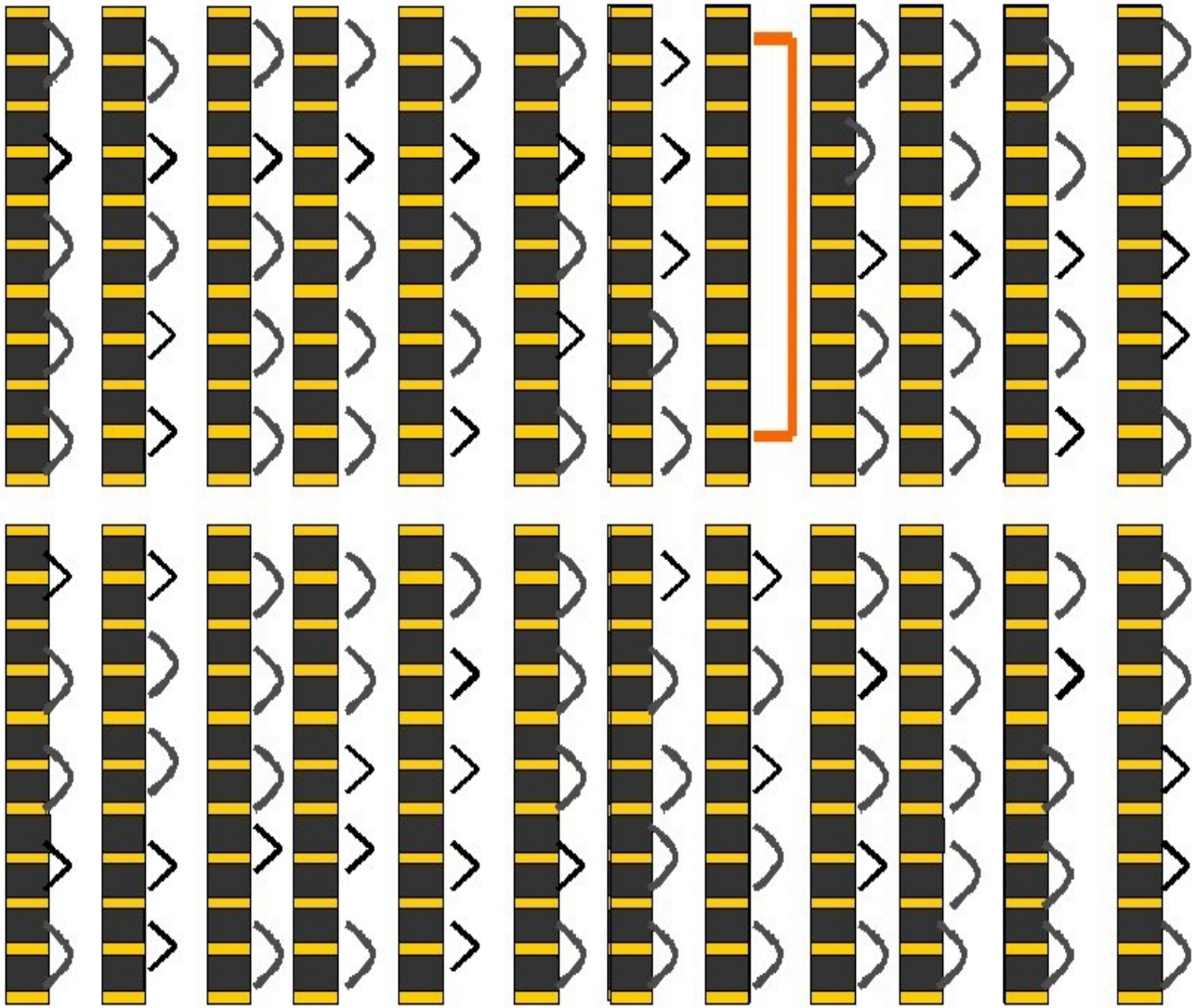


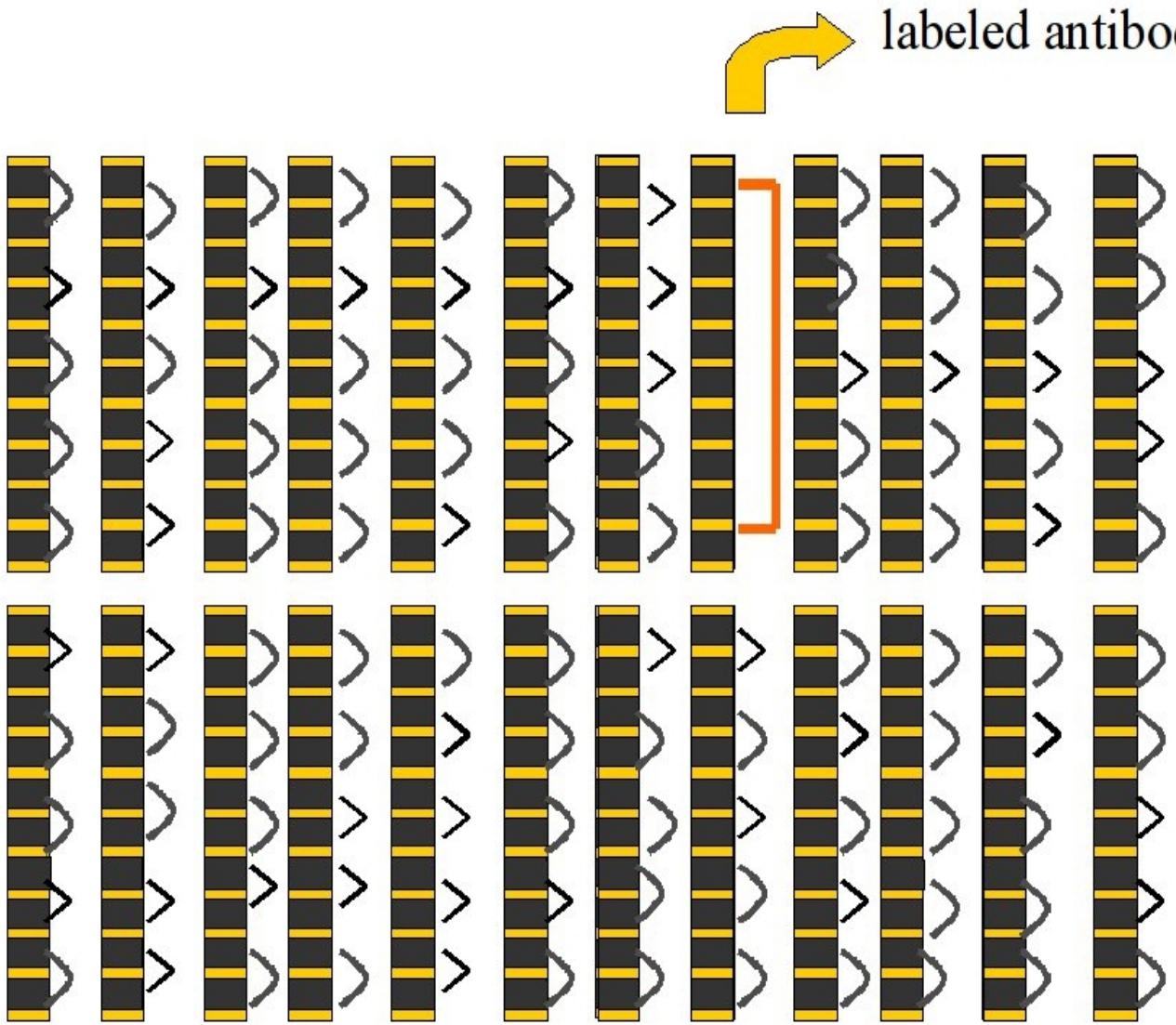



Labeled Antibody








labeled antibody

Complexity

- Number of peptides = $(n-2)!$
- Length of peptides = $O(n)$
- Number of antibodies = $O(n^2)$
- Number of Bio-steps is constant

Outline

1 Introduction
- Computing
- DNA strands
- Peptides and Antibodies

2 DNA Computing
- What is DNA Computing
- First papers on DNA Computing
- Theoretical Models

3 Peptide Computing
- As a Computational Model
- Solving SAT Problem
- Hamiltonian Path Problem
- Theoretical Models

4 Remarks and Open Problems

Infosys

Formal Model for Peptide Computing

- Capabilities and limitations of this computing paradigm
- Computability implies peptide computability. Converse?
- If converse true, under what conditions?

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$;
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$;
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$;
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$;
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Peptide Computer

Quintuple: $\mathcal{P} = (X, E, A, \alpha, \beta)$

- X is a finite alphabet;
- $E \subseteq X^+$ is a language;
- A is a countable alphabet with $A \cap X^* = \emptyset$;
- $\alpha \subseteq E \times A$ is a relation;
- $\beta : E \times A \to \mathbb{R}_+$ is a mapping such that $\beta(e, a) > 0$ if and only if $(e, a) \in \alpha$.

Infosys

Peptide Computer

- A-attachment: partial mapping τ from decomposition of $w \in X^*$ with respect to E to A. $z = w_\tau$.
- If affinity of a is more in z we say it dominates.
- Reaction between words and symbols – if a dominates (i, j) in z then multiset $R(z, a)$ is formed and $\tau \rightarrow \tau'$.
- Reaction between words – if a in z' dominates some position in z.

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

Infosys

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

Infosys

About reactions

- Reactions occur when instability occurs:
  - a dominates (i, j) in z.
  - a in z' dominates (i, j) in z.
- One basic reaction can trigger a sequence of reactions.

Infosys

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Definitions

- *Peptide configuration* is a finite multiset of words in $(X \cup \alpha)^+ \cup A$.
- Peptide configuration P is said to be *stable* if $R(P) = \{P\}$.
- *Peptide instruction* has the form $+P$ or $-P$ where P is a peptide configuration.
- *Peptide program* is the one which controls the instruction set and the halting function.
- *Peptide computation* is a sequence of transition of stable configurations from $c_0, c_1 \cdots c_i$ (with respect to the peptide program) where $\chi(c_i) = 1$ for the first time.
- A function f is peptide computable if we proper encoding and decoding together with a peptide program to carry out the computation.

Sufficiency Condition

Peptide computer is simulated by a Turing machine if

1. E and A are (at least) computably enumerable
2. α is decidable
3. β and χ are computable

Other Models

Automaton Models inspired by Peptide Computing

- Binding-Blocking Automata
- String Binding-Blocking Automata
- Rewriting Binding-Blocking Automata

Modelling gate operations and Boolean circuits

Model to do binary addition in parallel

Infosys

Other Models

Automaton Models inspired by Peptide Computing

- Binding-Blocking Automata
- String Binding-Blocking Automata
- Rewriting Binding-Blocking Automata

Modelling gate operations and Boolean circuits

Model to do binary addition in parallel

Infosys

Other Models

Automaton Models inspired by Peptide Computing

- Binding-Blocking Automata
- String Binding-Blocking Automata
- Rewriting Binding-Blocking Automata

Modelling gate operations and Boolean circuits

Model to do binary addition in parallel

Infosys

Solving Hard Problems

Problems

- Existing methods encode all possible solutions, hence number of molecules is exponential [Hartmanis 1995]
- Preprocessing steps

Solving Hard Problems

Possible Solutions

- Incremental (building) elimination of peptide sequence; compare with [M. Arita et al 1997] in DNA Computing.
  - Reduces number of peptide sequences
  - Preprocessing steps are avoided but included as a part of algorithm
  - Number of steps increases in the order of input size
  - What kind of problems can be solved in this way?
  - Will be an interesting problem to look into in Molecular computing as a whole

Infosys

Do we have to change the computing paradigm for molecular computing to a series of partial encodings and processing before decoding?

Future Problems

- Try solving a small instance of the Hamiltonian path problem in wetlab
- Extend that to basic operations
- Study how the theoretical model works out in wetlab
- Defining basic operations for peptide computing: using ideas from aqueous computing [Tom Head et al 2002]
- Conditions for a peptide computer where each step ends within a finite number of iterations
- Dynamic non-determinism in peptide computer

Infosys

Future Problems

- Try solving a small instance of the Hamiltonian path problem in wetlab
- Extend that to basic operations
- Study how the theoretical model works out in wetlab
- Defining basic operations for peptide computing: using ideas from aqueous computing [Tom Head et al 2002]
- Conditions for a peptide computer where each step ends within a finite number of iterations
- Dynamic non-determinism in peptide computer

Infosys

Future Problems

- Try solving a small instance of the Hamiltonian path problem in wetlab
- Extend that to basic operations
- Study how the theoretical model works out in wetlab
- Defining basic operations for peptide computing: using ideas from aqueous computing [Tom Head et al 2002]
- Conditions for a peptide computer where each step ends within a finite number of iterations
- Dynamic non-determinism in peptide computer

Infosys

Future Problems

- Try solving a small instance of the Hamiltonian path problem in wetlab
- Extend that to basic operations
- Study how the theoretical model works out in wetlab
- Defining basic operations for peptide computing: using ideas from aqueous computing [Tom Head et al 2002]
- Conditions for a peptide computer where each step ends within a finite number of iterations
- Dynamic non-determinism in peptide computer

Infosys

Future Problems

- Try solving a small instance of the Hamiltonian path problem in wetlab
- Extend that to basic operations
- Study how the theoretical model works out in wetlab
- Defining basic operations for peptide computing: using ideas from aqueous computing [Tom Head et al 2002]
- Conditions for a peptide computer where each step ends within a finite number of iterations
- Dynamic non-determinism in peptide computer

Infosys

Future Problems

- Try solving a small instance of the Hamiltonian path problem in wetlab
- Extend that to basic operations
- Study how the theoretical model works out in wetlab
- Defining basic operations for peptide computing: using ideas from aqueous computing [Tom Head et al 2002]
- Conditions for a peptide computer where each step ends within a finite number of iterations
- Dynamic non-determinism in peptide computer

Infosys

Future Problems

On Immunity-based Systems[Ishida 2004] and Peptide Computing

Immunity-based System	Peptide Computing
Information processing	Computational model
Self-nonself distinguishing	Self-nonself model
Survival is the main concern	Computing is the main concern

Study about common area between these two fields