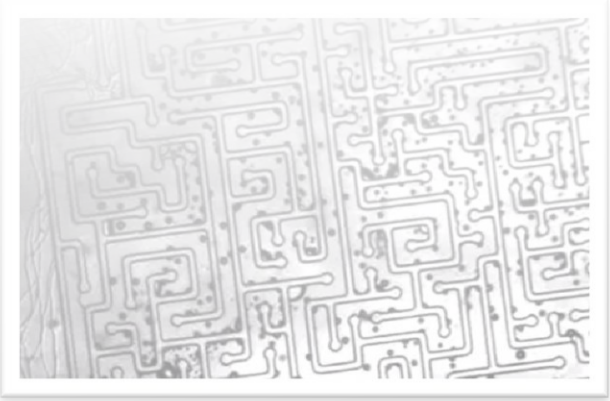
Unconventional Computing:

computation with networks biosimulation, and biological algorithms



Dan V. Nicolau, Department of BioEngineering, McGill University, Montreal, Canada

McGill UNIVERSITY

Acknowledgements

People

- Kristi Hanson, Luisa Filipponi, Swinburne University, Australia
- Marie Held, Ben Libberton, University of Liverpool, UK
- Hsien (Jamee) Lin, Elitsa Asenova, Eileen Fu, Viola Tokarova, Ondrej Kaspar, McGill University, Canada

- **ABACUS consortium:**
- **Heiner Linke**, **Mercy Lard**, Lund University, Sweden
- **Stefan Diez**, **Till Korten**, Technical University Dresden, Germany
- **Falco van Delft**, Philips Research, The Netherlands
- **Alf Mansson**, Linnaeus University, Sweden
- **Dan Nicolau Jr**., Oxford (2x), UC Berkeley, Molecular Sense Ltd.
- Abe Lee, UC Irvine
- Clive Edwards, University of Liverpool
- Sylvain Martel, Ecole Polytechnique Montreal, Canada

- Henry Hess, Columbia University
- Nick Reed, University of Edinburgh, UK
- Len Adleman, USC

- Marcus Roper, UCLA
- Andrew Adamatzky, UWE, UK
- Toshi Nakagaki, University of Hokkaido, Japan

Funding Agencies

DARPA

Australian Government
Australian Research Council

The Leverhulme Trust

NSERC
CRSNG

Disclosure

Co-founder of, and interests in Molecular Sense Ltd.

Outline

- **Introduction**
  - Unconventional computing
  - Motile biological agents
- **Biocomputation**
  - Encoding mathematical problems in networks
  - An example of solving the subset sum problem
- **Biosimulation**
  - Simulation of traffic with biological agents
- **Biological algorithms** for space searching
  - “Intelligent” biological (micro)agents
- **Sum-up and perspectives**

Introduction
Unconventional computing
Motile biological agents

Biocomputation
Encoding mathematical problems in networks
An example of solving the subset sum problem

Biosimulation
Simulation of traffic with biological agents

Biological algorithms for space searching
“Intelligent” biological agents

Sum-up and perspectives

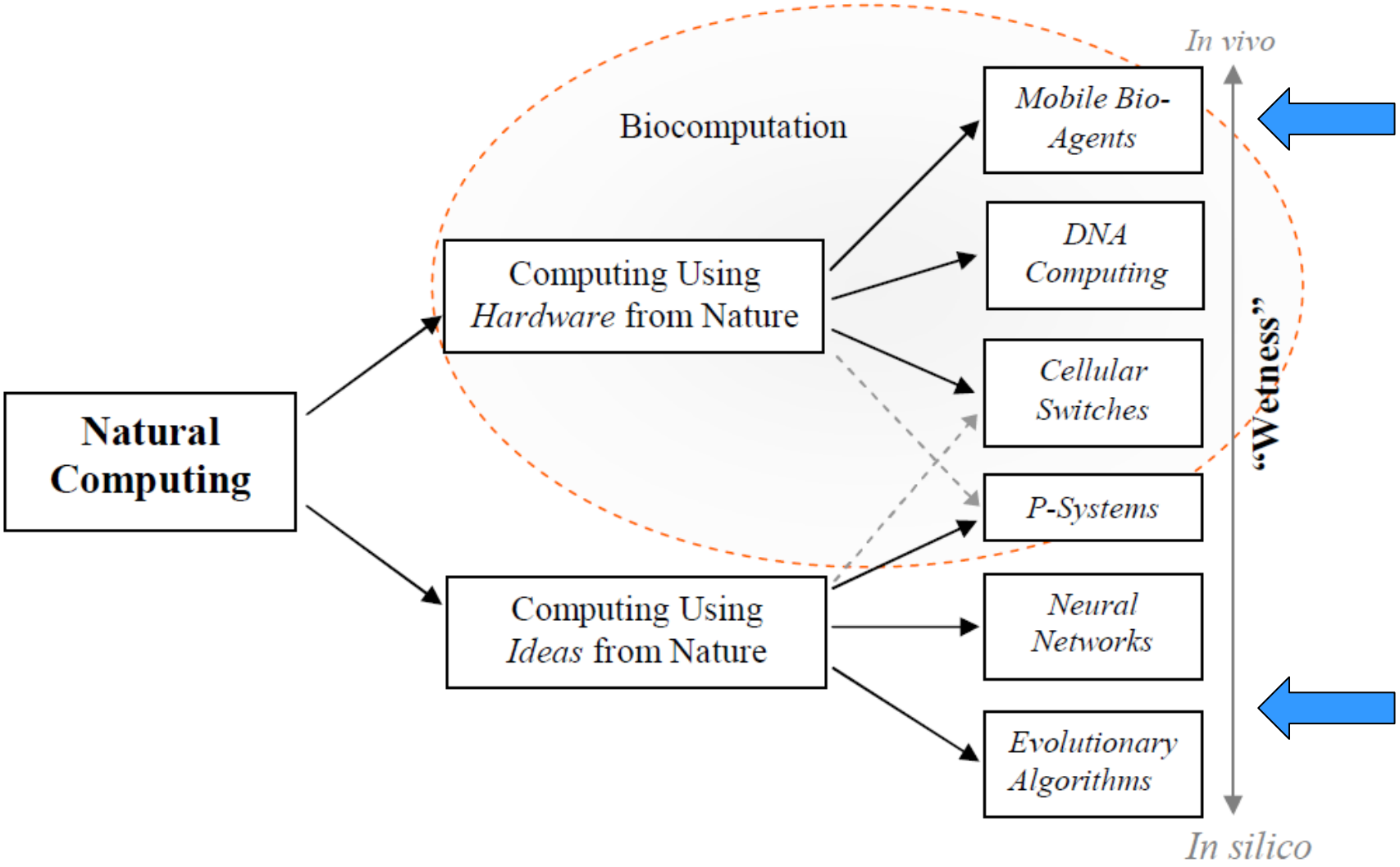
Some concepts and definitions…..

- **Unconventional computation - Biocomputation:**
  - computing process which uses biological entities, e.g., DNA, to perform calculations involving storing, retrieving, and processing data, e.g., DNA computing.

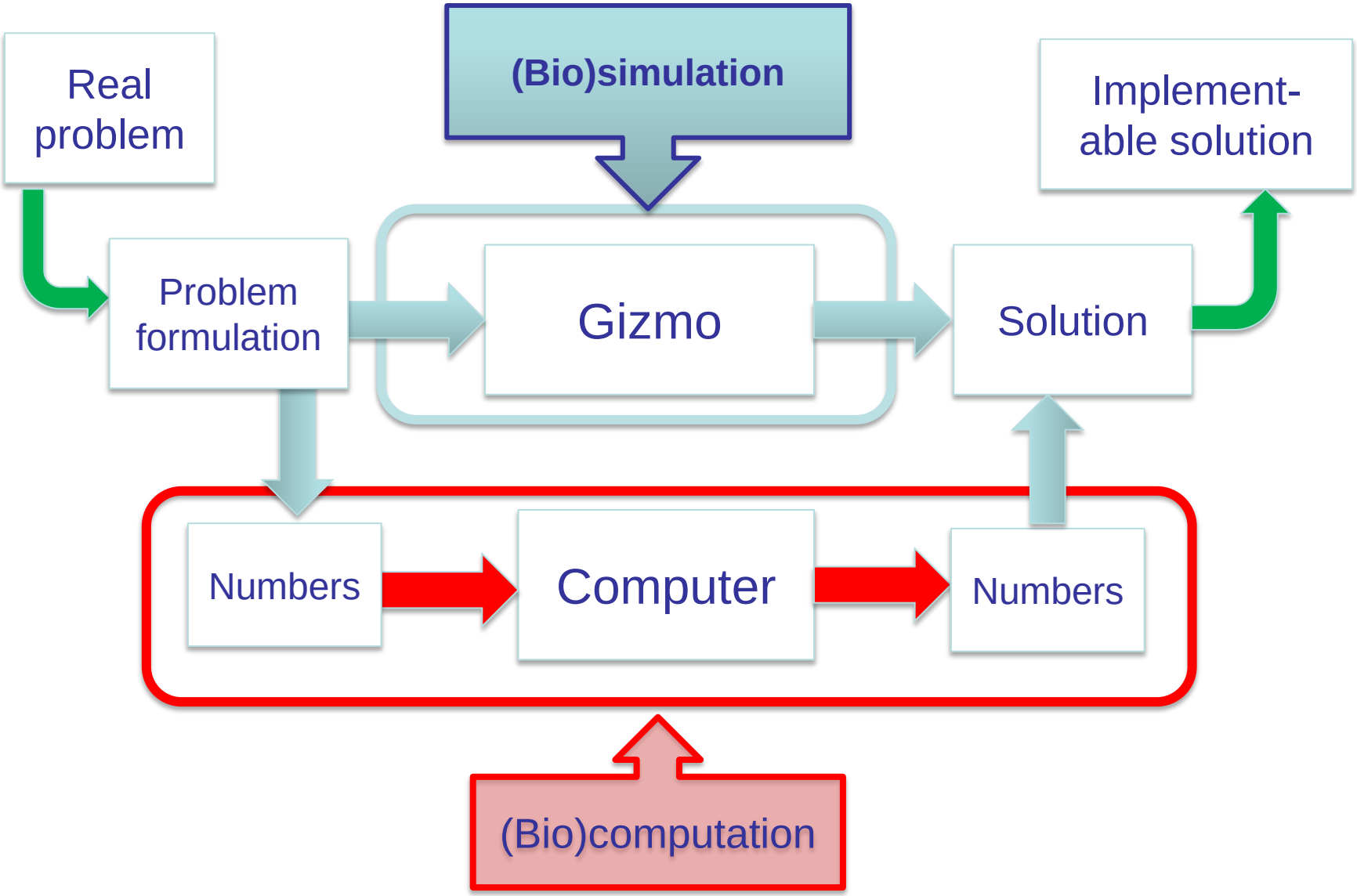
- **Biosimulation**:
  - physical (not in silico) simulation of a real-life process, e.g., traffic of automobiles in city networks, using biological entities, e.g., motile microorganisms moving in scaled down physical networks

- **Biological algorithms** [http://www.algorithmsinnature.org/]:
  - Computer scientists have designed algorithms to process biological data, e.g. microarrays; and biologists discovered operating principles that have inspired new optimization methods, e.g. neural networks.
  - Recently, these two directions have been converging based on the view that biological processes are inherently algorithms that nature has designed to solve computational problems.….

Some concepts and definitions…..

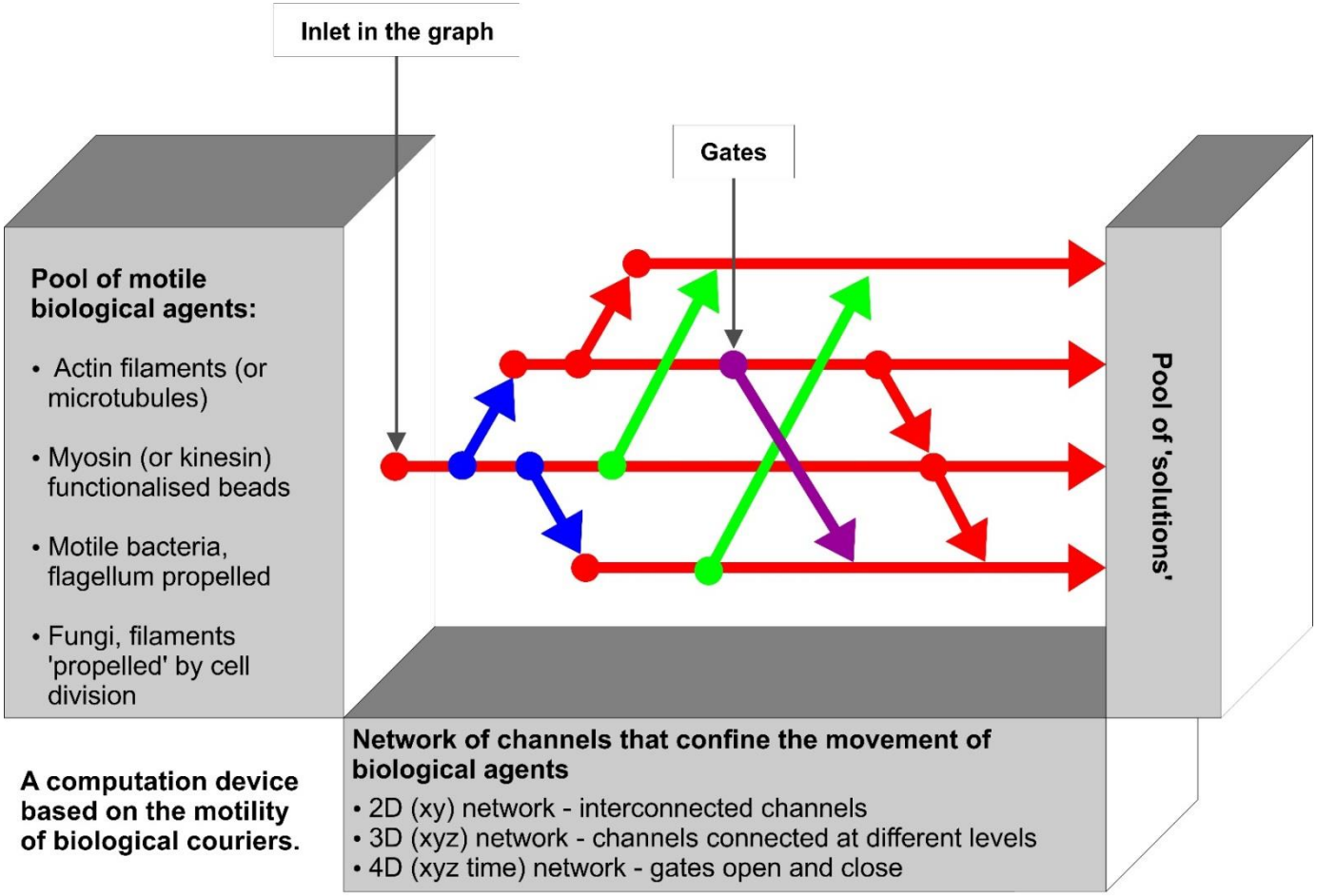


Biocomputation vs. biosimulation



Computation in networks (including bio~)

- **Computing with biological agents in networks:**
  - purposefully designed structures visited by motile biological agents



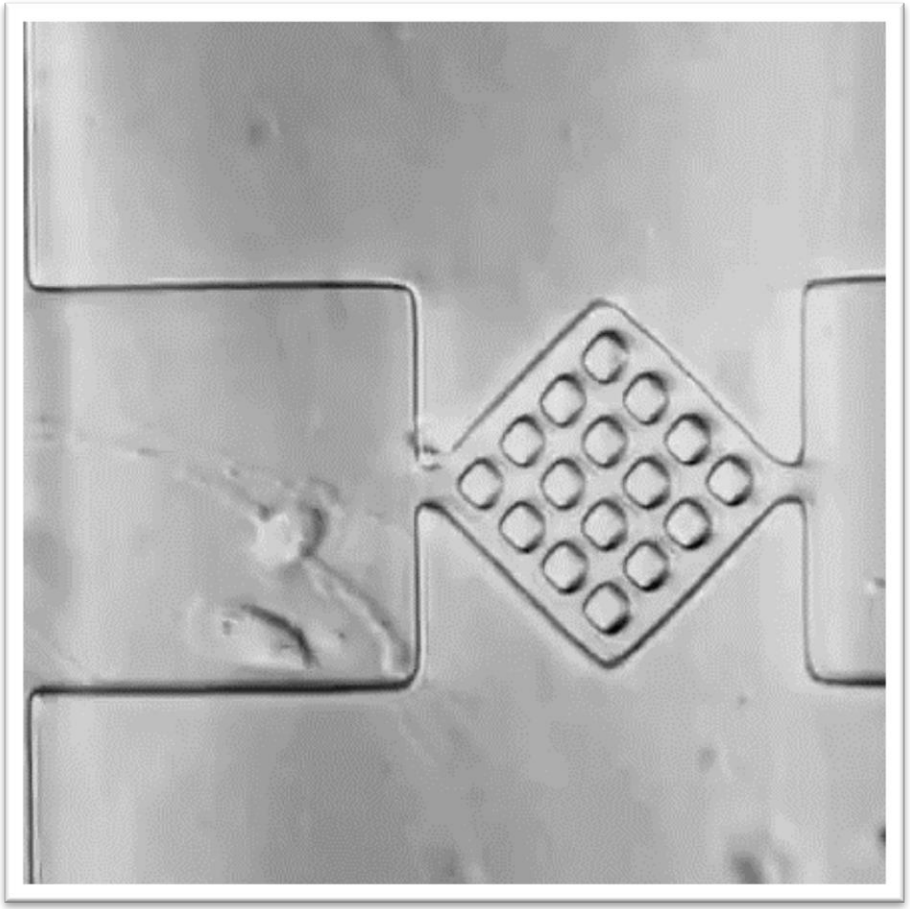
Motile biological agents

- **Motile biological agents**
  - Increasing interest in their dynamic behavior in confined spaces similar to their sizes, e.g., nano- and micro-geometries for protein molecular motors, microorganisms, mammalian cells….

- **Motivations**:
  - **biomedical**, e.g., cell networks for neuronal tissue, quorum sensing in bacterial biofilms, fungal colonization of litter, or space between cells in invaded tissue
  - **engineering**, e.g., cell-based sensors, microfluidics devices with trapped cells, stem cell research
  - **urban planning**, e.g., traffic optimization, evacuation from crowded areas
  - **unconventional computation…..**

- Sizes from nm to µm
- Velocity from µm/s to mm/s

Example of a motile (and static) biological agent in a confined space



Motile glial cells (and quasi-static neuronal cells) in a microfluidic structure

Introduction

Unconventional computing

Motile biological agents

Biocomputation

Encoding mathematical problems in networks

An example of solving the subset sum problem

Biosimulation

Simulation of traffic with biological agents

Biological algorithms for space searching

“Intelligent” biological agents

Sum-up and perspectives

Computation with networks

- Many combinatorial problems of practical importance require that a large number of possible candidate solutions are explored in a brute-force manner to discover the actual solution.
- Examples: design and verification of circuits, folding and design of proteins, optimal network routing.
- Because the time required for solving these problems grows exponentially with their size, they are intractable for conventional electronic computers, which operate sequentially, leading to impractical computing times even for medium-sized problems.
- [Additionally, electronic computers need a large amount of energy.]

Computation with networks

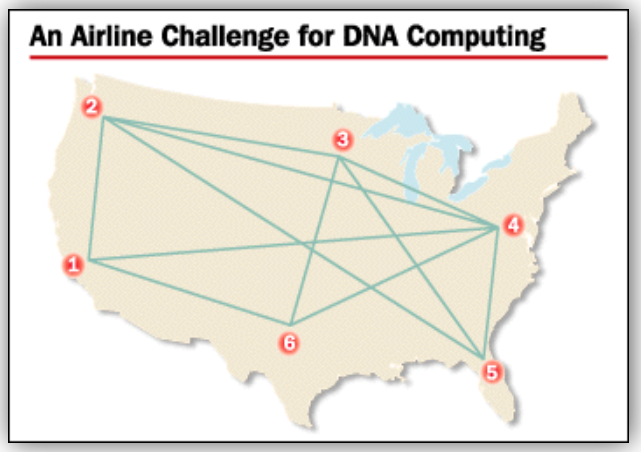
- Solving such problems requires efficient parallel-computation approaches, but those proposed suffer from drawbacks that prevented their implementation:

- DNA computation, which generates mathematical solutions by recombining DNA strands, or nanostructures, is limited by the need for impractically large amounts of DNA.

- Quantum computation is limited in scale by decoherence and by the small number of qubits that can be integrated.

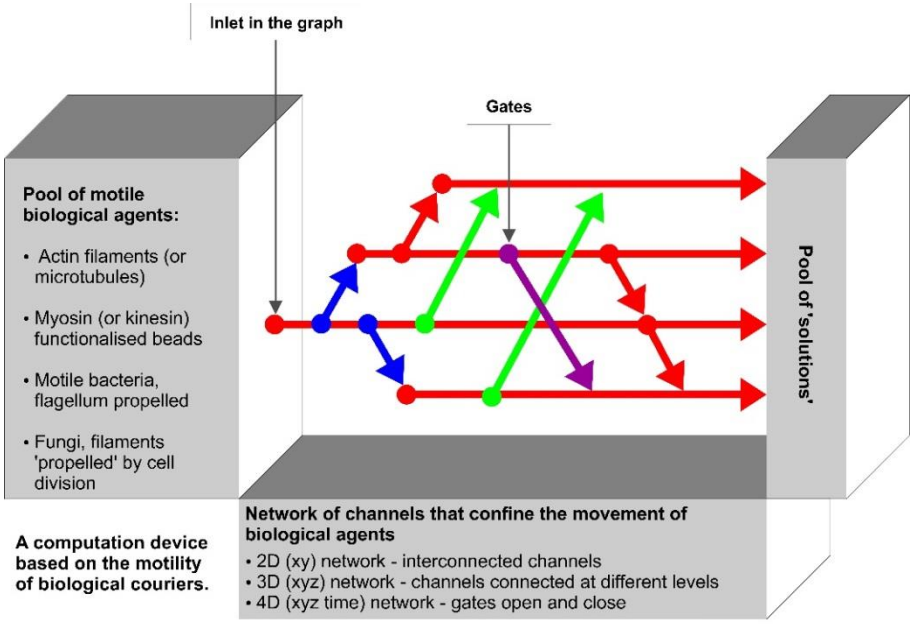
- Microfluidics-based parallel computation is difficult to scale up in practice due to rapidly diverging physical size and complexity of the computation devices with the size of the problem, as well as the need for impractically large external pressure.

Computation with networks



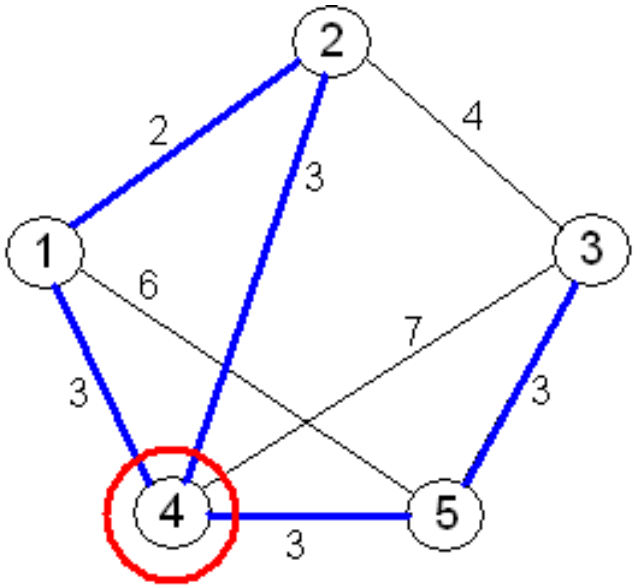
http://corninfo.ps.uci.edu/writings/CHEstory/14dna.gif

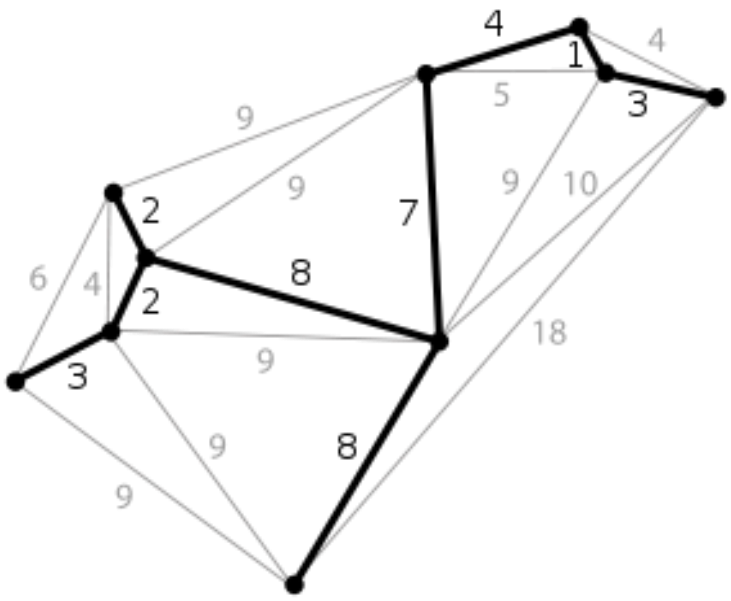
- Most mathematical problems fall into two categories
  - problems in P: polynomial growth of the time required for a solution with the size of the input, and
  - NP complete problems (intractable for a decent size): solution time grows exponentially with the input size.
- Examples NP complete problems:
  - Mathematical formulation: traveling salesman problem, the knapsack problem, the clique problem, the exam scheduling problem, the processor allocation problem
  - Applications: cryptography, scheduling, logistical support, strategizing scenarios
- Difficulty comes from “visiting” all possible solutions, not solving them
- Then, could we (?):
  - Design a graph that encodes a mathematical problem;
  - Translate this graph in a layout of physical network;
  - Allow motile agents explore the network;
  - Derive the solution from their movement.



Computation with networks

- Taxonomy for math problems (combinatorial vs non-, P vs NP)
- Think of rats solving mazes - how many rats does it take to solve a maze efficiently, relative to the maze size?
- Organisational principles:
  - (1) if N agents exploring in parallel can solve a maze/network of size B, then
  - (2) to solve “any” discrete problem, we just need a way to be able to convert a given problem into network “format”
- Couple of tempting places to start (to the right): Travelling Salesman, Minimum Spanning Tree

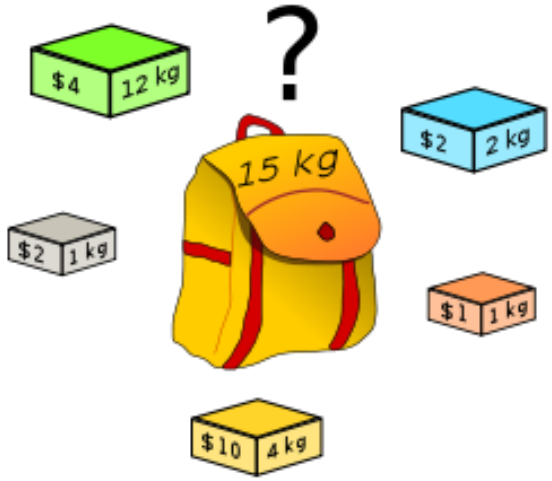




Examples

- **Space allocation** [http://www.claymath.org/millennium-problems/p-vs-np-problem]:
- Housing accommodations for a group of 400 students.
- Space is limited; only 100 students will receive places in the dormitory.
- The Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice.
- The total number of ways of choosing 100 students from the 400 applicants is greater than the number of atoms in the known universe!

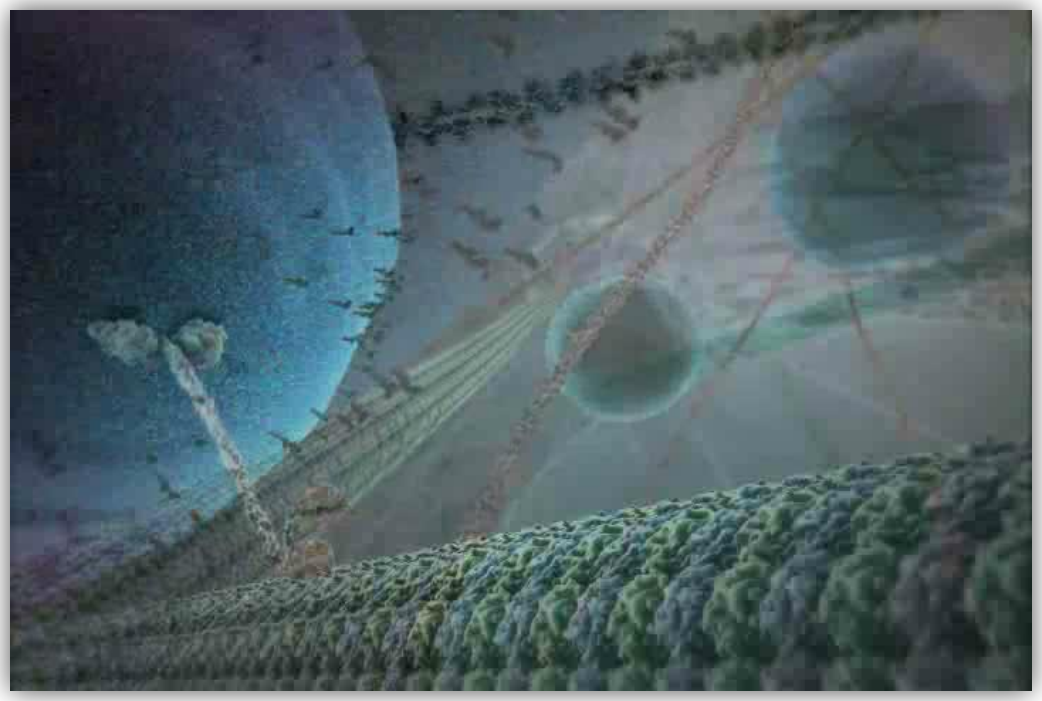
- A 1D (constraint) **Knapsack Problem**:
- Which boxes should be chosen to maximize the amount of money while still keeping the overall weight under or equal to 15 kg?
- A multiple constrained problem could consider both the weight and volume of the boxes.



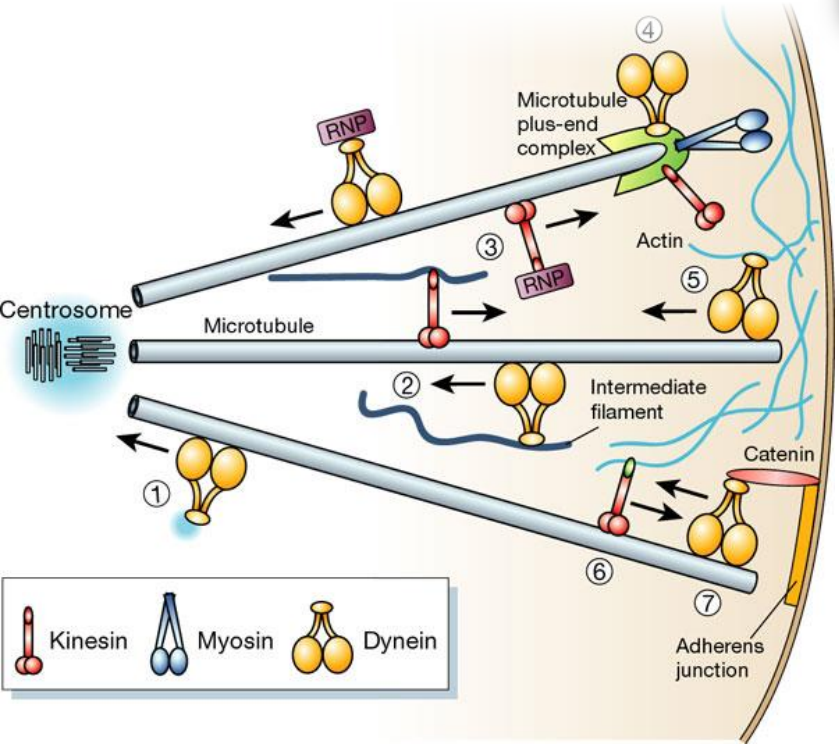
https://en.wikipedia.org/wiki/Knapsack_problem

Biological agents

- Protein linear molecular motors:
- Assemblies of proteins involved in many central biological processes – essentially for biological transport and movement



Alain Viel - http://multimedia.mcb.harvard.edu/

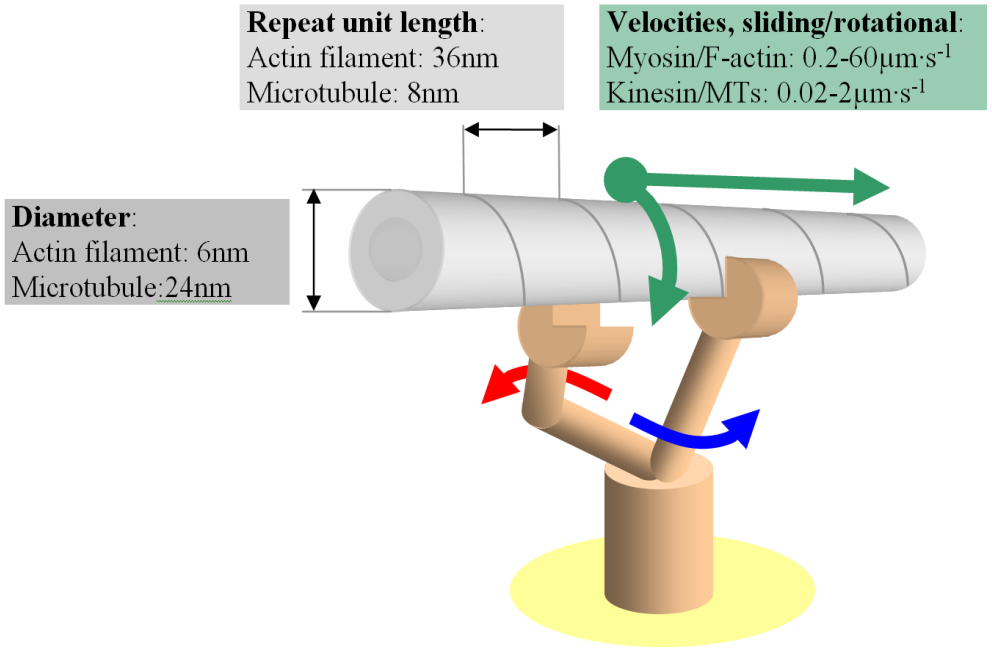


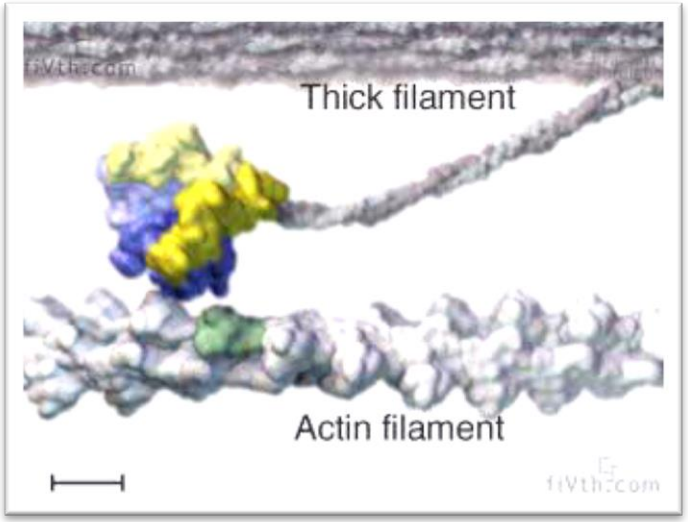
- Some distinguishing features
  - Very complex and coordinated system of molecular motors-based devices
  - Some motors are required for precision, others for force
  - Overall quasi-directional, but otherwise a 3D geometry

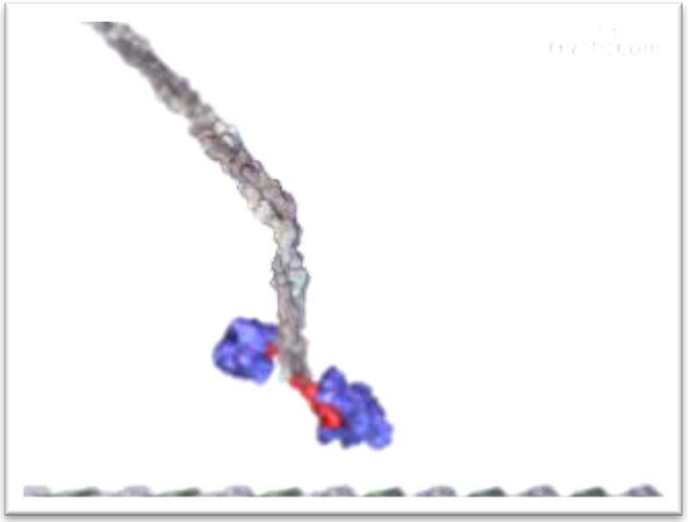
Molecular motors: **Manfred Schliwa** and **Günther Woehlke**
Nature 422, 759-765(17 April 2003) doi:10.1038/nature01601

Biological agents

- Two main types of linear motor systems used in devices
- **Kinesin** – precise, processive, slow; runs on **microtubules**
- ↓ **Myosin** – quick, forceful; runs on **actin filaments**



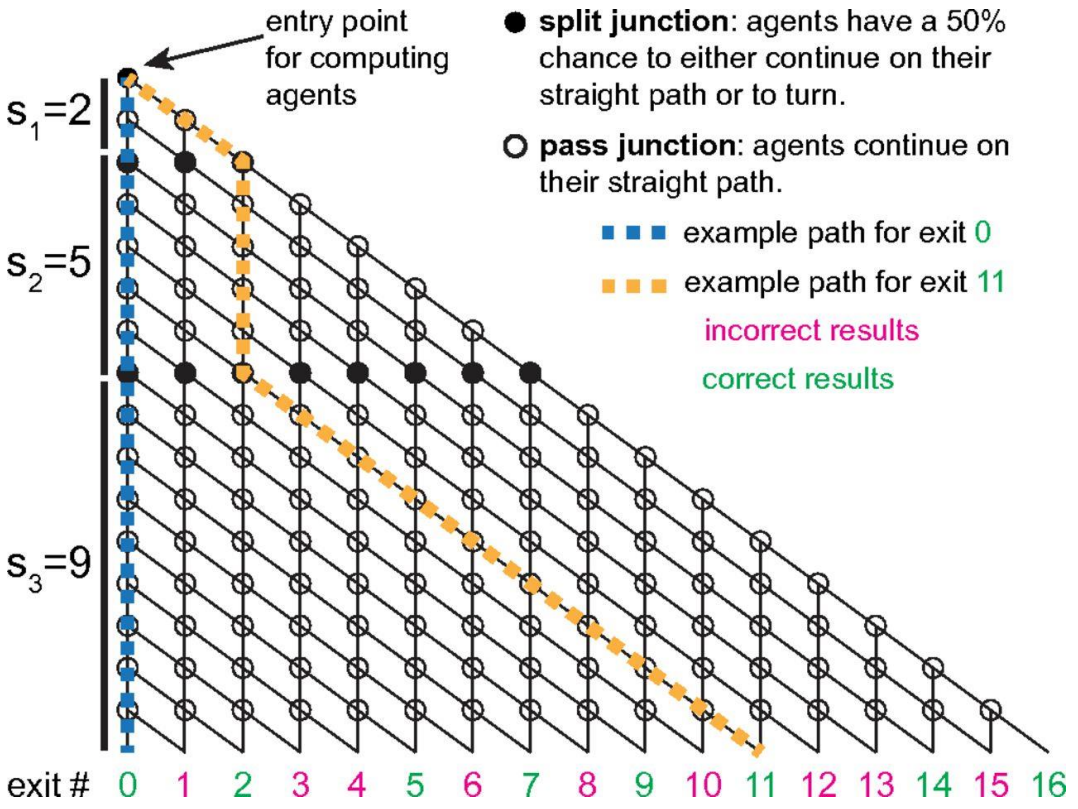




Graham Johnson - https://www.youtube.com/channel/UCz7CvhTKmz6wklnQUWcIK8g

Example: Computation network for the SSP {2, 5, 9}.

- The agents enter the network from the top-left corner.
- Filled circles represent **split junctions** where it is equally probable that agents continue straight ahead or turn.
- Empty circles represent **pass junctions** where agents continue straight ahead.
- Moving diagonally down at a split junction corresponds to adding that integer (numbers 2 and 9 for the yellow example path).
- The actual value of the integer potentially added at a split junction is determined by the number of rows of junctions until the next split junction.
- The exit numbers correspond to the target sums T (potential solutions) represented by each exit; correct results for this particular set {2, 5, 9} are labeled in green, and incorrect results (where no agents will arrive) are labeled in magenta.



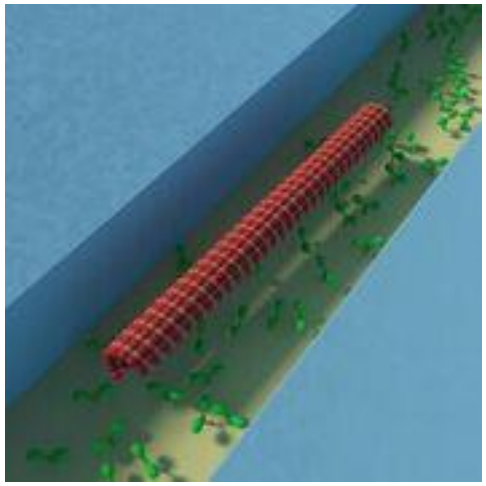
Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

Computation with networks – and agents

Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

Computation with networks – and agents

- Channels functionalized with protein molecular motors, either kinesin, or myosin, are conduits for “agents”, i.e., microtubules, or actin filaments



- Two types of junctions:
- Split junction (left) and
- Pass junction (right)

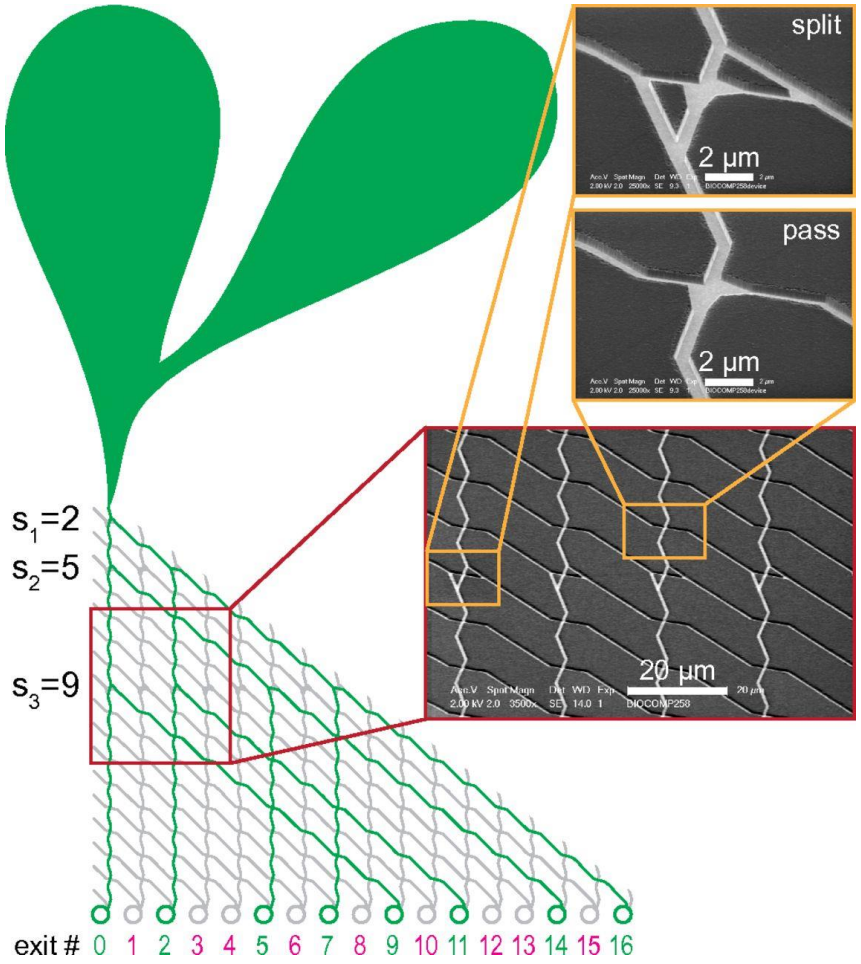
Courtesy of Till Korten @ Technical University Dresdend, Germany

Example: Computation device for the SSP {2, 5, 9}.

- Device layout of a computation device for the SSP {2, 5, 9}.
- Schematic of the device layout using kinesin-microtubules system.
- Green balloon areas: loading zones for the microtubules

- Green lines: the channels traversed by the microtubules during calculation
- Gray lines: channels that should not be traversed (gray lines).

- Exit numbers corresponding to correct results are shown in green
- Numbers corresponding to incorrect results are shown in magenta.
- The circles at each exit are designed to store filaments for easy readout.
- Inset: Scanning electron micrographs of parts of the network used for microtubules, showing a split- and a pass junction.

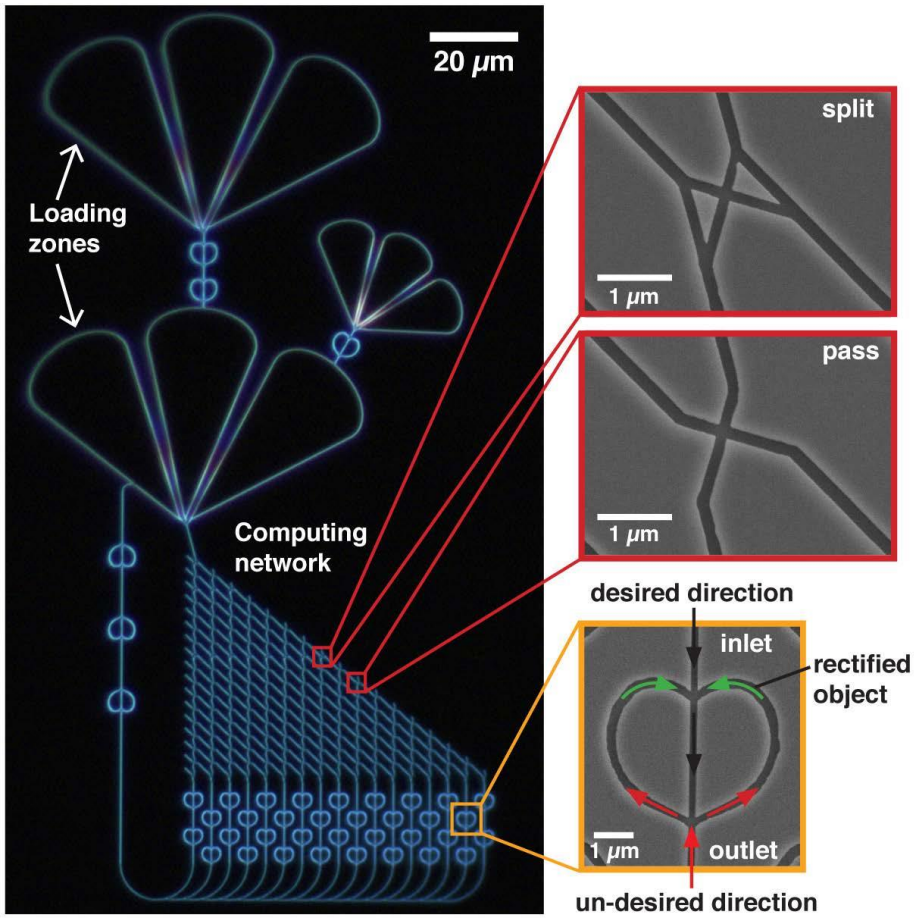


Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

Example: Computation device for the SSP {2, 5, 9}.

- Design of the {2, 5, 9} subset sum device for testing with myosin-actin filaments.
- A series of loading zones (large areas, top left), functionalized with motors are used to bind actin filaments to the surface and to guide them toward the device entrance along the loading-zone edges

- Insets: scanning electron micrographs
- the split and pass junctions, and of
- the heart-shaped rectifiers used to maintain unidirectional actin filament motion.



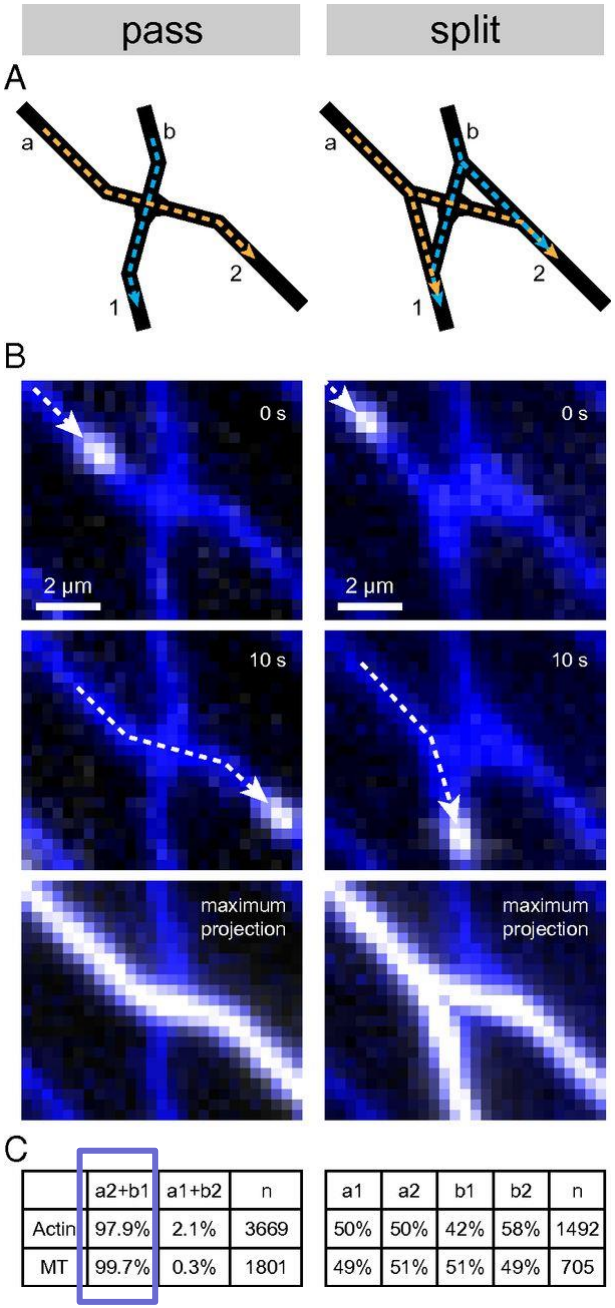
Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

Error rates in junctions

- **Top (A).**
- (a): entrance channels for diagonal movement
- (b): entrance channels for straight down movement.
- (1): exit channels for straight downward
- (2): for diagonally moving agents
- Yellow: intended paths for diagonal movement
- Blue: same, for straight downward movement.

- **Middle (B).** Agents (microtubules) moving in a pass- (Left) or a split junction (Right).

- **Bottom (C).** Error rates.
  - Pass junctions: agents moving from entrance (a) to exit (2); and from (b) to (1) behave correctly (column a2 + b1).
  - Split junctions: agents from each entrance to be split evenly between exits (ideally, 50%).
  - n = number of filaments analyzed for each junction type (MT: microtubules).



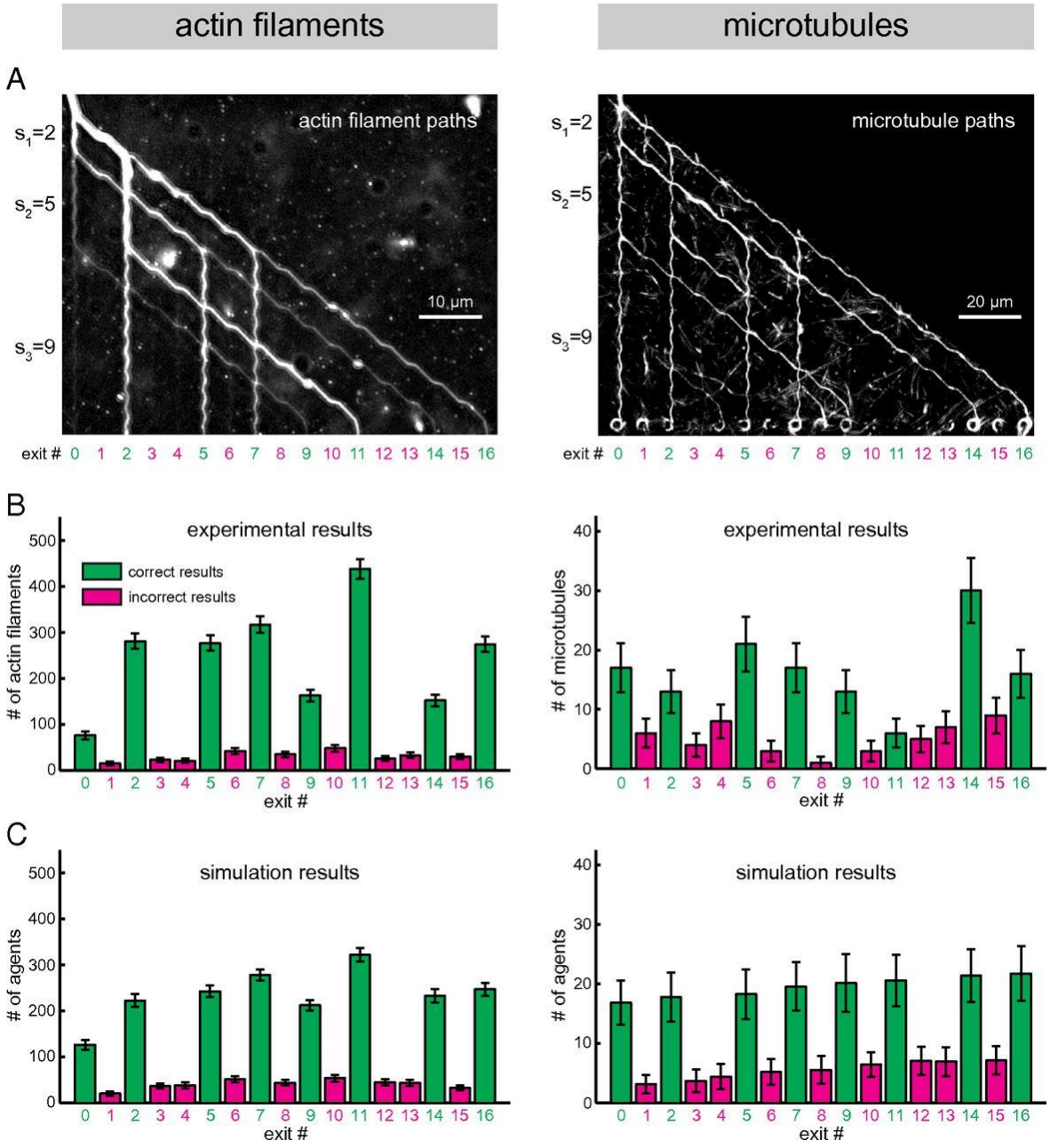
	a2+b1	a1+b2	n
Actin	97.9%	2.1%	3669
MT	99.7%	0.3%	1801

a1	a2	b1	b2	n
50%	50%	42%	58%	1492
49%	51%	51%	49%	705

Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

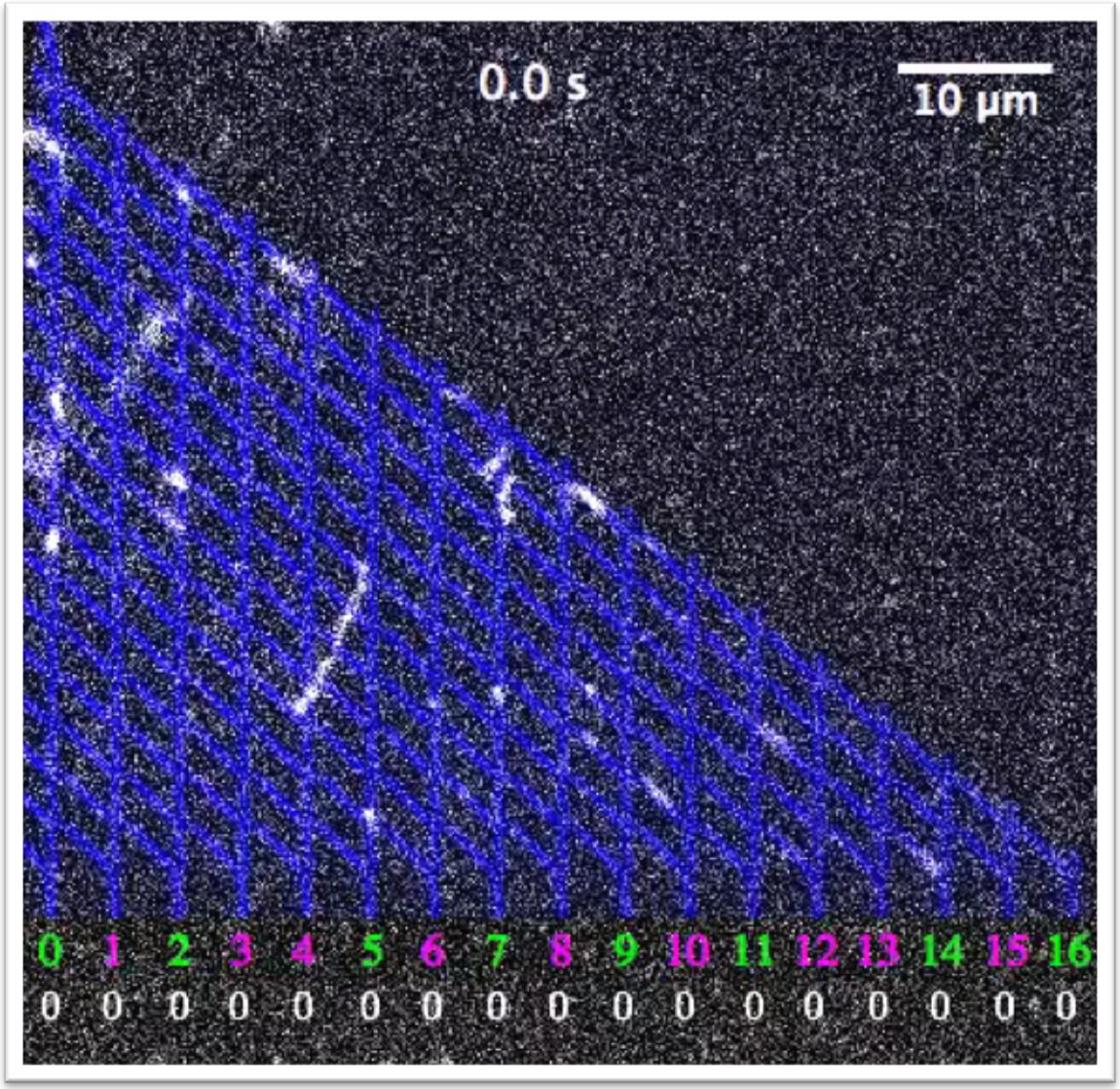
Solving SSP {2,5,9}

- (A) overlap of the trajectories of motile agents.
- (B) Experimental results obtained from actin filaments (Left; time: 26 min) and microtubules (Right; time: 180 min).
- (C) Monte Carlo simulation results (mean ± SD of 100 simulations
- **In A–C:**
- green = correct results
- magenta = incorrect results.



Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

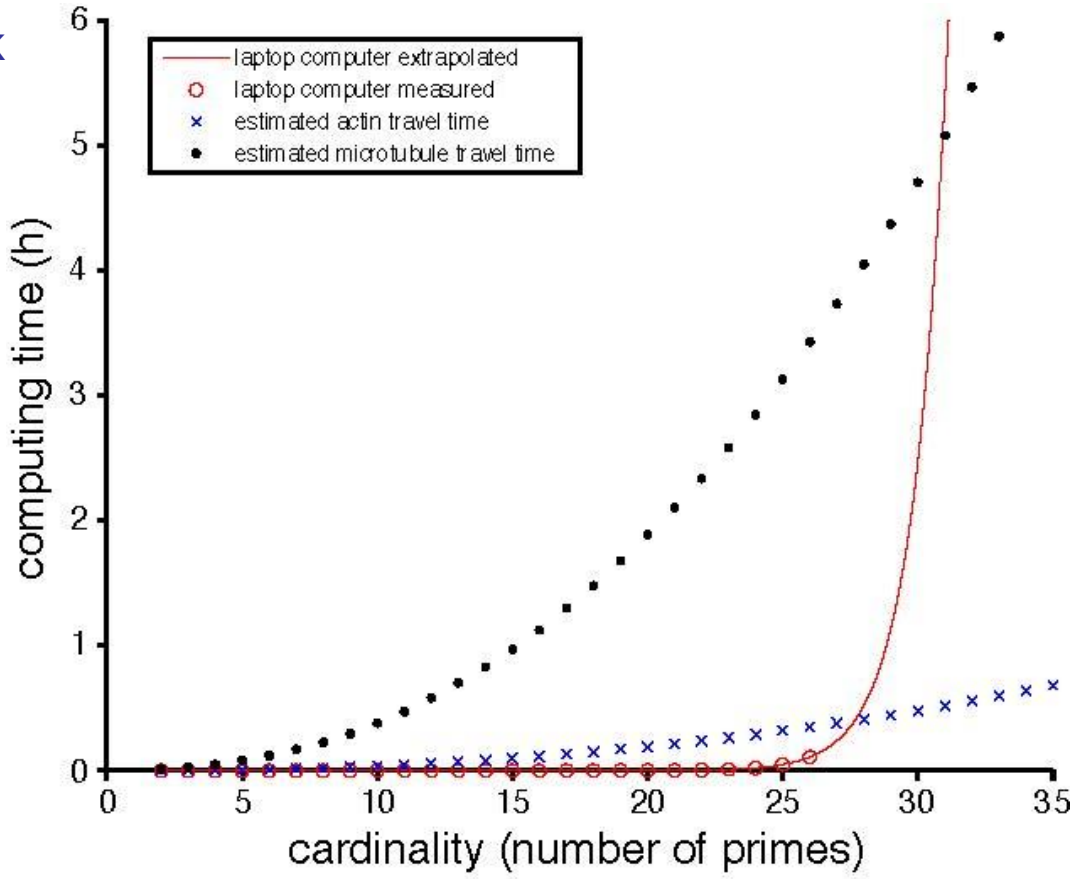
Solving SSP {2,5,9}



Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

Scalability

- Calculated and simulated computing times for sets containing the given number of prime numbers (e.g., {2, 5, 7} has a cardinality of 3).
- The times that actin filaments and microtubules take to travel the longest path through the networks are estimated from their speed and the dimensions of a unit cell of the network.
- The time that a laptop (MacBook Pro, 2.6 GHz core i5 CPU) need to solve the SSP by brute force was measured up to the first 26 primes and then extrapolated with an exponential function.
- The measurement of the computing times for the SSP beyond the first 26 primes was not possible because the limitations of both the CPU and the PC memory resulted in computing times that increased more than exponentially.



Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016

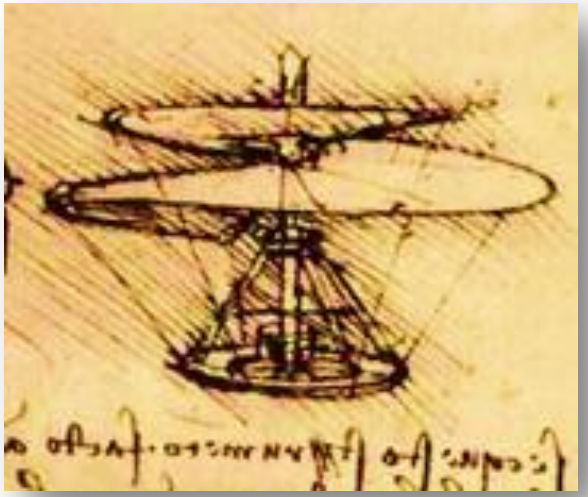
Energy consumption

- Energy efficiency for various computing systems

Computation system	Mode of calculation	Energy required J/operation	Notes, and source
Thermodynamic limit	Theoretical	2.90×10^{-22}	Thermodynamics considerations (26)
Electronic	Actual	6.33×10^{-10}	Intel Sandy Bridge (28)
	Actual	5.26×10^{-10}	Tianhe-2 (MilkyWay-2), top speed, 2013 (29)
	Actual	2.34×10^{-10}	TSUBAME-KFC, top energy efficiency, 2013 (29)
Microfluidics	Estimated	1.29×10^{-12}	Chip with d=200nm x L= 1000nm units; fluidics design (33)
DNA	Estimated	5.00×10^{-20}	Thermodynamics considerations (34)
Molecular motors	Estimated	4.95×10^{-14}	Kinesin/microtubule system, Thermodynamics + design
	Estimated	2.00×10^{-14}	Myosin/actin filament system

Nicolau,Jr. D.V., et al. Proc. Natl. Acad. Sci. U.S.A., 113 (10), 2591-2596, 2016; and references therein

Sum-up: Biocomputing



- **Parallel-computation system** consisting of:
  - a combinatorial problem is encoded in a network
  - embedded in a nanofabricated planar device;
  - network exploration in parallel, by a large number of independent agents then solves the problem.
- Several advantages; and improvements:
  - Molecular motors use distributed energy, and very little of it, needed orders of magnitude less energy than conventional computers.
  - The problem is encoded into a planar network, and not into the agents ….
  - This simplifies the fabrication, as the network grows polynomially with the problem, but the exponentially scaling number of agents are produced in bulk.
  - Once they perform a computation (a path), the agents can be recirculated
  - Need to reduce the error rates, in particular in pass junctions
  - The number of agents can self-adjust to the problem size, e.g., cytoskeletal filaments can self-replicate, or motile dividing microorganisms
  - Any kind of agent multiplication will also solve problems with sequential feeding of the agents through a single entrance, - a bottleneck for large N.
  - ….and so on….

Introduction
Unconventional computing
Motile biological agents

Biocomputation
Encoding mathematical problems in networks
An example of solving the subset sum problem

Biosimulation
Simulation of traffic with biological agents

Biological algorithms for space searching
“Intelligent” biological agents

Sum-up and perspectives

Biological traffic

- Traffic is a common “problem” for living organisms,
- from the macro level, e.g., city traffic,
- to cellular and sub-cellular level, e.g., traffic of motors-carried vesicles
- …for instance traffic in fungal hyphae

Nuclear dynamics in a fungal chimera

Courtesy of Marc Roper@Mycofluidics Lab: https://www.youtube.com/watch?v=_FSuUQP_BBc

Biological traffic

- Biological traffic looks like human-made traffic, e.g., vehicle traffic in cities….

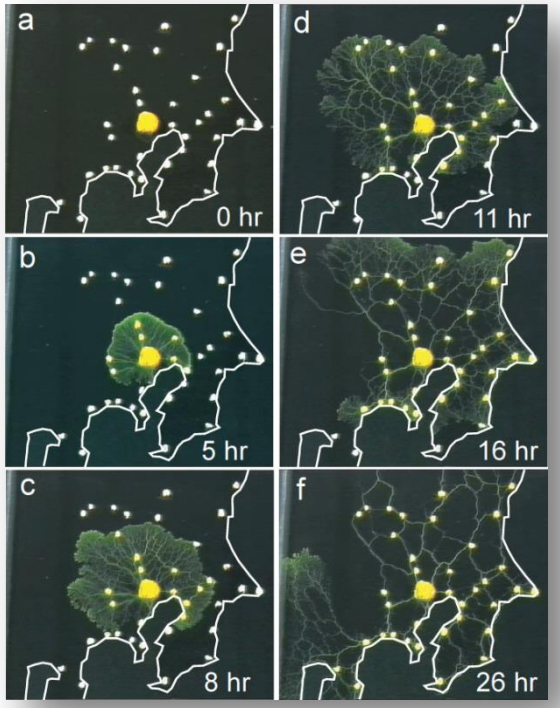
- …however this is much more organised, e.g., straight roads, lanes, traffic lights,
- …and much more homogenous
- To be able to have a reasonable chance to simulate the human-made traffic by biological traffic, the former has to be seen at a much larger scale

Biosimulation of traffic

- An amoeboid constructs very traffic networks similar to Japanese (and many other countries) highways:
  - If it is has chemotactic “clues”, i.e., nutrients in positions equivalent to cities, thus mimicking the “objective function”: connect points with the shortest route;

and

  - If it is provided with deterrents, e.g., lighted areas, which repel amoeboid, in areas equivalent to “forbidden” regions for highways, e.g., mountainous landscape

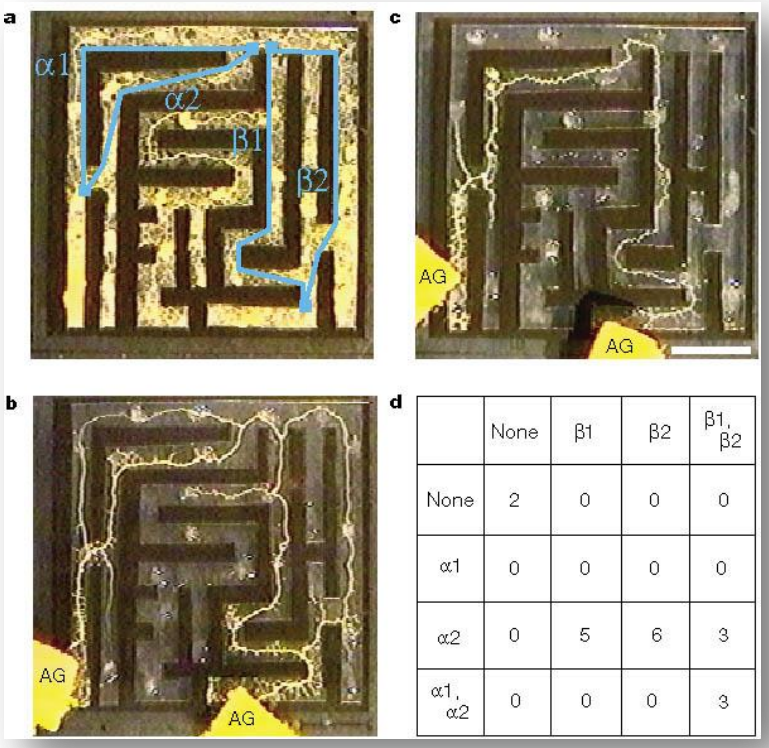


Tokyo rail network designed by *Physarum plasmodium*

A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi and T. Nakagaki, *Science*, 2010, **327**, 439-442.

Other problems….

- The amoeboid finds, eventually, the shortest path in a maze
- Slime mold has been used, repeatedly for various computation tasks, but no sizeable effort to “harvest” the biological algorithms



	None	β1	β2	β1, β2
None	2	0	0	0
α1	0	0	0	0
α2	0	5	6	3
α1, α2	0	0	0	3

T. Nakagaki, H. Yamada and A. Toth, *Nature*, 2000, **407**, 470-470.

Sum-up: Biosimulation

- Microorganisms have an innate capability of optimizing their allocation of resources, such as biomass
- This capability appears evident when attempting to distribute their biomass spatially in order to maximize nutrient uptake
- Biological agents, e.g., amoeboid organism *Physarum*, can be used to ‘simulate’ optimal traffic networks, and the ‘computational evolution’ of getting to this optimum, if ‘targets’, i.e., nutrient-rich locations are spatially defined
- This process uses organism’s innate biological ‘algorithms’
- More “information-rich” experiments, e.g., solving mazes, can be attempted
- Biosimulation bypasses the actual computation, as no numerical output is generated

Introduction
Unconventional computing
Motile biological agents

Biocomputation
Encoding mathematical problems in networks
An example of solving the subset sum problem

Biosimulation
Simulation of traffic with biological agents

Biological algorithms for space searching
“Intelligent” biological agents

Sum-up and perspectives

Types of motile computing agents

- **Artificial**
  - beads, Janus particles, RFID chips
  - easy to fabricate, large range of sizes (x100 nm – x10 µm), can be ID-ed
  - do not move by themselves, in general will require external source energy

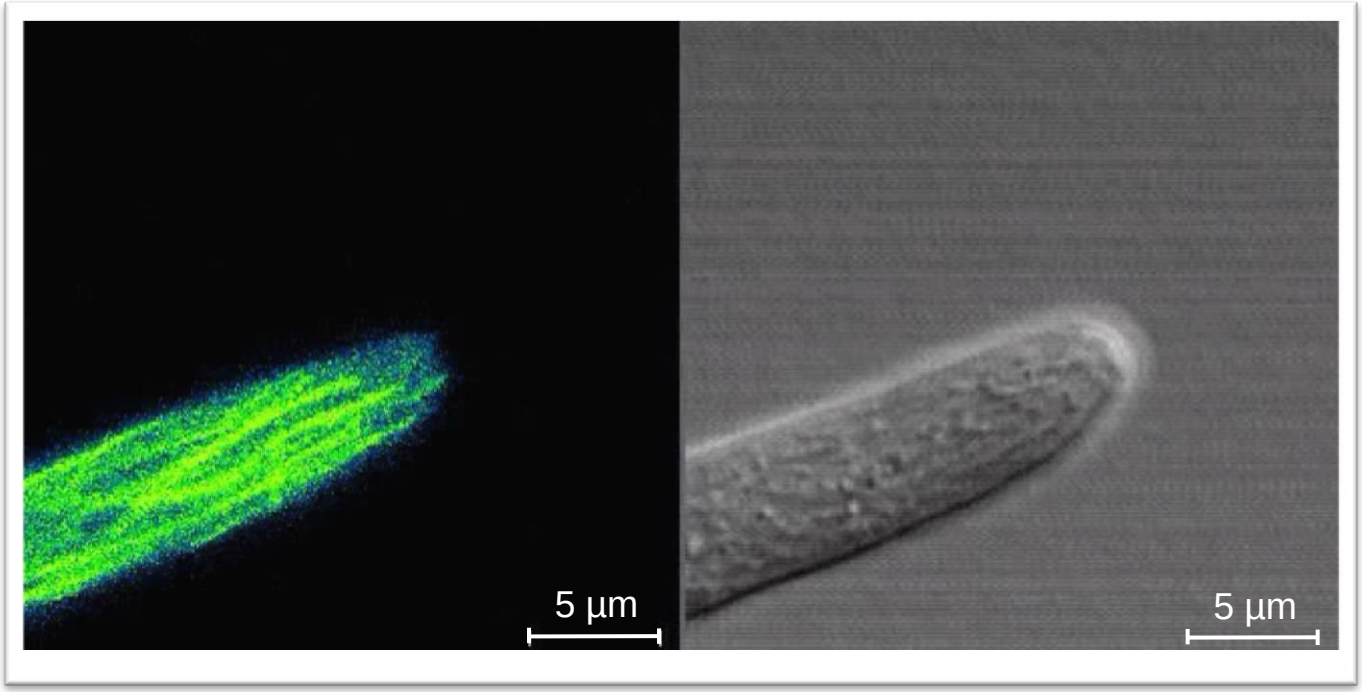
- **Biomolecules**
  - cytoskeleton proteins, RNA molecules, synthetic molecular motors
  - very small sizes (sub nm - x10 nm), truly motile
  - non-trivial synthesis, could denaturate

- **Cells**
  - bacteria, algae, protozoans (e.g., *Paramecium*), mammalian cells (e.g., glia, cancerous)
  - extreme diversity, high speeds, divide
  - larger sizes (x10 µm – x mm), die (i.e., non-motile) extreme variability

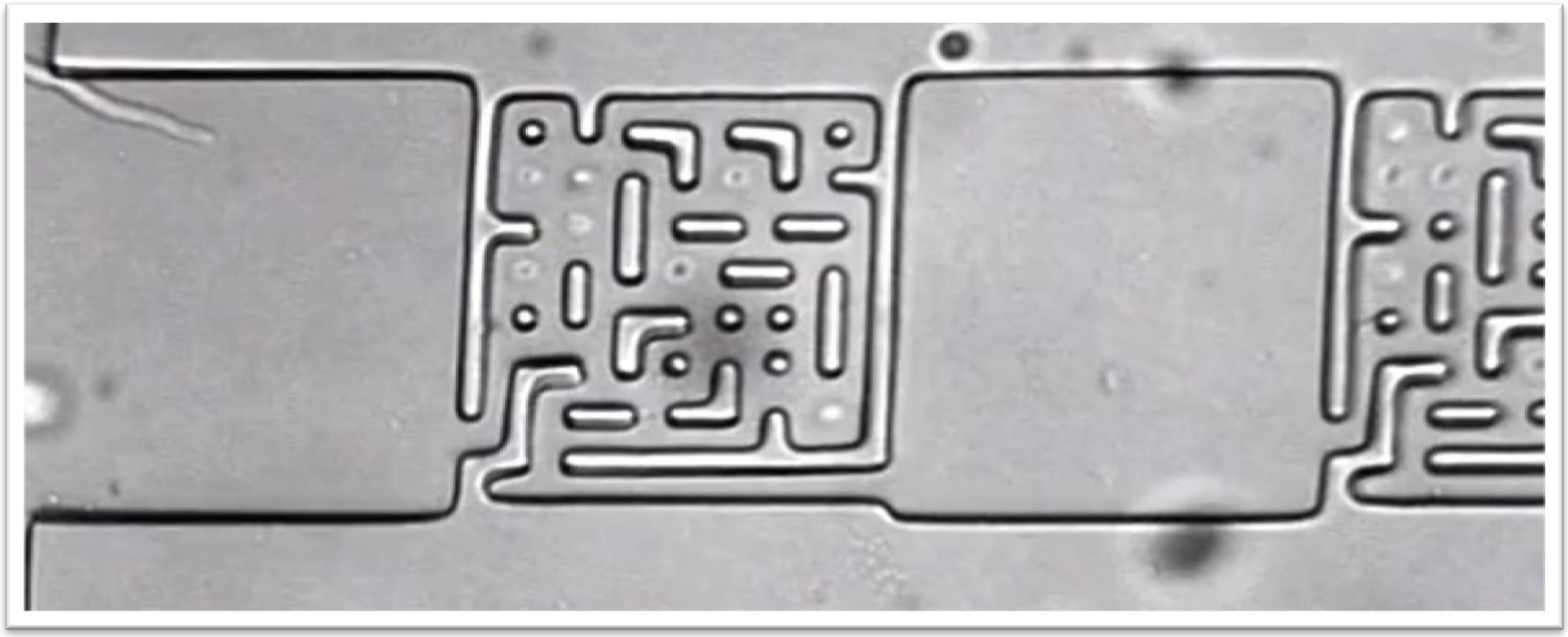
- **Organisms**
  - fungi, worms, e.g., *C. elegans*
  - extreme diversity, high speeds, divide, **intelligent**
  - large sizes (x10 µm – x mm), die, extreme variability

Fungi: efficient space searching “machines”

- The ecological success of these organisms can be attributed largely to the **efficient** expansion of branched filaments (hyphae) in the process of seeking out nutritional resources in the surrounding environment.
- This suggests that they may be efficient solving agents of geometrical problems.
- Basidiomycetous fungi account for 1/3 of known fungal species, e.g., white-rot fungi, edible mushrooms, plant and human pathogens.



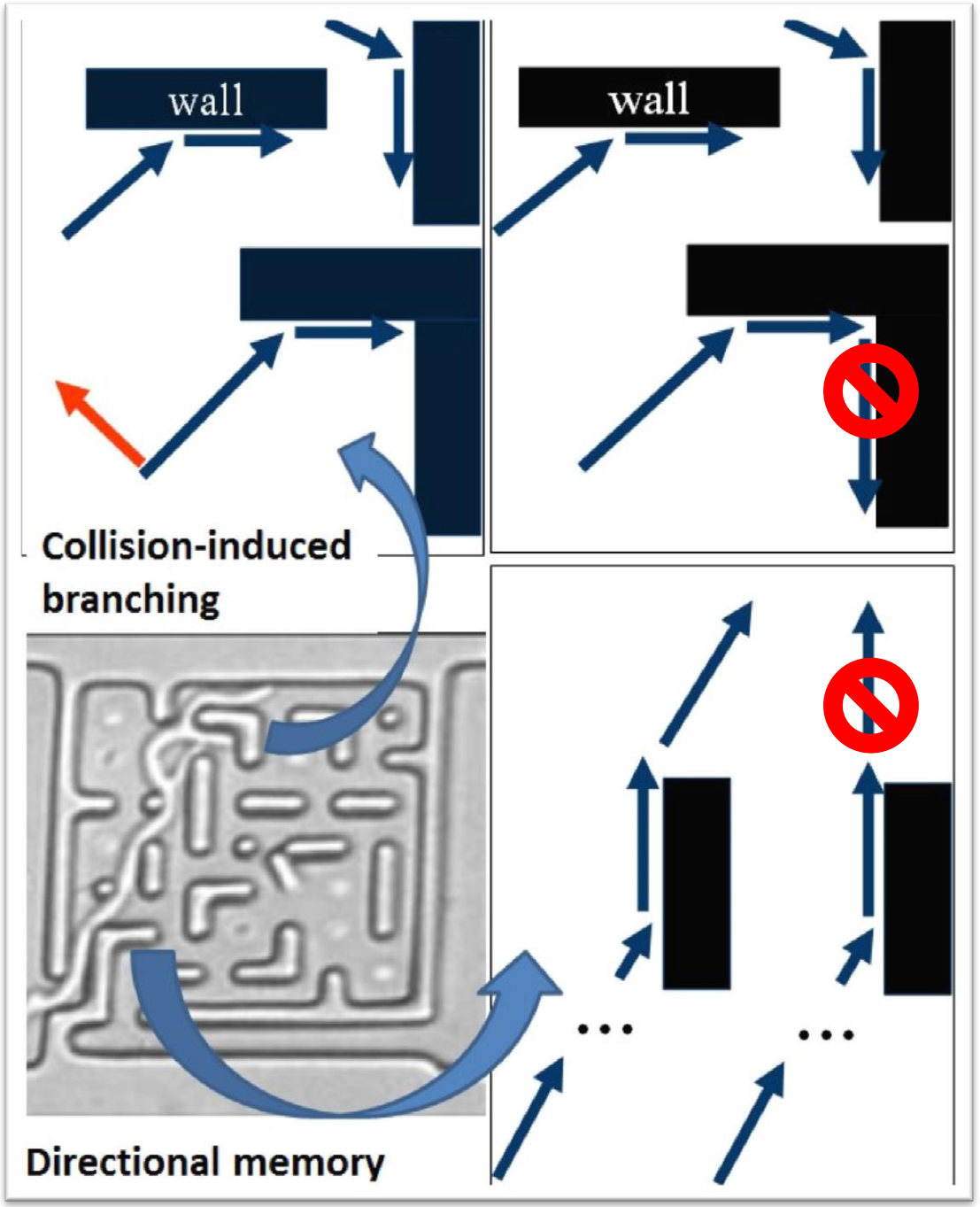
Fungi: efficient space searching “machines”



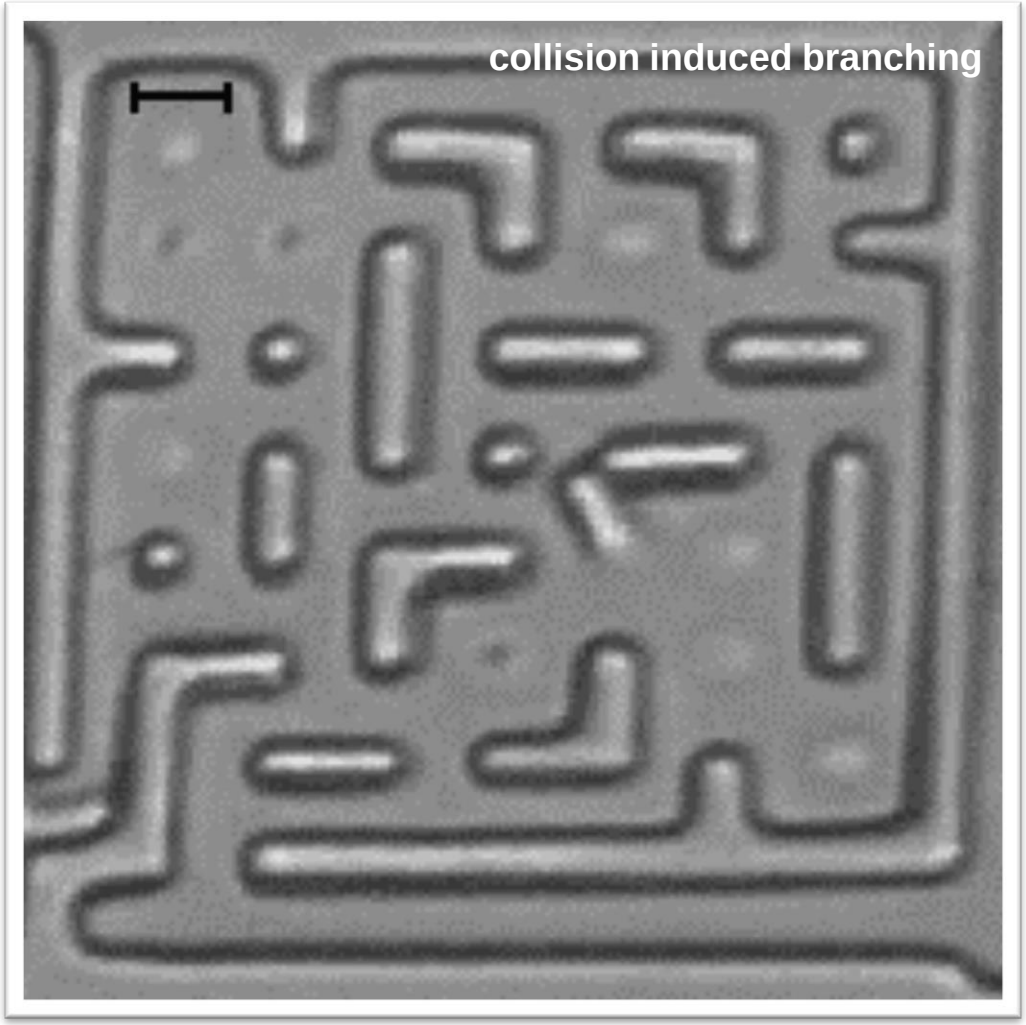
- Fungus negotiating µm-sized confined, closed networks, with dimensions similar with cell size

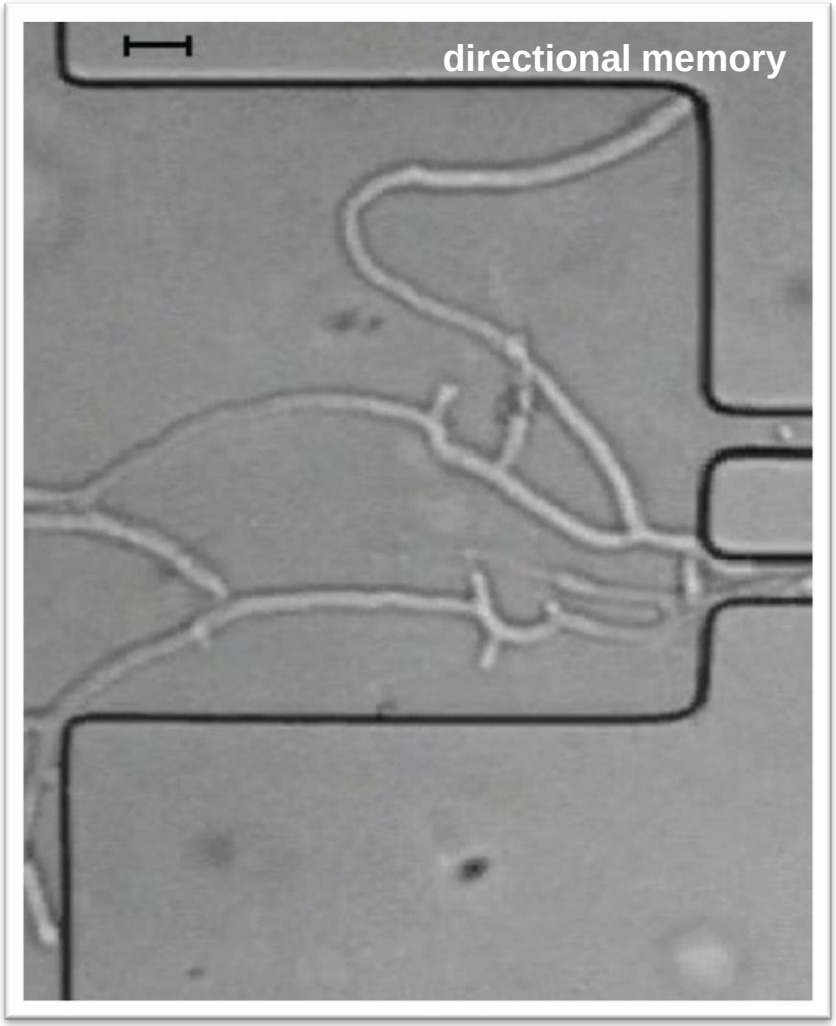
Space searching Master Program

- Space searching algorithms used by a fungus (here *P. cinnabarinus*).
- Top left panel: Collision-induced branching. The hyphae can slide along walls if the angle of attack is shallow. When facing a corner, the hyphae will branch (red arrow), unlike “no collision-induced branching” (top right panel).
- Bottom right panel: Directional memory.



Space searching algorithms

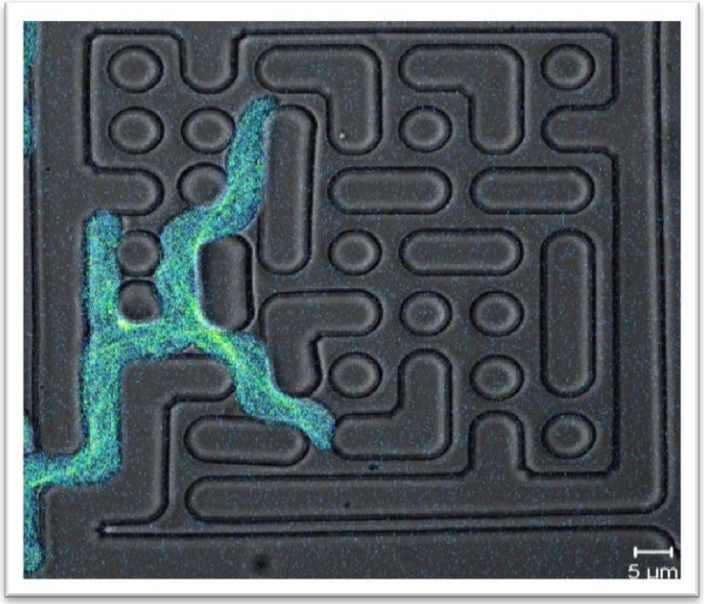


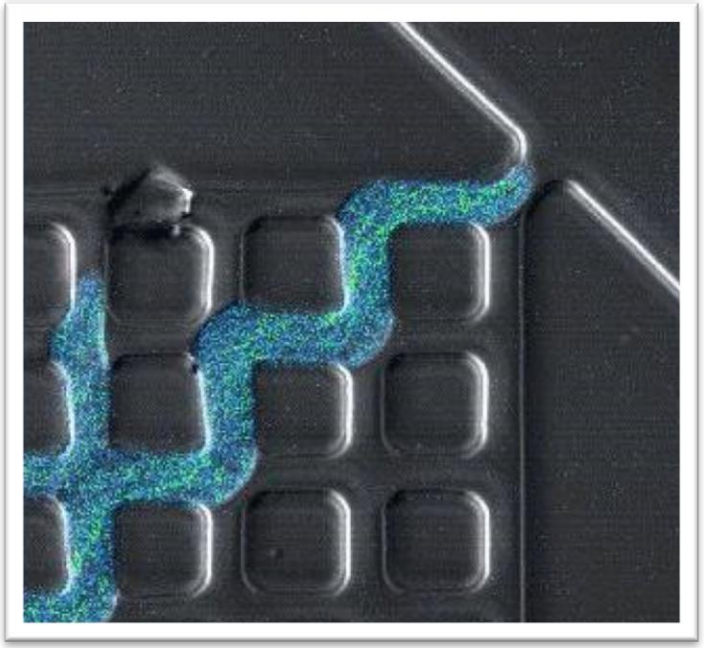


- Two synergetic algorithms: collision induced branching (left) and directional memory (right)

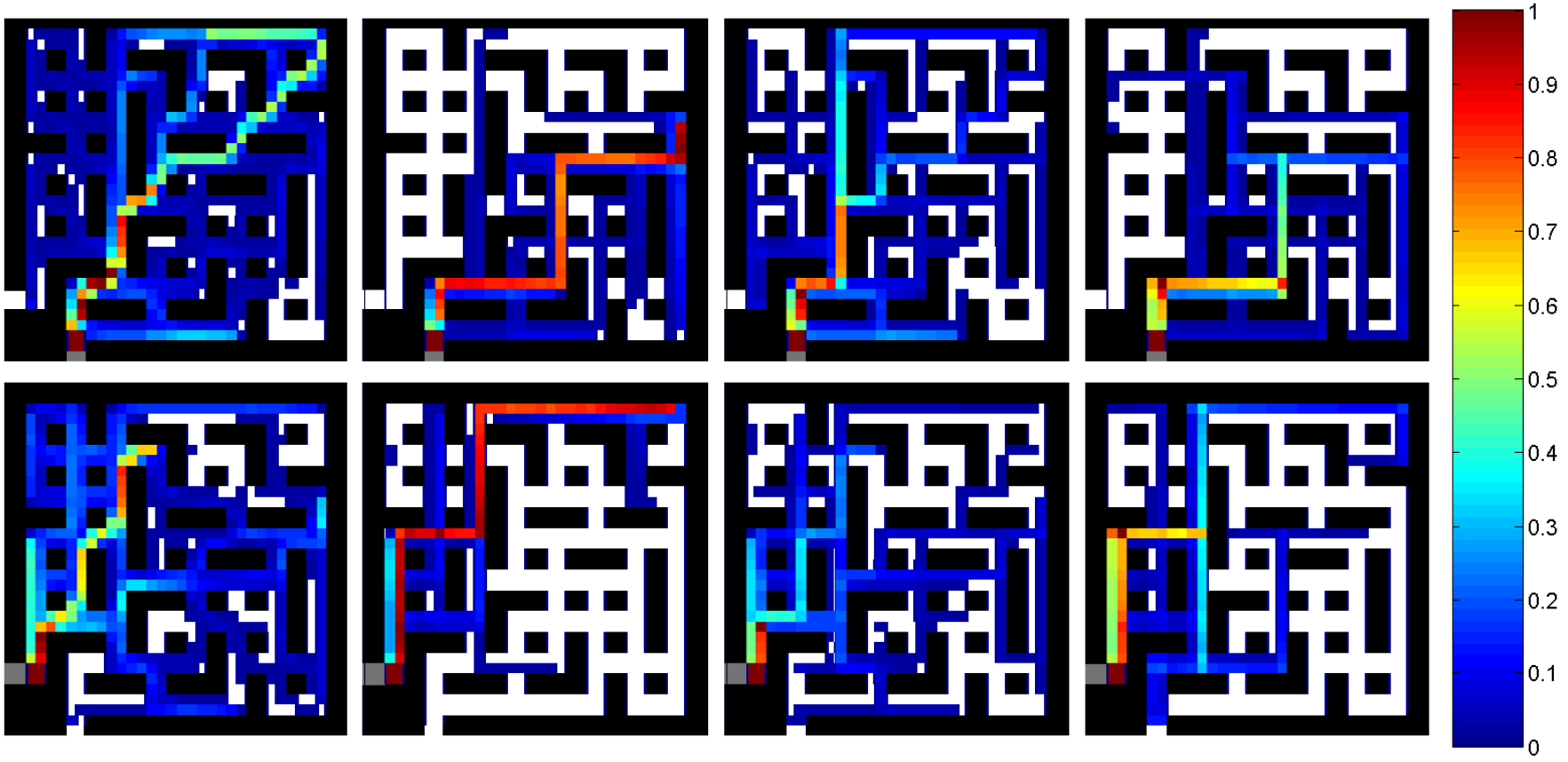
Biological mechanisms behind space searching algorithms

Directional memory: MTs ”cutting corners”





Fungal intelligence



- ↑ Probability distribution of the tip modelled stochastically, from experimental data
- Four possible search strategies, for combination of two “subroutines”
- Directional memory + collision induced-branching (left)
- No directional memory + collision induced-branching (2nd from left)
- Directional memory + random branching (3rd from left)
- No directional memory + random branching (3rd from left)

Operation Mode[1]	Exit Time[2] (min)		Success Rate[3]	
	Left Entrance	Lower Entrance	Left Entrance	Lower Entrance
1a	334	386	0.92	0.94
1b	372	457	0.51	0.07
2a	398	408	0.83	0.81
2b	322	528	0.94	0.21

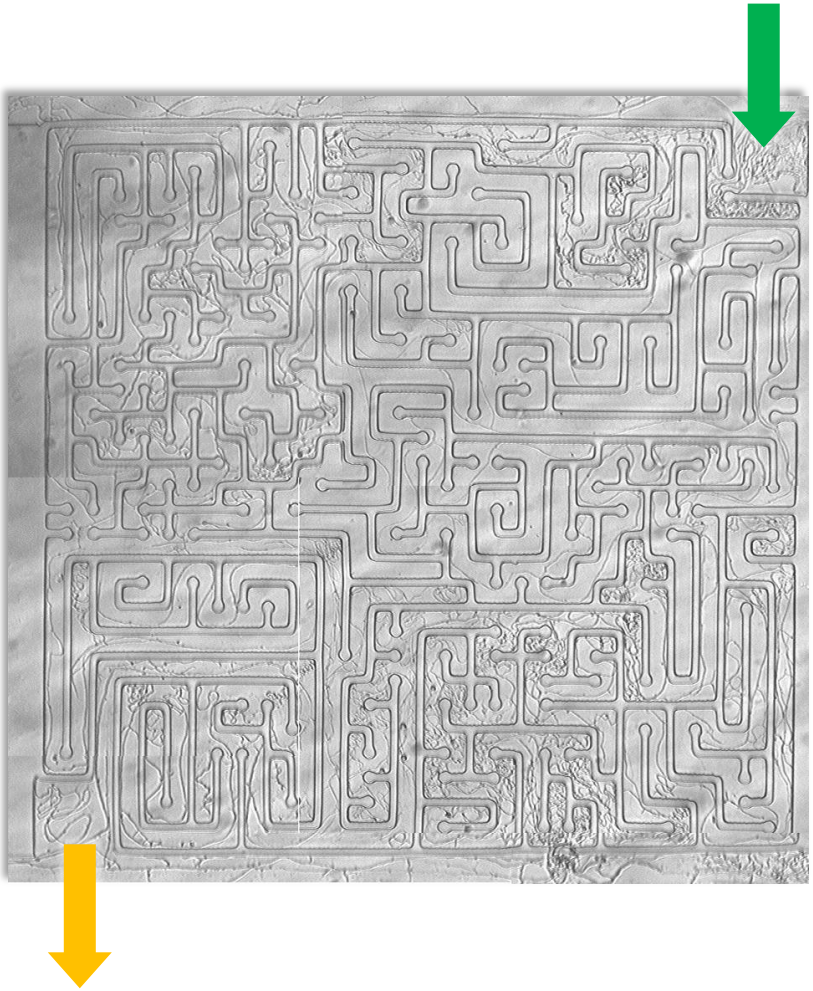
1. Filament turning response at corners was simulated as either (1) dependent on initial branching direction or (2) independent of branching angle. Filament directionality was simulated as either (a) with memory of original branching direction or (b) without memory.

2. Time required for first filament to find exit from the maze.

3. Success was defined as exiting the maze before the theoretical hyphal volume exceeded the total volume of the maze, i.e., as a measure of the ability to exit before overcrowding occurs.

- Fungal hyphae could solve a complex problem, despite couple of errors →

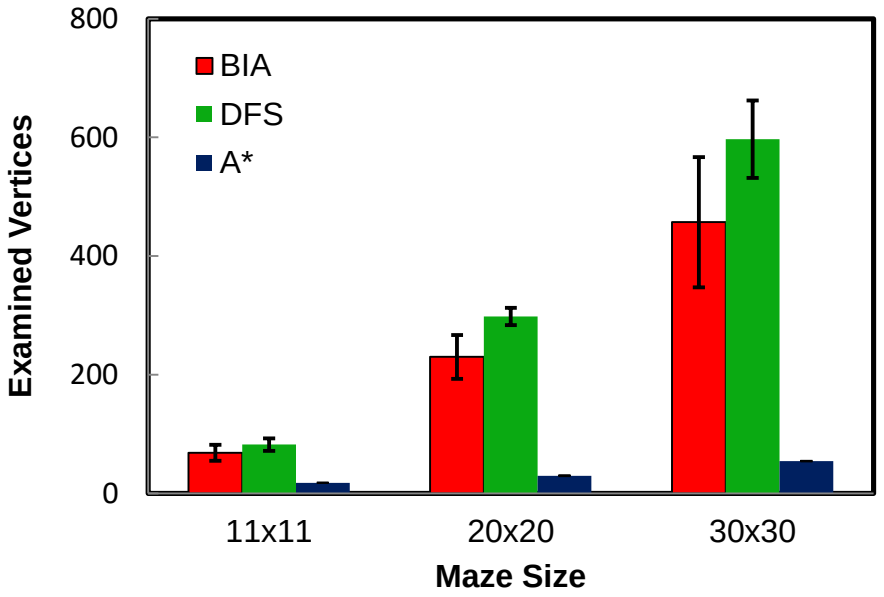
Fungal intelligence:
← results of stochastic simulation

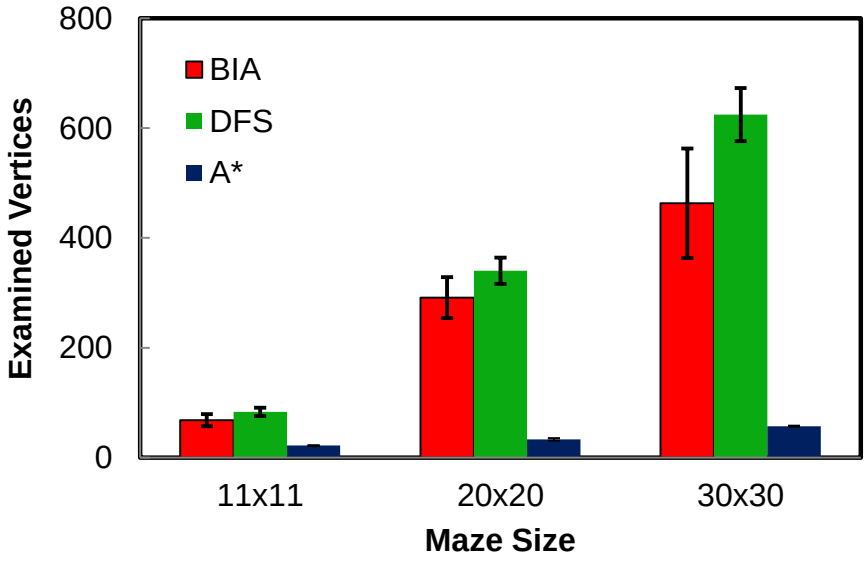


Performance comparison with un-informed space searching algorithms

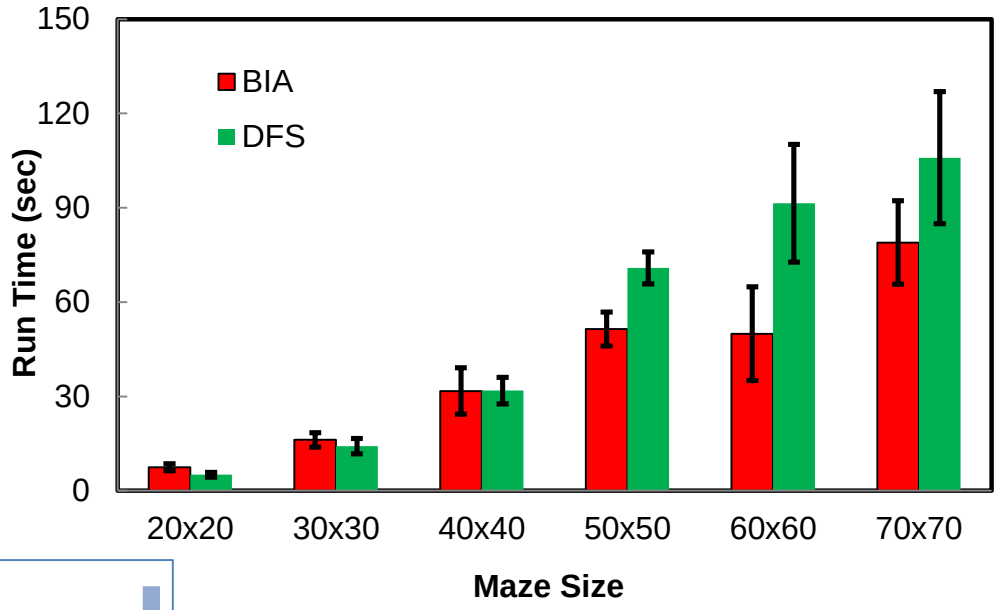
- Reachable space, for non-randomized (top), and randomized (bottom) space, as a result of the exploration of mazes with various sizes by

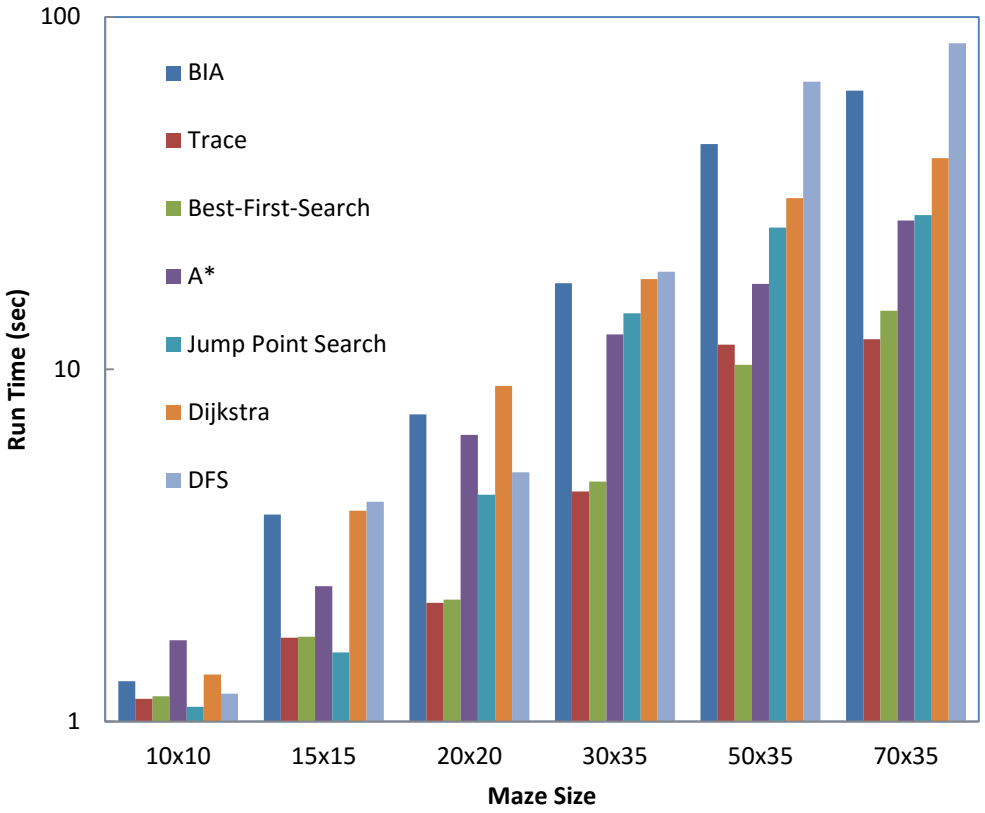
- “BioInspired” Algorithm (BIA),
- Depth-First-Search (DFS) and
- Heuristic Determination of Minimum Cost Paths (A*) algorithms, respectively.





Performance comparison with un- or informed space searching algorithms

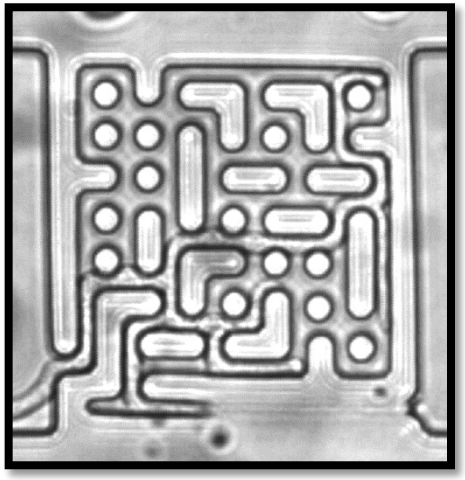




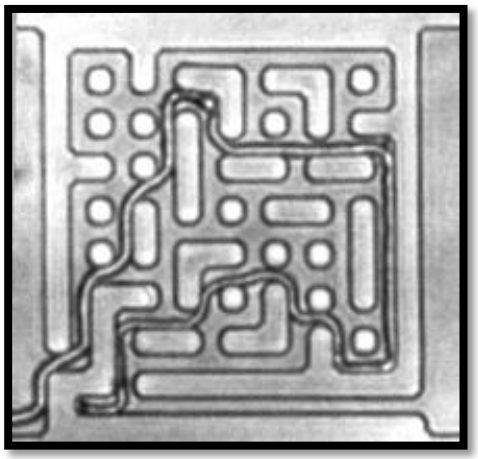
- Computing time for space searching for
- BIA and DFS (top) and
- Informed algorithms (left).

Are all species solving the mazes similarly?

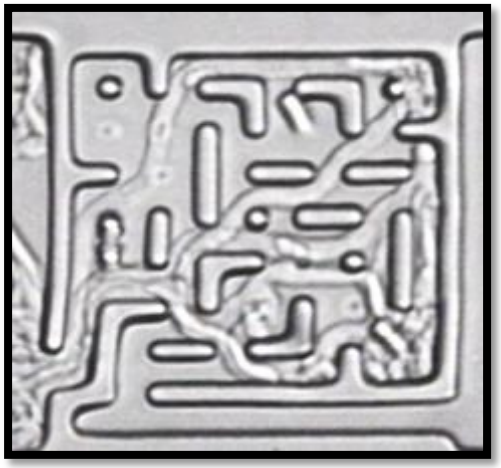
- **Different species show different behaviour**



Neurospora crassa



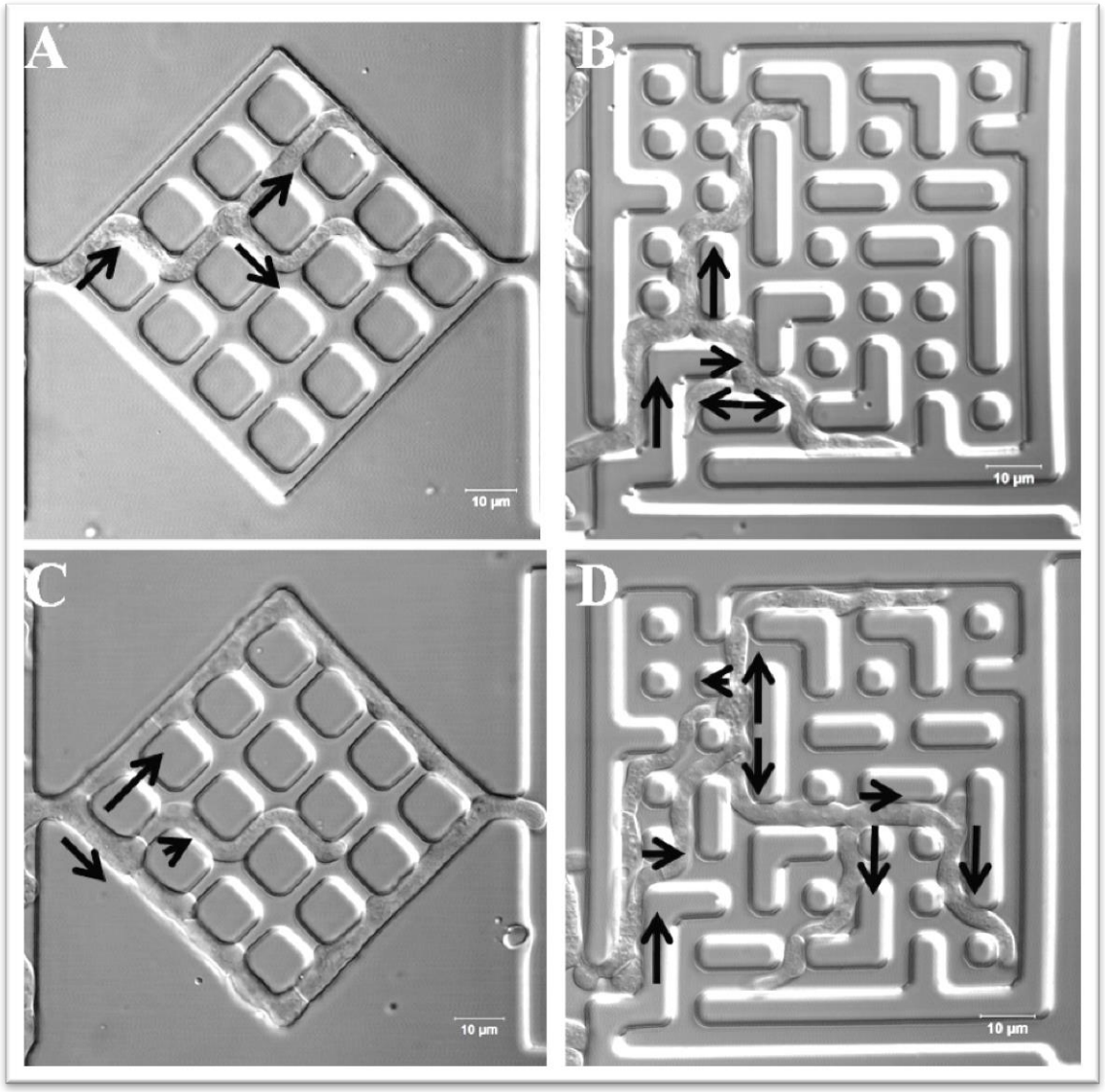
Armillaria mellea



Pycnoporus cinnabarinus

Are some species “smarter”?

- Same! species show very different behaviour:
- Images of the *N. crassa* wild type strain (A,B) solving the diamond and the maze; and *N. crassa* ro-1 mutant
- Arrows = growth directions of the hyphae at the entrance and at the branching points.
- Would it be possible to design and ‘fabricate’ microorganisms that perform simple logical tasks for the exploration of networks?



Biological algorithms

- Algorithms used by fungi in confined geometries for searching available space:
  - Collision-induced branching
  - Directional memory
- Are these strategies “optimal”? Yes, and apparently better than some artificial ones!
- Are they “robust”? Or “Are they geometry-dependent? Work in progress
- Are these strategies species-dependent?
  - The fundamental algorithms appear to be “universal”
  - Some features are species-dependent, e.g., branching location, frequency
- Work in progress: collective behavior, i.e., several hyphae, involving quorum sensing
- … much more to be extracted from the much more complex intra-cellular traffic

- Can biological algorithms, if effective, be translated in computational procedures?
- More comprehensive “harvesting” could reveal strategies that can be “reverse engineered” in new optimal, robust and non-self-evident mathematical algorithms.

- While materials biomimetics is well established, “IT biomimetics” is not

Introduction
Unconventional computing
Motile biological agents

Biocomputation
Encoding mathematical problems in networks
An example of solving the subset sum problem

Biosimulation
Simulation of traffic with biological agents

Biological algorithms for space searching
“Intelligent” biological agents

Sum-up and perspectives

Sum-up and Perspectives

- **Computation with networks and biological agents**
  - Solved a small instance of a NP-complete problem by brute force, using a designed network and biological agents
  - Much work ahead regarding scaling, new network designs, new agents, interfacing with ‘real world’
- **Biosimulation**
  - Simple organisms have innate programs for resource optimization
  - Much work ahead to pass beyond the ‘metaphor’ between biological- to human-relevant traffic
- **Biological algorithms**
  - Biological algorithms for space searching are efficient and robust
  - Much work ahead to ‘harvest’ these algorithms for ‘reverse engineering’

“Some people talk to animals.
Not many listen though. That's the problem.”
[A.A. Milne, Winnie-the-Pooh]

“*Simplicity is complexity resolved*”

[Constantin Brancusi]