Part 1: DNA Computing
What is DNA Computing?

DNA Computing: Using DNA as hardware of computer due to its molecular-scale storage capabilities.
Barriers to DNA Computing

**DNA Computing:** Using DNA as hardware of computer due to its molecular-scale storage capabilities.

**STOP:** What practical barriers do you see for using DNA as a system of storage?
Barriers to DNA Computing

**DNA Computing:** Using DNA as hardware of computer due to its molecular-scale storage capabilities.

**STOP:** What practical barriers do you see for using DNA as a system of storage?

**Answer:** Reading DNA is expensive, and (until recently) editing DNA has been impossible.
Some DNA Manipulations are Easy

**DNA Computing:** Using DNA as hardware of computer due to its molecular-scale storage capabilities.

However, there are some things that aren’t hard:

- Synthesizing a strand (**oligonucleotide**) of DNA.
Some DNA Manipulations are Easy

**DNA Computing:** Using DNA as hardware of computer due to its molecular-scale storage capabilities.

However, there are some things that aren’t hard:

- Synthesizing a strand (oligonucleotide) of DNA.
- Forcing a DNA strand to base pair given free nucleotides and DNA polymerase.
Some DNA Manipulations are Easy

**DNA Computing:** Using DNA as hardware of computer due to its molecular-scale storage capabilities.

However, there are some things that aren’t hard:

- Synthesizing a strand (**oligonucleotide**) of DNA.
- Forcing a DNA strand to base pair given free nucleotides and DNA polymerase.
- Filtering all fragments of DNA in a sample of some (approximate) length.
Some DNA Manipulations are Easy

**DNA Computing:** Using DNA as hardware of computer due to its molecular-scale storage capabilities.

However, there are some things that aren’t hard:

- Synthesizing a strand (oligonucleotide) of DNA.
- Forcing a DNA strand to base pair given free nucleotides and DNA polymerase.
- Filtering all fragments of DNA in a sample of some (approximate) length.
- Amplifying a strand of DNA with given start/end into many copies (PCR, Nobel Prize in 1993).
Recall: Easy and Difficult Problems

**Hamiltonian Cycle Problem**

*Input:* a directed network with $n$ nodes.

*Output:* “Yes” if there is a cycle visiting every *node* in the network; “No” otherwise.

**Eulerian Cycle Problem**

*Input:* a directed network with $n$ nodes.

*Output:* “Yes” if there is a cycle visiting every *edge* in the network; “No” otherwise.
Recall: Easy and Difficult Problems

Hamiltonian Cycle Problem

Input: a directed network with $n$ nodes.
Output: “Yes” if there is a cycle visiting every node in the network; “No” otherwise.

In particular, all NP-Complete problems are equivalent; if we solve the Hamiltonian Cycle Problem, then we solve them all.
Programming a DNA computer to solve the Hamiltonian cycle problem

Molecular computation of solutions to combinatorial problems
LM Adleman - Science, 1994 - science.sciencemag.org
The tools of molecular biology were used to solve an instance of the directed Hamiltonian path problem. A small graph was encoded in molecules of DNA, and the "operations" of the computation were performed with standard protocols and enzymes. This experiment demonstrates the feasibility of carrying out computations at the molecular level.

Key insight: Rather than trying to use DNA as storage, why not use it to solve difficult problems?

Note: Adleman is also famous for public key cryptography (he is the “A” in “RSA” cryptosystem).
Programming a DNA computer to solve the Hamiltonian cycle problem

Molecular computation of solutions to combinatorial problems
LM Adleman - Science, 1994 - science.sciencemag.org
The tools of molecular biology were used to solve an instance of the directed Hamiltonian path problem. A small graph was encoded in molecules of DNA, and the "operations" of the computation were performed with standard protocols and enzymes. This experiment demonstrates the feasibility of carrying out computations at the molecular level.

Key insight: Rather than trying to use DNA as storage, why not use it to solve difficult problems?

The difficulty here is that it’s not clear at all what it means to “program” a DNA computer.
Adleman’s Algorithm

Algorithm for Determining if there is Hamiltonian Path in Graph $G$ Connecting $v_1$ to $v_n$

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.

© 2022 Phillip Compeau
Adleman’s Algorithm

Algorithm for Determining if there is Hamiltonian Path in Graph $G$ Connecting $v_1$ to $v_n$

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.
2. Keep only those paths that begin with $v_1$ and end with $v_n$. 
Adleman’s Algorithm

Algorithm for Determining if there is Hamiltonian Path in Graph $G$ Connecting $v_1$ to $v_n$

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.
2. Keep only those paths that begin with $v_1$ and end with $v_n$.
3. Keep only those paths that have $n$ nodes.

© 2022 Phillip Compeau
Adleman’s Algorithm

Algorithm for Determining if there is Hamiltonian Path in Graph $G$ Connecting $v_1$ to $v_n$

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.
2. Keep only those paths that begin with $v_1$ and end with $v_n$.
3. Keep only those paths that have $n$ nodes.
4. Keep only those paths that enter all the nodes of the graph at least once.
Adleman’s Algorithm

Algorithm for Determining if there is Hamiltonian Path in Graph $G$ Connecting $v_1$ to $v_n$

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.
2. Keep only those paths that begin with $v_1$ and end with $v_n$.
3. Keep only those paths that have $n$ nodes.
4. Keep only those paths that enter all the nodes of the graph at least once.
5. If any paths remain, return “Yes”; otherwise, return “No”.

© 2022 Phillip Compeau
1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.
1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.

Associate every node $i$ of $G$ with a DNA $k$-mer denoted $O_i$. Call its complement $O'_i$. 

\[
\begin{align*}
O_i &\quad TGACGC \\
O'_i &\quad ACTGCG
\end{align*}
\]
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.

Associate every edge $(i, j)$ with a DNA $k$-mer $E_{i,j}$ consisting of last $k/2$ symbols of $O_i$ followed by first $k/2$ symbols of $O_j$. (Preserves edge orientation.)

\[ O_i \quad TGACGC \]
\[ O_j \quad AAGACT \]
\[ E_{i,j} \quad CGCAAG \]
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.

Note: If $i = 1$, use all $k$ symbols of $O_1$. If $j = n$, use all $k$ symbols of $O_n$. 
Converting Each Step to Experiment

1. Generate many “random paths” through G to ensure that any Hamiltonian path is present.

**Key Point:** recall that generating oligonucleotides is cheap and easy.
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.

Produce many oligonucleotides:
- copies of $E_{i,j}$ for every edge $(i, j)$
- copies of $O'_i$ for every node other than $v_1$ and $v_n$. 
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.

Produce many oligonucleotides:
- copies of $E_{i,j}$ for every edge $(i, j)$
- copies of $O'_i$ for every node other than $v_1$ and $v_n$.

STOP: What will happen when we combine all these DNA oligonucleotides in the lab?
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is present.

Produce many oligonucleotides:
- copies of $E_{i,j}$ for every edge $(i, j)$
- copies of $O'_i$ for every node other than $v_1$ and $v_n$.

**Answer:** edge $E_{i,j}$ will hybridize to $O'_i$ and $O'_j$. Adjacent edges will therefore join into a path.
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.

Hypothetical path in $G$

1 $\rightarrow$ 4 $\rightarrow$ 2 $\rightarrow$ 7

$E_{1,4}$

CATTATAAAG

TTCTGA

$O_4'$

$O_1$ CATTAT

$O_2$ CGTCCCA

$O_4$ AAGACT

$O_7$ CTTTAG
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.

Hypothetical path in $G$

1 $\rightarrow$ 4 $\rightarrow$ 2 $\rightarrow$ 7

$E_{1,4}$ $\quad$ $E_{4,2}$

CATTATAAGACTCGT

TTCTGA

$O'_4$

$O_1$ CATTAT

$O_2$ CGTCCA

$O_4$ AAGACT

$O_7$ CTTTAG

© 2022 Phillip Compeau
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.

Hypothetical path in $G$

1 ➔ 4 ➔ 2 ➔ 7

$E_{1,4}$, $E_{4,2}$

CATTATAAGACTCGT

TTCTGAGCAGGTT

$O_1$ CATTAT

$O_2$ CGTCCA

$O_4$ AAGACT

$O_7$ CTTTAG

© 2022 Phillip Compeau
Converting Each Step to Experiment

1. Generate many “random paths” through G to ensure that any Hamiltonian path is captured.

Hypothetical path in G

1 → 4 → 2 → 7

E_{1,4}  E_{4,2}  E_{2,7}

CATTATAAGACTCGTCCACTTTTAG

TTCTGAGCAGGT

O_1  CATTAT

O_2  CGTCCA

O_4  AAGACT

O_7  CTTTAG

O'_{4}  O'_{2}
Converting Each Step to Experiment

1. Generate many “random paths” through $G$ to ensure that any Hamiltonian path is captured.

As a result, every path in the graph will be present as some double-stranded DNA molecule.
2. Keep only those paths that begin with $v_1$ and end with $v_n$.

Use PCR to amplify only those remaining fragments of DNA that begin with $O_1$ and end with $O_n$. 

© 2022 Phillip Compeau
Converting Each Step to Experiment

3. Keep only those paths that have $n$ nodes.

Filter remaining DNA fragments by length, and throw out all fragments that don’t have length approximately equal to $n \times k$ nucleotides.
4. Keep only those paths that enter all the nodes of the graph at least once.

Convert all DNA to single strands, and hybridize DNA against $O'_i$ for some $i$. Filter out strands that don’t bind. Repeat for all $O'_i$.

```
CATTATAAGAAGCGTCCACTTTTAG

O_1   CATTAT
O_2   CGTCCA
O_3   GACCGT
```
4. Keep only those paths that enter all the nodes of the graph at least once.

Convert all DNA to single strands, and hybridize DNA against $O'_i$ for some $i$. Filter out strands that don’t bind. Repeat for all $O'_i$.

<table>
<thead>
<tr>
<th>CATTATAAGAAGCGTCCACTTTTAG</th>
<th>$O_1$</th>
<th>CATTAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTAATA</td>
<td>$O_2$</td>
<td>CGTCCA</td>
</tr>
<tr>
<td>$O'_1$</td>
<td>$O_3$</td>
<td>GACCGT</td>
</tr>
</tbody>
</table>
4. Keep only those paths that enter all the nodes of the graph at least once.

Convert all DNA to single strands, and hybridize DNA against $O'_{i}$ for some $i$. Filter out strands that don’t bind. Repeat for all $O'_{i}$.

\[
\text{CATTATAAGAAGCGTCCACTTTTAG} \quad O_{1} \quad \text{CATTAT}
\]

\[
\text{GCAGGTT} \quad O_{2} \quad \text{CGTCCCA}
\]

\[
O'_{2} \quad O_{3} \quad \text{GACCGGT}
\]
Converting Each Step to Experiment

4. Keep only those paths that enter all the nodes of the graph at least once.

Convert all DNA to single strands, and hybridize DNA against $O'_i$ for some $i$. Filter out strands that don’t bind. Repeat for all $O'_i$.

```
CATTATAAGAAGCGTCCACTTTTAG
```

```
O_1  CATTAT
O_2  CGTCCA
O_3  GACCGT
```

$O'_3 = CTGGCA$ doesn’t align!
Converting Each Step to Experiment

5. If any paths remain, return “Yes”; otherwise, return “No”.

If any DNA remains from our experiment, then we know that the answer must be “Yes”! Otherwise, it is “No”.

© 2022 Phillip Compeau
We’ve Solved an $NP$-Complete Problem?!

STOP: What issues do you see with this approach?
We’ve Solved an NP-Complete Problem?! 

**STOP:** What issues do you see with this approach? 

**Answer:** Three immediate barriers: 

1. Possibility of errors is high. 
2. We still need to generate, at a minimum, $n!$ strands of DNA. So this is impossible for networks with, say, 100 nodes. 
3. An enormous amount of lab work needs to be done, with hours of waiting times.
DNA is nevertheless promising as hard drive storage

**DNA Fountain enables a robust and efficient storage architecture**

*Y Erlich, D Zielinski* - Science, 2017 - science.sciencemag.org

DNA is an attractive medium to store digital information. Here we report a storage strategy, called DNA Fountain, that is highly robust and approaches the information capacity per nucleotide. Using our approach, we stored a full computer operating system, movie, and ...

🌟 Cited by 342  Related articles  All 14 versions

---

November 12, 2020 03:15 PM Eastern Standard Time

SOUTH SAN FRANCISCO, Calif.--(BUSINESS WIRE)--Twist Bioscience Corporation (NASDAQ: TWST), Illumina, Inc. (NASDAQ: ILMN) and Western Digital (NASDAQ: WDC) today announced the formation of an alliance with Microsoft to advance the field of DNA data storage. These founding companies, alongside member organizations, will work together to create a comprehensive industry roadmap that will help the industry achieve interoperability between solutions and help establish the foundations for a cost-effective commercial archival storage ecosystem for the explosive growth of digital data.
Part 2: Self-Replicating Cellular Automata
Can a Machine Replicate Itself?
Why Haven’t We Seen Alien Spacecraft?

Fermi paradox: No evidence of alien life has been found in the galaxy despite its likelihood.
Why Haven’t We Seen Alien Spacecraft?

von Neumann Probes: a theorized space probe that can use resources it finds to self-replicate.
Von Neumann’s Question

What is the simplest possible self-replicating system?

John von Neumann

© 2022 Phillip Compeau
Von Neumann’s Question

What is the simplest possible self-replicating system?

Learn some biology, John!

Stanislaw Ulam

John von Neumann

© 2022 Phillip Compeau
Cells are Self-Replicators

We are the Von Neumann machines


© 2022 Phillip Compeau
Psst ... There’s a Simpler Self-Replicator
Cellular Automata

**Cellular Automaton:** A grid of (typically square) cells, along with a collection of simple rules that allow the cells to change from one “state” to another.
Cellular Automata

**Cellular Automaton**: A grid of (typically square) cells, along with a collection of simple rules that allow the cells to change from one “state” to another.

A lot of the cellular automata you could come up with are pretty boring, but then there is …
The Game of Life: Rules

A: If a cell is alive and has either two or three live neighbors, then it remains alive.
B: If a cell is alive and has zero or one live neighbors, then it dies out.
C: If a cell is alive and has four or more live neighbors, then it dies out.
D: If a cell is dead and has more than or fewer than three live neighbors, then it remains dead.
E: If a cell is dead and has exactly three live neighbors, then it becomes alive.
What’s the Next Generation?

A: If a cell is alive and has either two or three live neighbors, then it remains alive.
B: If a cell is alive and has zero or one live neighbors, then it dies out.
C: If a cell is alive and has four or more live neighbors, then it dies out.
D: If a cell is dead and has more than or fewer than three live neighbors, then it remains dead.
E: If a cell is dead and has exactly three live neighbors, then it becomes alive.

Dark = alive
What’s the Next Generation?

A: If a cell is alive and has either two or three live neighbors, then it remains alive.
B: If a cell is alive and has zero or one live neighbors, then it dies out.
C: If a cell is alive and has four or more live neighbors, then it dies out.
D: If a cell is dead and has more than or fewer than three live neighbors, then it remains dead.
E: If a cell is dead and has exactly three live neighbors, then it becomes alive.

© 2022 Phillip Compeau
What’s the Next Generation?

<table>
<thead>
<tr>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>B</td>
<td>E</td>
<td>E</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>E</td>
<td>E</td>
<td>B</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
**Exercise:** What is the next generation?

- **A:** If a cell is alive and has either two or three live neighbors, then it remains alive.
- **B:** If a cell is alive and has zero or one live neighbors, then it dies out.
- **C:** If a cell is alive and has four or more live neighbors, then it dies out.
- **D:** If a cell is dead and has more than or fewer than three live neighbors, then it remains dead.
- **E:** If a cell is dead and has exactly three live neighbors, then it becomes alive.
Quick Quiz

**Exercise:** Carry out the next few generations of this board. What happens?
Oscillators
Oscillators

© 2022 Phillip Compeau
Oscillators

© 2022 Phillip Compeau
Oscillators
Oscillators
Oscillators

© 2022 Phillip Compeau
Getting More Complicated …
The Curious Case of the R-Pentomino
The Curious Case of the R-Pentomino
Is it possible for the number of live cells to grow without bound as time goes on?

Courtesy: Thane Plambeck

© 2022 Phillip Compeau
Bill Gosper’s “Glider Gun”

Definitely!
Bill Gosper’s “Glider Gun”
Self-Replicating Automata: A History

John von Neumann
- Year: 1952
- Number of States: 29
- Size of Self Replicator: 130,622 cells

Edgar Codd
- Year: 1968
- Number of States: 8
- Size of Self Replicator: 283,126,588 cells

Chris Langton
- Year: 1984
- Number of States: 8
- Size of Self Replicator: 86 cells

© 2022 Phillip Compeau
Langton Loops: A Self-Replicating Cellular Automaton
Part 3: Spatial Games
Cooperation is Everywhere. But Why?

Courtesy: milkwood.net
“Prisoner’s Dilemma”

**Prisoner’s Dilemma:** A simple two-player game with choices between “cooperation” and “defection” against an opponent.

<table>
<thead>
<tr>
<th>Your decision</th>
<th>Partner’s decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>Cooperate</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Defect</td>
<td>Cooperate</td>
</tr>
<tr>
<td></td>
<td>$b &gt; 1$</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Defect</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Prisoner’s Dilemma: A simple two-player game with choices between “cooperation” and “defection” against an opponent.

Partner’s decision

<table>
<thead>
<tr>
<th></th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Defect</td>
<td>$b &gt; 1$</td>
<td>0</td>
</tr>
</tbody>
</table>

STOP: Why would you cooperate?
Axelrod’s Tournament (1978): What if the game is played *multiple times*?

**Group Exercise:** Design a strategy.

- We are playing an (unknown) \( m \) number of games.
- Opponent’s strategy is hidden.
- We play a variety of opponents but use same strategy.

<table>
<thead>
<tr>
<th>Partner’s decision</th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Defect</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Your decision</th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
Idea 0: “Poor-Trustling Fool”

for every integer \( i \) between 1 and \( m \) cooperate!

STOP: What are its strengths and weaknesses?
Idea 1: “All-Defect”

for every integer $i$ between 1 and $m$ defect!

STOP: What are its strengths and weaknesses?
Idea 2: “Grudger”

```
betrayed ← false
for every integer i between 1 and m
  if betrayed
    defect!
  else
    if opponent defected in game i - 1
      betrayed = true
      defect!
    else
      cooperate!
```
The Clear Winner: “Tit-for-Tat”

cooperate in game 1
for every integer $i$ between 2 and $m$
do whatever opponent did in game $i - 1$

This was the winning strategy (and the simplest!) among 15 submissions.

STOP: What properties of this strategy make it so good?
More tournaments have shown that the highest-scoring strategies tend to have four qualities:

- **Niceness**: Never be the first to defect.
- **Provocability**: Get mad quickly at defectors and retaliate.
- **Forgiveness**: Do not hold a grudge once you have vented your anger.
- **Clarity**: Act in ways that are straightforward for others to understand.

http://www2.econ.iastate.edu/classes/econ308/tesfatsion/axeltmts.pdf
**Spatial Games**: Every cell in a 2-D field plays a simplified Prisoner’s Dilemma with each of its neighbors.

<table>
<thead>
<tr>
<th>Partner’s decision</th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Defect</td>
<td>$b &gt; 1$</td>
<td>0</td>
</tr>
</tbody>
</table>

© 2022 Phillip Compeau
Spatial Games: Every cell in a 2-D field plays a simplified Prisoner’s Dilemma with each of its neighbors.

All use same strategy: in “generation” $i$, each cell adopts the strategy of its “best-scoring” Moore neighbor in generation $i - 1$. 
Idea: Let’s animate the spatial game board over the generations.

- Cells choosing to cooperate with their neighbors = blue
- Cells choosing to defect = red

© 2022 Phillip Compeau
Spatial Game Theory: Playing Games on a 2-D Automaton

**Idea:** Let’s animate the spatial game board over the generations.

- Cells choosing to **cooperate** with their neighbors = blue
- Cells choosing to **defect** = red

Let’s put one defector in the middle of our board. I wonder what we will see?
Spatial Game Theory: Playing Games on a 2-D Automaton

**Idea:** Let’s animate the spatial game board over the generations.
- Cells choosing to cooperate with their neighbors = blue
- Cells choosing to defect = red

Let’s put one defector in the middle of our board. I wonder what we will see?

**Note:** Our early reasoning would imply that the defectors would take over completely...
Spatial Games with $b = 1.65$
Nowak, a Harvard mathematical biologist, seems to have been Epstein’s favorite scientist, regularly mentioned in press releases issued by his foundations. The financier donated $6.5 million to launch Nowak’s Program for Evolutionary Dynamics at Harvard in 2003 — although Epstein claimed, apparently falsely, to have given $30 million.
Part 4: Tripping with Turing
Why Do Animals Have Stripes (or Spots)?

- https://i.eybayimg.com/images/g/gy4AAOxylpNTUTCI/s-1300.jpg
- https://en.wikipedia.org/wiki/Mbu_pufferfish#/media/File:Giant_Puffer_fish_skin_pattern.JPG

© 2022 Phillip Compeau
Alan Turing has the answer!

Turing patterns: stripe/spot patterns that occur due to specific reactions and diffusion.

Well, the stripes are easy. But what about the horse part?

The chemical basis of morphogenesis

AM Turing - Bulletin of mathematical biology, 1990 - Springer

It is suggested that a system of chemical substances, called morphogens, reacting together and diffusing through a tissue, is adequate to account for the main phenomena of morphogenesis. Such a system, although it may originally be quite homogeneous, may later develop a pattern or structure due to an instability of the homogeneous equilibrium, which is triggered off by random disturbances. Such reaction-diffusion systems are considered in some detail in the case of an isolated ring of cells, a mathematically convenient, though …
We’ve already seen reaction-diffusion...

‘Member MCell?!

‘Member modeling network motifs?!

Ooh I ‘member …
The model we will use is similar to a predator-prey simulation.

“Prey” molecules enter the system at a constant feed rate $f$.  

“Predator” molecules are removed at a constant kill rate $k$.  

© 2022 Phillip Compeau
The model we will use is similar to a predator-prey simulation.

“Prey” molecules enter the system at a constant feed rate $f$.

“Predator” molecules are removed at a constant kill rate $k$. 
The model we will use is similar to a predator-prey simulation.

The predators can “eat” the prey and reproduce via

\[ 2 \text{ Predator } + \text{ Prey } \rightarrow 3 \text{ Predator } \]
The model we will use is similar to a predator-prey simulation.

The predators can “eat” the prey and reproduce via:

\[ 2 \text{ Predator} + \text{ Prey} \rightarrow 3 \text{ Predator} \]

But the prey are “faster swimmers”, having a diffusion rate that is twice as fast.
Running our simulation

If the kill rate is too high, then the predators die out more quickly than they can eat the prey, and so only prey will survive.

\[ f = 1,000 \text{ and } k = 500,000 \]
Running our simulation

On the other hand, if \( f \) is too high, then the prey will increase, feeding the predators, and we will see an explosion in the number of predators.

\[ f = 1,000,000 \text{ and } k = 100,000 \]
Running our simulation

Finding a sweet spot set of parameters produces waves of predator “stripes” expanding outward against a background of prey.

\[ f = 100,000 \text{ and } k = 200,000 \]

© 2022 Phillip Compeau
Running our simulation

Holding $k$ fixed and increasing $f$ by a little increases the likelihood of predator-prey interactions, producing even more predator stripes.

$f = 140,000$ and $k = 200,000$
Running our simulation

Increasing $f$ further produces a chaotic stripe pattern because there are so many pockets of predators that they constantly collide and mix.

$f = 175,000$ and $k = 200,000$
Running our simulation

Increasing $f$ further produces a chaotic stripe pattern because there are so many pockets of predators that they constantly collide and mix.

$f = 175,000$ and $k = 200,000$
Running our simulation

Once $f$ equals $k$, the stripes disappear. We might expect to see a uniform mix, but instead, we see a “mottling” of red and green clusters, or spots.

$f = 200,000$ and $k = 200,000$
Turing pattern systems have been identified in fish (but not zebras)!

These pufferfish are very similar genetically, and yet they display different patterns.

Unlike other robust biological systems, Turing patterns are fine-tuned.
An Automaton-Like Model of Reaction-Diffusion

**Gray-Scott Model:** a “discretized” simulation with improved runtime by partitioning space into blocks and assuming concentration of a given molecule in a block is uniform throughout the block.

**Autocatalytic reactions in the isothermal, continuous stirred tank reactor:**
Oscillations and instabilities in the system A+ 2B → 3B; B → C

P Gray, SK Scott - Chemical Engineering Science, 1984 - Elsevier
The prototype, cubic autocatalytic reaction (A+ 2B → 3B) forms the basis for the simplest homogeneous system to display “exotic” behaviour. Even under well-stirred, isothermal, open conditions (CSTR) we may find multistability, hysteresis, extinction, ignition and …

Cited by 639  Related articles  All 4 versions
An Automaton-Like Model of Reaction-Diffusion

Gray-Scott Model: a "discretized" simulation with improved runtime by partitioning space into blocks and assuming concentration of a given molecule in a block is uniform throughout the block.

A = prey; B = predators

https://www.karlsims.com/rd.html
Parameters $d_A$ and $d_B$ indicate what fraction of a cell’s two particle types to diffuse into neighbors.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>0,0</td>
<td>0,0</td>
<td>1,1</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,0</td>
</tr>
</tbody>
</table>

$\hspace{.2cm}d_A = 0.2 \hspace{.2cm}d_B = 0.1$
Implementing Reactions as Cellular Operations

Parameters $d_A$ and $d_B$ indicate what fraction of a cell’s two particle types to diffuse into neighbors.

Then, we apply reactions cell by cell.

$$[A]_{\text{new}} = [A] + f(1 - [A]) - [A][B]^2$$

$$[B]_{\text{new}} = [B] - k[B] + [A][B]^2$$

Here, $f$ is the feed rate parameter, and $k$ is the kill rate parameter; both are between 0 and 1.
A cell’s color is based on its value of $\frac{[B]}{([A] + [B])}$. If this value is close to zero (many prey), then it will be colored red, and if it is close to 1 (many predators), then it will be colored dark blue.

$f = 0.034$ and $k = 0.095$
Running the Gray-Scott Model

Creating multiple initial predator locations leads to more complex patterns.

\[ f = 0.034 \text{ and } k = 0.095 \]
If we hold the feed rate constant and increase $k$ by just 0.002, then the patterns change significantly into spots.

\[ f = 0.034 \text{ and } k = 0.097 \]
If we make the prey just a little bit happier, raising $f$ by 0.004 and $k$ by 0.002, then we get a striped pattern again, but a different one.

$f = 0.038$ and $k = 0.099$
Running the Gray-Scott Model

And if we raise $f$ by another 0.004 and $k$ by another 0.002, we again see a spot pattern.

$f = 0.042$ and $k = 0.101$
Convergent patterns are very parameter dependent.

This plot shows final convergent patterns for varying values of $k$ (x-axis) and $f$ (y-axis).

Key point: Gray-Scott is a faster model that confirms the highly fine-tuned parameters of this system.

Image source: Robert Munafo

© 2022 Phillip Compeau
PRETTY TRIppy HUh
We all trip in similar ways. But why?

Form constant (Klüver, 1928): a commonly recurring shape in visual hallucinations.
We all trip in similar ways. But why?

Key point: Hallucinations happen in the blind and don’t move in visual field, so they originate in brain.
The brain encodes signals from retina

Cowan 1978: determined details for transformation of retinal coordinates (polar) to cortex (rectangular).

Ermentrout and Cowan, 1979
The brain encodes signals from retina

Cowan 1978: determined details for transformation of retinal coordinates (polar) to cortex (rectangular).

All the form constants reduce to “stripes” in the visual cortex!

Ermentrout and Cowan, 1979
How Does This Relate to Hallucinations?

The visual cortex contains "activator" neurons that tend to be connected more tightly and "inhibitor" neurons with fewer, sparser connections.

**Hypothesis:** activators/inhibitors are analogous to "predator"/"prey" molecules; some events (migraines, hallucinogens) change the underlying parameters of the system and produce Turing patterns within the visual cortex.
The End?