UNCONVENTIONAL COMPUTING

Hans W. Guesgen

Massey University, New Zealand

What Is UC?

- Wide range of new and/or unusual methods to perform computations
- Also called alternative computing
- Includes:
  - Optical computing
  - Quantum computing
  - Chemical computing
  - Natural computing
  - Biologically-inspired computing
  - Wetware computing
  - DNA computing
  - Molecular computing
  - Amorphous computing
  - Nanocomputing
  - Analogue computing
  - Domino Computation

Church-Turing Thesis

- Prevailing paradigm in classical computation theory
- States that no realisable computing device can be more powerful than a universal Turing machine (aside from relative speedups)
- Any "reasonable" model of computation can be effectively simulated by a (probabilistic) Turing machine

Supporting Reasons

- **Philosophical argument:**
  Very difficult to imagine some other method which falls outside the scope of Turing's description
- **Mathematical evidence:**
  Every mathematical notion of computability so far was proven equivalent to Turing computability
- **Sociological evidence:**
  All example of classical computing devices produced so far can be simulated by a Turing machine

Beyond Reasonable Doubt?

"… how can we ever exclude the possibility of our being presented, some day (perhaps by some extraterrestrial visitors), with a (perhaps extremely complex) device or "oracle" that "computes" an uncomputable function?"

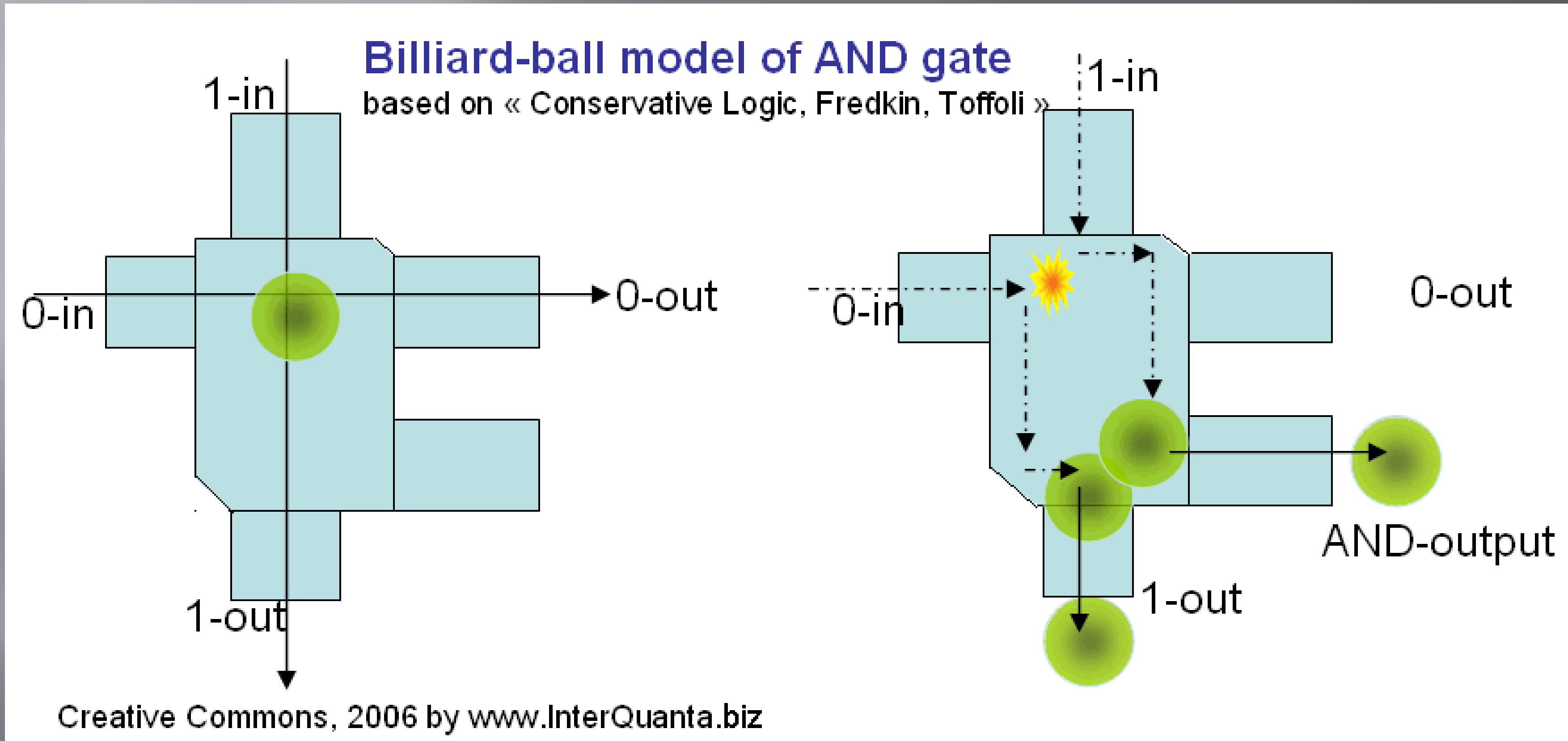
M. Davis
Computability and Unsolvability
McGraw-Hill, New York, 1958

Billiard Ball Computer

- Also known as a conservative logic circuit
- Idealized model of a computing machine based on Newtonian dynamics
- Relies on the motion of spherical billiard balls in a friction-free environment made of buffers against which the balls bounce perfectly
- Purpose:
  - To investigate the relation between computation and reversible processes in physics
  - To provide context to the halting problem and similar results in computability theory

E. Fredkin and T. Toffoli
Conservative Logic
Int. J. Theor. Phys. 21 (1982), 219–253

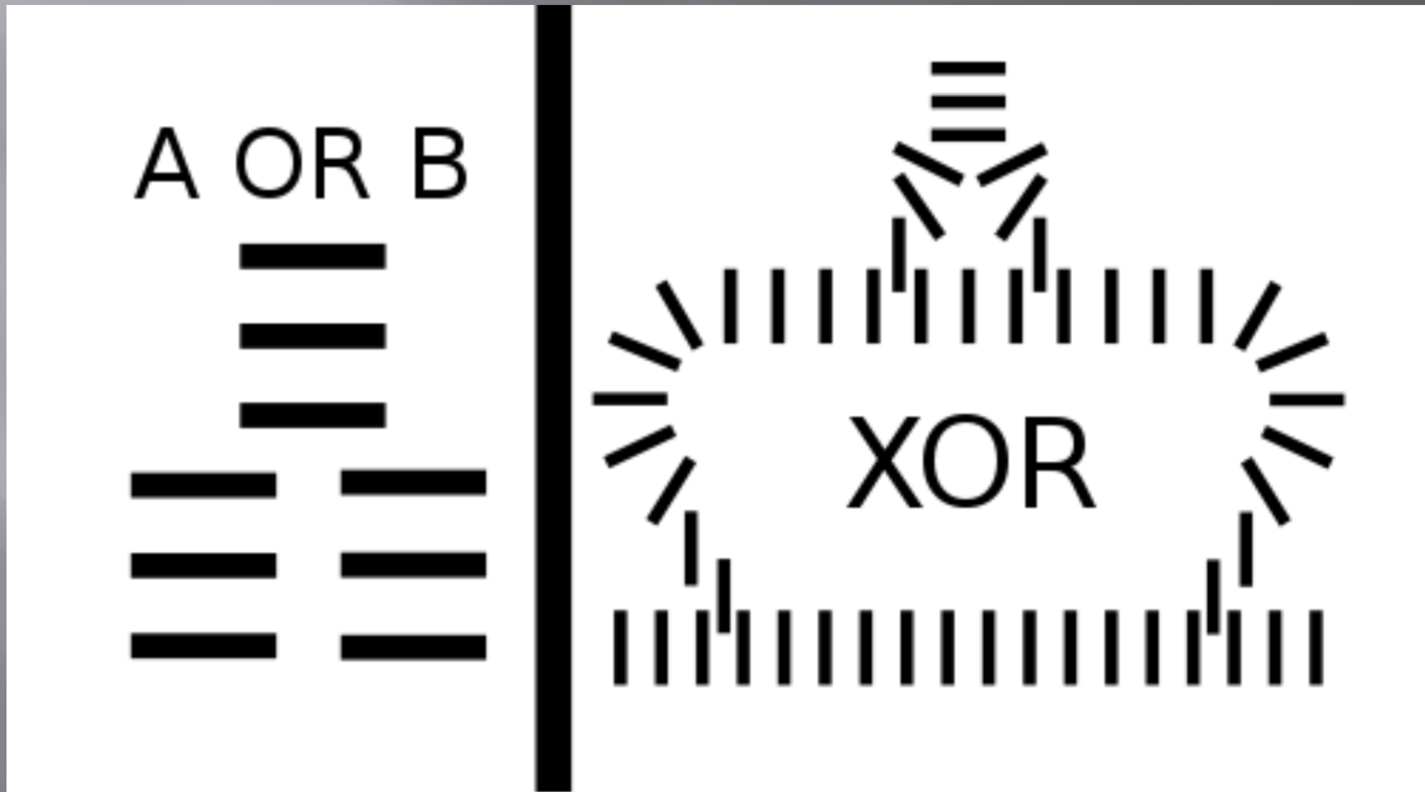
AND Gate


Billiard-ball model of AND gate
based on « Conservative Logic, Fredkin, Toffoli »
1-in
0-in
0-out
1-out
1-in
0-in
0-out
AND-output
1-out
Creative Commons, 2006 by www.InterQuanta.biz

Domino Computer

- Mechanical computer built using dominoes to represent mechanical amplification or logic gating of digital signals
- The two logic gates XOR and OR are the easiest to make with dominoes
- All other gates can be built from these:
  - A XOR 1 = $\bar{A}$
  - (A OR B) XOR (A XOR B) = A AND B
- The logic elements are one-use-only, as a domino cannot be reset

OR and XOR Gates



Natural Computing

- Encompasses three classes of methods:
  - Those that take inspiration from nature for the development of novel problem-solving techniques
  - Those that are based on the use of computers to synthesize natural phenomena
  - Those that employ natural materials (e.g., molecules) to compute

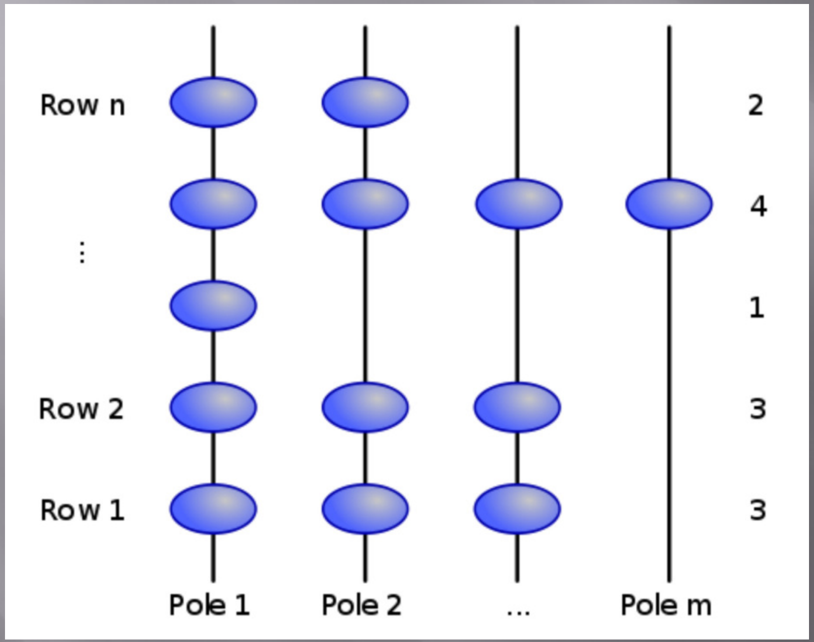
- Computational paradigms studied by natural computing are abstracted from natural phenomena :
  - Self-replication
  - Functioning of the brain
  - Darwinian evolution
  - Cell membranes
  - DNA and RNA

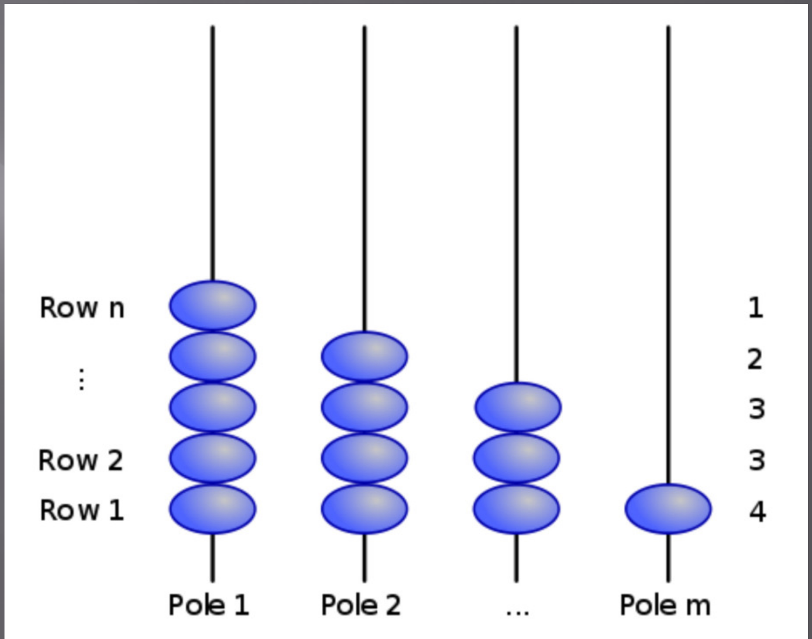
Bead-Sort

- Natural sorting algorithm
- Uses beads to represent and sort positive integers
- Both digital and analog hardware implementations can achieve a sorting time of $O(n)$
- Requires $O(n^2)$ space even in the best case

J.J. Arulanandham, C.S. Calude, M.J. Dinneen
Bead–Sort: A Natural Sorting Algorithm
The Bulletin of the European Association for Theoretical Computer Science 76 (2002), 153-162

Implementation with Rods





Particle Swarms Optimisation

- Designed to simulate social behaviour

  J. Kennedy, R. Eberhart
  Particle Swarm Optimization
  Proceedings of IEEE International
  Conference on Neural Networks, pp. 1942–1948, 1995

- Representation of the movements in bird flocks or fish schools
- Optimises a problem by iteratively trying to improve a candidate solution
- Does not use the gradient of the problem being optimised, which means PSO does not require that the optimisation problem be differentiable

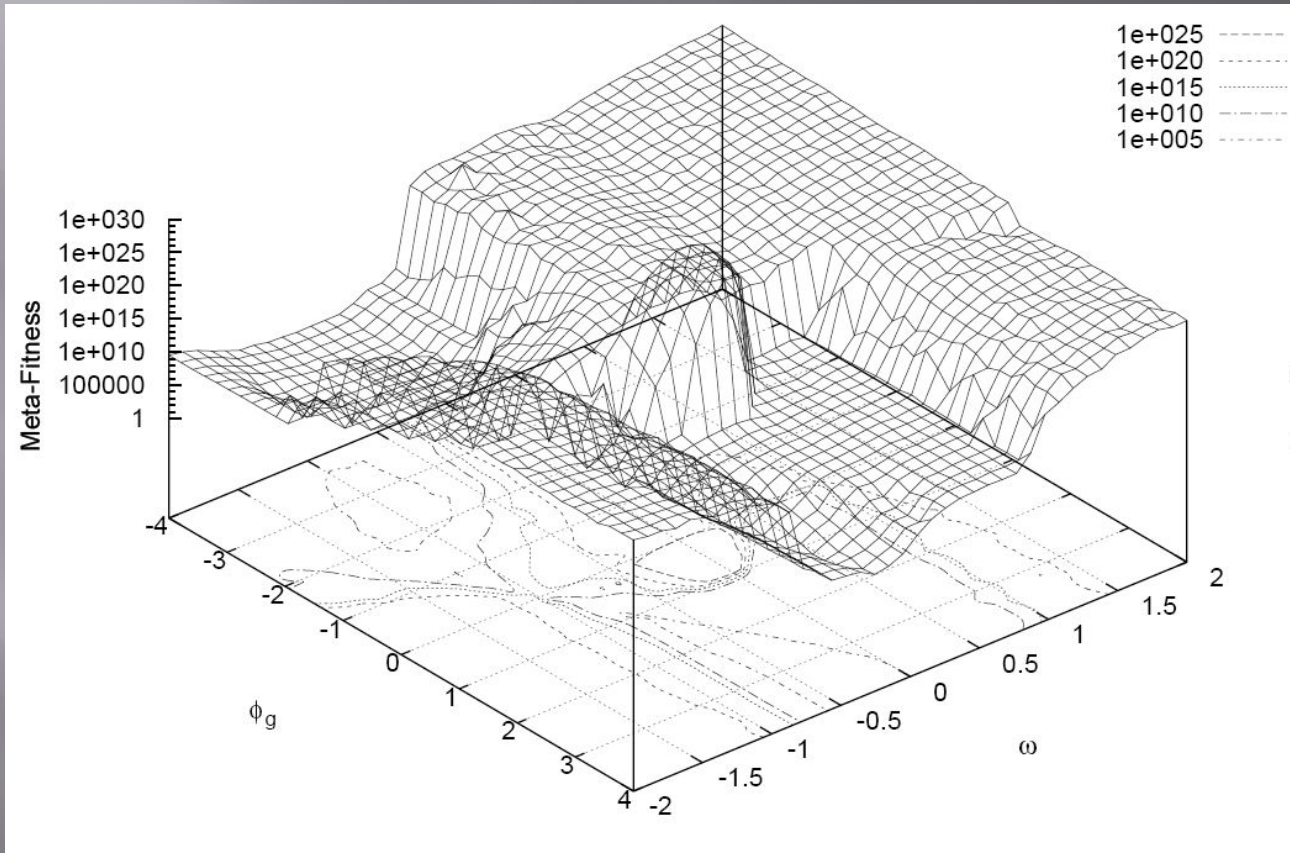
Basic PSO Algorithm

- Let S be the number of particles in the swarm, each having:
  - A position $x_i \in \mathbb{R}^n$ in the search space
  - A velocity $v_i \in \mathbb{R}^n$
- Let p_i be the best known position of particle i
- Let g be the best known position of the entire swarm
- For each particle $i = 1, ..., S$ do:
  - Initialize the particle's position with a uniformly distributed random vector: $x_i \sim U(b_{lo}, b_{up})$, where b_{lo} and b_{up} are the lower and upper boundaries of the search space
  - Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
  - If $(f(p_i) < f(g))$ update the swarm's best known position: $g \leftarrow p_i$
  - Initialize the particle's velocity: $v_i \sim U(-|b_{up}-b_{lo}|, |b_{up}-b_{lo}|)$
- Until a termination criterion is met, repeat:
  - For each particle $i = 1, ..., S$ do:
    - Pick random numbers: $r_p, r_g \sim U(0,1)$
    - Update the particle's velocity: $v_i \leftarrow \omega\, v_i + \varphi_p\, r_p\, (p_i-x_i) + \varphi_g\, r_g\, (g-x_i)$
    - Update the particle's position: $x_i \leftarrow x_i + v_i$
    - If $(f(x_i) < f(p_i))$ do:
      - Update the particle's best known position: $p_i \leftarrow x_i$
      - If $(f(p_i) < f(g))$ update the swarm's best known position: $g \leftarrow p_i$
- Return g

The Magic of PSO

- Swarm behaviour varies between two behaviours:
  - Exploratory behaviour:
    Searching a broader region of the search-space
  - Exploitative behaviour:
    Locally oriented search to get closer to a (possibly local) optimum
- Parameters ω, φ_p, and φ_g must be chosen to properly balance between exploration and exploitation:
  - To avoid premature convergence to a local optimum
  - To ensure a good rate of convergence to the optimum

PSO Meta-Fitness Landscape



M.E.H. Pedersen
Tuning & Simplifying Heuristical Optimization
PhD Thesis, 2010
University of Southampton
Computational Engineering and Design Group

DNA Computing

- Field initially developed by Leonard Adleman of the University of Southern California in 1994
- Fundamentally similar to parallel computing in that it takes advantage of the many different molecules of DNA to try many different possibilities at once

L.M. Adleman
Molecular Computation of Solutions to Combinatorial Problems
Science 266 (1994), 1021–1024

Tools of DNA Computing

- Watson-Crick pairing
  - DNA molecules essentially are chains consisting of
    - Adenine (A)
    - Thymine (T)
    - Guanine (G)
    - Cytosine (C)
  - Every strand of DNA has a Watson-Crick complement
    - A is the complement of T (and vice versa)
    - G is the complement of C (and vice versa)
  - Two complementing strands of DNA can anneal to form a double helix

Tools of DNA Computing

- DNA polymerase
  - Makes a Watson-Crick complementary DNA strand from a DNA template
  - Needs a “start signal” to tell it where to begin making the copy
  - Provided by a piece of DNA that is annealed to the template by Watson-Crick complementarity
- DNA ligase
  - Takes two strands of DNA in proximity and covalently bonds them into a single strand
  - Used in cells to repair breaks in DNA strands

Tools of DNA Computing

- Gel electrophoresis
  - Solution of heterogeneous DNA molecules is placed in one end of a slab of gel and current is applied
  - Negatively charged DNA molecules move toward the anode
  - Shorter strands move more quickly than longer ones
  - It is possible to see bands in the gel where the DNA molecules of various lengths have come to rest
- DNA synthesis
  - Commercial synthesis facilities are able to produce arbitrary DNA strands with great precision

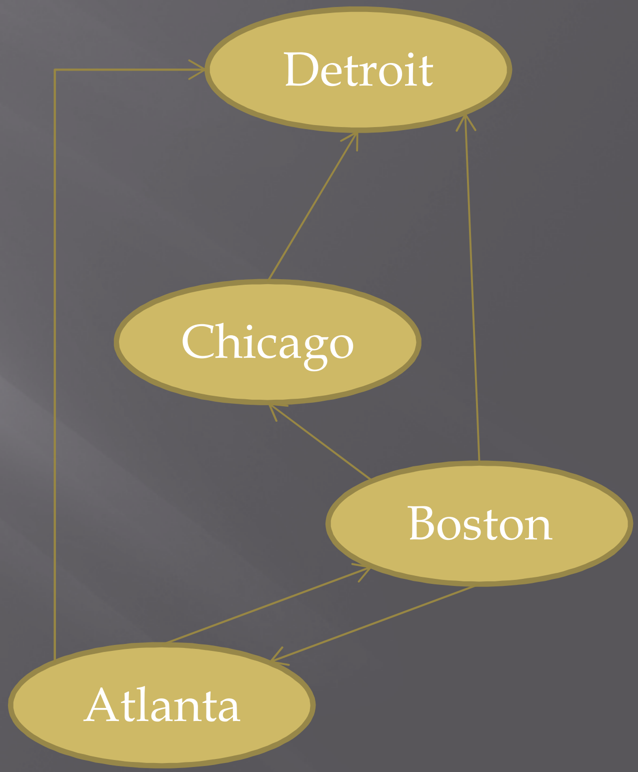
Hamiltonian Path Problem

- Used by Adleman as proof of concept
- Find a path through a directed graph from a start node to an end node that visits each node exactly once
- NP-complete problem, but algorithm exists:
  - Generate set of random paths through graph (with *n* nodes)
  - For each path in the set:
    - Remove path if it does not start at the start node or end with the end node
    - Remove path if it does not pass through exactly *n* nodes
    - Remove path if it does not pass through each node
  - If the resulting set is not empty, then it contains the Hamiltonian paths; otherwise, there is no such path

DNA Encoding

City	DNA name	Complement
Atlanta	ACTTGCAG	TGAACGTC
Boston	TCGGACTG	AGCCTGAC
Chicago	GGCTATGT	CCGATACA
Detroit	CCGAGCAA	GGCTCGTT

Flight	DNA flight number
Atlanta-Boston	GCAGTCGG
Atlanta-Detroit	GCAGCCGA
Boston-Chicago	ACTGGGCT
Boston-Detroit	ACTGCCGA
Boston-Atlanta	ACTGACTT
Chicago-Detroit	ATGTCCGA



Producing a Solution

- Synthesise the complementary DNA city names and flight numbers
- Put them in a test tube with water, ligase, salt, and some other ingredients
- Matching sequences meet by chance and form complexes that encode random paths through the graph
- Some of them are Hamiltonian paths, but most of them are not
- Non-Hamiltonian paths need to be removed

Polymerase Chain Reaction

- Add many copies of two short pieces of DNA as primers to start Watson-Crick replication
- Primers are the second half of the start city and the first half of the end city
- As a result, molecules with both the right start and end cities are reproduced exponentially

Gel Electrophoresis

- Identify the molecules that have the right length
- Discard the other molecules

Affinity Separation

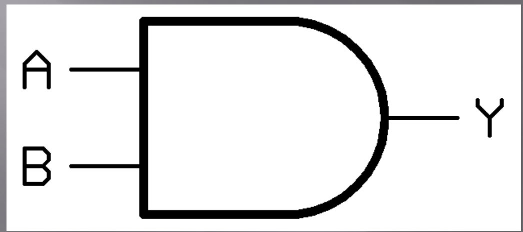
- Use a DNA "probe" to find a particular city
- Probes are attached to microscopic iron balls that are suspended in the tube
- Molecules that contain the desired city anneal to the probes
- Use a magnet to hold the iron balls to the side of the tube and pour out the rest
- Remove magnet, add new solvent, and heat to free molecules from the iron balls
- Apply magnet again, but liquid in a new tube, and repeat procedure for remaining cities

Result

- DNA molecules left in the tube encode Hamiltonian paths
- Additional PCR step, followed by another gel-electrophoresis operation can verify this

Limits of Conventional Computers

- Most operations in computing are irreversible



- Irreversible operation dissipate heat
- Heat is a problem when miniaturising conventional computers
- If a device is physically (not only logically) reversible, then the 2nd law of thermodynamics guarantees that it dissipates no heat

Quantum computers

- On a quantum computer, programs are executed by unitary evolution of an input that is given by the state of the system
- Since all unitary operators are invertible, we can always reverse a computation on a quantum computer
- Details are beyond the scope of this lecture