University of BRISTOL

Introduction to GAs

Seth Bullock

bristol.ac.uk

Introduction

Evolutionary Computing (EC)/Evolutionary Algorithms (EAs)

- used for black-box function optimization/design
  - e.g., find a wing design with low drag & weight and good lift
- also used for discovering software agent strategies, robot control structures, neural network designs, art, etc. …
- and sometimes used for biological modelling
- or even exploring “life-as-it-could-be” (“Artificial Life”)

University of BRISTOL

Biological Inspiration

Evolutionary algorithms are a nature-inspired/bio-inspired computing paradigm, of which there are many examples:

- Neural Networks
- Ant Colony Optimization
- Swarm Intelligence
- Artificial Chemistry
- DNA Computing
- Artificial Immune Systems

University of BRISTOL

Biological Inspiration

Nature is a great source of inspiration, but:

- All of these paradigms are simplified compared to nature
  - is something important missing?
  - has something irrelevant been included?
  - what exactly makes these paradigms work well in nature?
  - in what situations will they work well for us?

University of BRISTOL

Biological Inspiration

Nature is a great source of inspiration, but:

- Nature's solutions may not be best for us:
  - brains: highly parallel, but individual elements are very slow
  - desktop computer: mostly serial, very fast components
- Nature has only explored a subset of possible solutions:
  - e.g., what about three or four parents, rather than just two
  - e.g., what if kids inherited parents' knowledge in their genes?
    - Not possible in nature, but it is possible in a computer…

University of BRISTOL

Biological Inspiration

Nature is a great source of inspiration, but:

- Natural systems have limits
  - Evolved creatures are amazing, but they evolved to fit a *niche*
  - They did not evolve to be general purpose or universal
  - An immune system has limits; An ant colony has limits
  - Even a brain has limits – it is not good at all types of thinking
  - Just because biology is cool, doesn't mean a bio-inspired approach is the best… Copying nature is not a magic bullet!
    - Naturalistic Fallacy: "You can't get an *ought* from an *is*."

University of BRISTOL

Engineering View

EC is a form of stochastic (randomized) search

- We seek the highest quality solution in the least time
- Evaluating all solutions is guaranteed to find the best one
  - ...but it is slow!
- EC's defining feature: a *population* of solutions search over the (possibly infinite) space of solutions
- EC is often used when traditional search methods are not appropriate or the problem is poorly understood…

University of BRISTOL

Biological View

- Nature rewards effective behaviour with more offspring:
  - Better at catching food => more offspring ☺
  - More effective immune system => more offspring ☺
  - Better eyesight => more offspring ☺
  - Tendency to fall off cliffs => fewer offspring 😐
- "Fit behaviour": Behaviour that "fits" its environment.
- "Fitness": The tendency to generate more offspring.
  - …*in a context*, environment, task, setting…

University of BRISTOL

Biological View

Nature is a ‘blind watchmaker’:

- Creates complex things… …that seem *consciously designed*
- But it is just a noisy copying process that rewards success
  - ‘heredity+variation+selection’ ..or.. ‘copying+error+constraints’

To breed dogs, we hijack the mechanisms that nature uses:

- To get faster dogs / we select the fastest doggy parents
- We do ‘artificial selection’ .. Nature does ‘natural selection’

University of BRISTOL

Evolution in an Algorithm

We need to have a few elements:

1. A *population* of individuals, each represented by ‘genes’
2. A way to evaluate their fitness: a *fitness function*
3. A way to *select* the fitter individuals as parents
4. A way to introduce some *variety* in their offspring:
   - copying errors (‘mutation’)
   - mixing genes from two parents (‘sex’)
5. A *loop* over steps 2, 3, & 4 + a *stopping condition*

University of BRISTOL

Evolution in an Algorithm

We specify what we want evolution to find by defining a *fitness function*

- Analogous to an *objective function*, or *reward function* in Machine Learning or Reinforcement Learning
- A fitness function assigns a fitness score to one entire individual, based on its overall performance:
  - A Wing: $fitness_i = f(drag_i, weight_i, lift_i)$, e.g., $lift_i/(drag_i+weight_i)$
    - A Robot Cat: $fitness_i = 1/(1+num\_rats\_in\_my\_house)$

University of BRISTOL

Law of Unintended Consequences

- An unwritten assumption: Defining a good fitness function is easier than working out the solution yourself by hand
- But designing the fitness function can be difficult…
- It may be tricky to exploit our knowledge of the problem:
  - E.g., assigning a fitness boost to a chess player when it captures the queen may end up rewarding poor chess-playing strategies
- Fitness functions can encourage *degenerate* solutions:
    - E.g., car fitness = “don’t crash” => a car that never moves

University of BRISTOL

Further Reading/Watching

- *The Blind Watchmaker*, Richard Dawkins, Norton & Co., 1986.
- *Biomorphs*: Interactive Web App + Explanation + Simple Code https://www.defy.org/hacks/biomorphs/
- Karl Sims' *Genetic Images*: Explanation + Gallery + Papers, https://www.karlsims.com/genetic-images.html
- David Ha's *Neurogram*: https://blog.otoro.net/2015/07/31/neurogram/
- Video of Richard Dawkins' *Horizon* Episode (45 min.): http://www.dailymotion.com/video/x223a3n
    - "Methinks it is like a weasel", Biomorphs, Rechenberg, Holland..

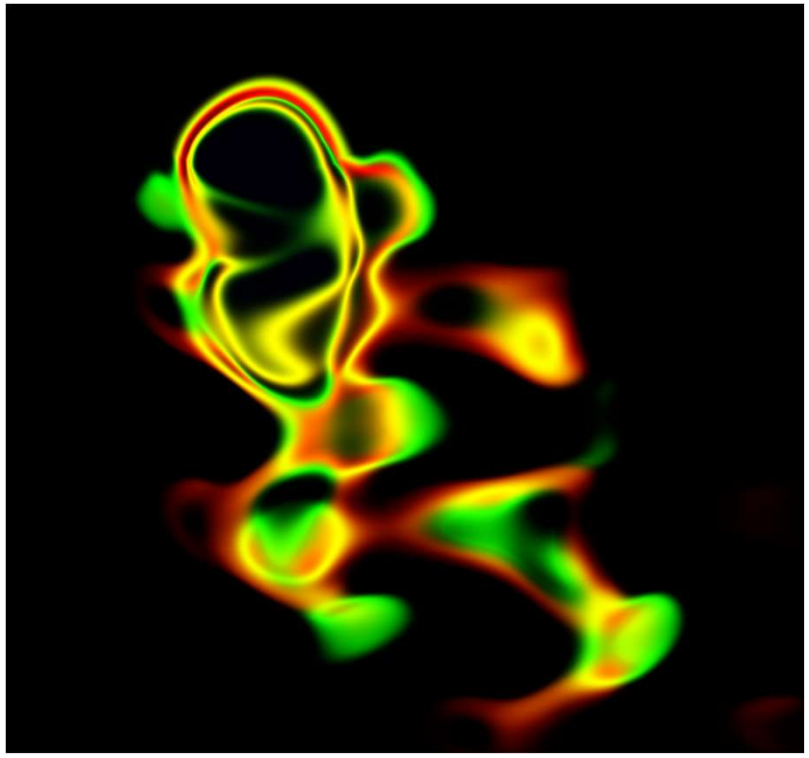
University of BRISTOL

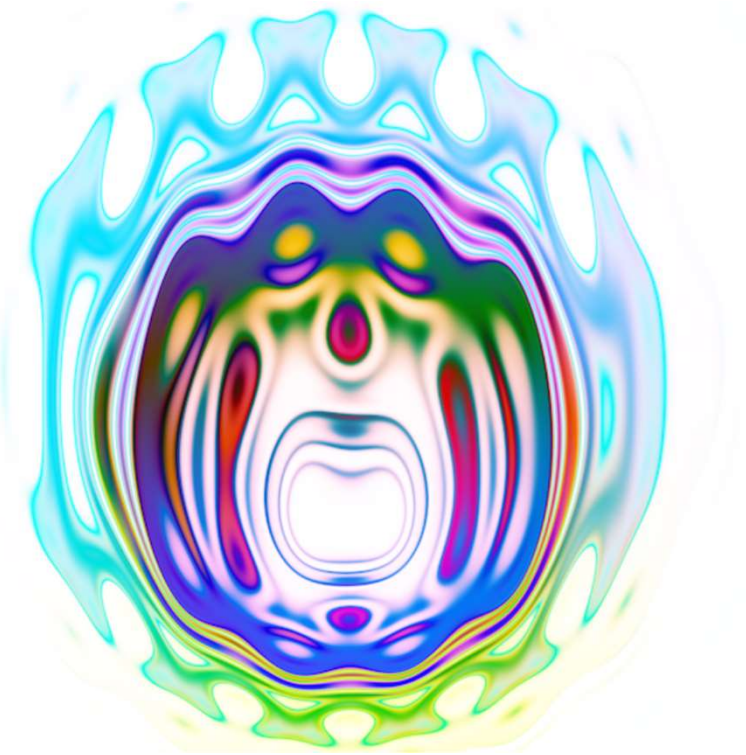
Karl Sims (1991)

https://www.karlsims.com/papers/siggraph91.html

University of BRISTOL

David Ha, Google Brain (2015)





https://blog.otoro.net/2015/07/31/neurogram/

University of BRISTOL

Applications: Optimization

- Some papers from last year (top page of Google Scholar):
  - *Automatically designing CNN architectures using the genetic algorithm for image classification,* Y Sun, B Xue, M Zhang, G Yen ..
  - *Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm*, E Ruiz, V Soto-Mendoza, AER Barbosa ..
  - *Extended genetic algorithm for solving open-shop scheduling problem*, AAR Hosseinabadi, J Vahidi, B Saemi, AK Sangaiah ..
  - *Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis*, GT Reddy, MPK Reddy, K Lakshmanna ..

University of BRISTOL

Applications: AI, ALife & Robotics

- Karl Sims (Blockies): https://youtu.be/mA8z0GndiYI
- Hod Lipson (GOLEM): https://youtu.be/qSI0HSkzG1E
- Hod Lipson (Walking Robot): https://youtu.be/iNL5-0_T1D0?t=273
- OpenAI (Hide & Seek): https://openai.com/blog/emergent-tool-use/
- Josh Bongard (Soft Robots): https://youtu.be/gXf2Chu4L9A
- F. Corucci (Swimming Robots): https://youtu.be/4ZqdvYrZ3ro
- Joachimczak et al. (Development) https://youtu.be/CXTZHHQ7ZiQ

University of BRISTOL

Example Questions

- Which of the following are/are not required for evolution: *selection*, *sex*, *metabolism*, *reproduction* *[1 mark]*
- Name three kinds of bio-inspired algorithm . *[1 mark]*
- Was Richard Dawkins' Biomorphs system an example of natural selection or artificial selection? Explain. *[3 marks]*
- Give one reason why Dawkins' "methinks it is like a weasel" example is a *good* way to introduce evolution as an algorithm and one reason why it is *not*. *[4 marks]*

University of BRISTOL

Thank you!

Metaheuristics

Seth Bullock

bristol.ac.uk

University of BRISTOL

What is Evolutionary Computing?

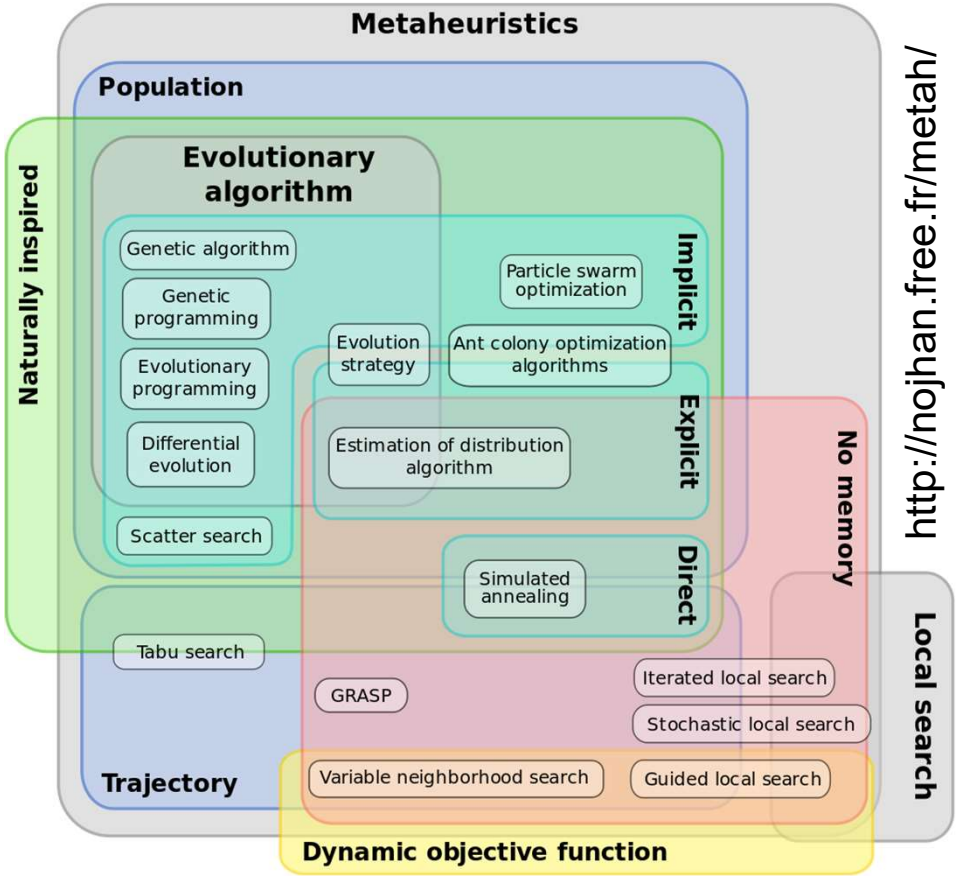
Evolutionary Computing (EC) techniques:

- a diverse collection of ‘nature-inspired’ techniques
- *heuristic*: not guaranteed to find the best solution
  - (the opposite of an ‘exact’ algorithm)
- *stochastic*: they use random number generators
  - (the opposite of ‘deterministic’ algorithms)
- intended for hard optimization problems (e.g., NP-hard)

University of BRISTOL

Metaheuristics

- EC techniques are examples of *metaheuristic* approaches.
  - A metaheuristic is a way to guide search
  - Some are nature inspired
  - Some are population based
  - Some build an explicit model of the problem as it's being solved



University of BRISTOL

EC Examples

There are several different kinds or flavours of EC algorithm:

- Genetic Algorithms (GA)
- Genetic Programming (GP)
- Evolutionary Strategies
- Differential Evolution
- Etc.

..often invented (quasi-)independently by different people.

University of BRISTOL

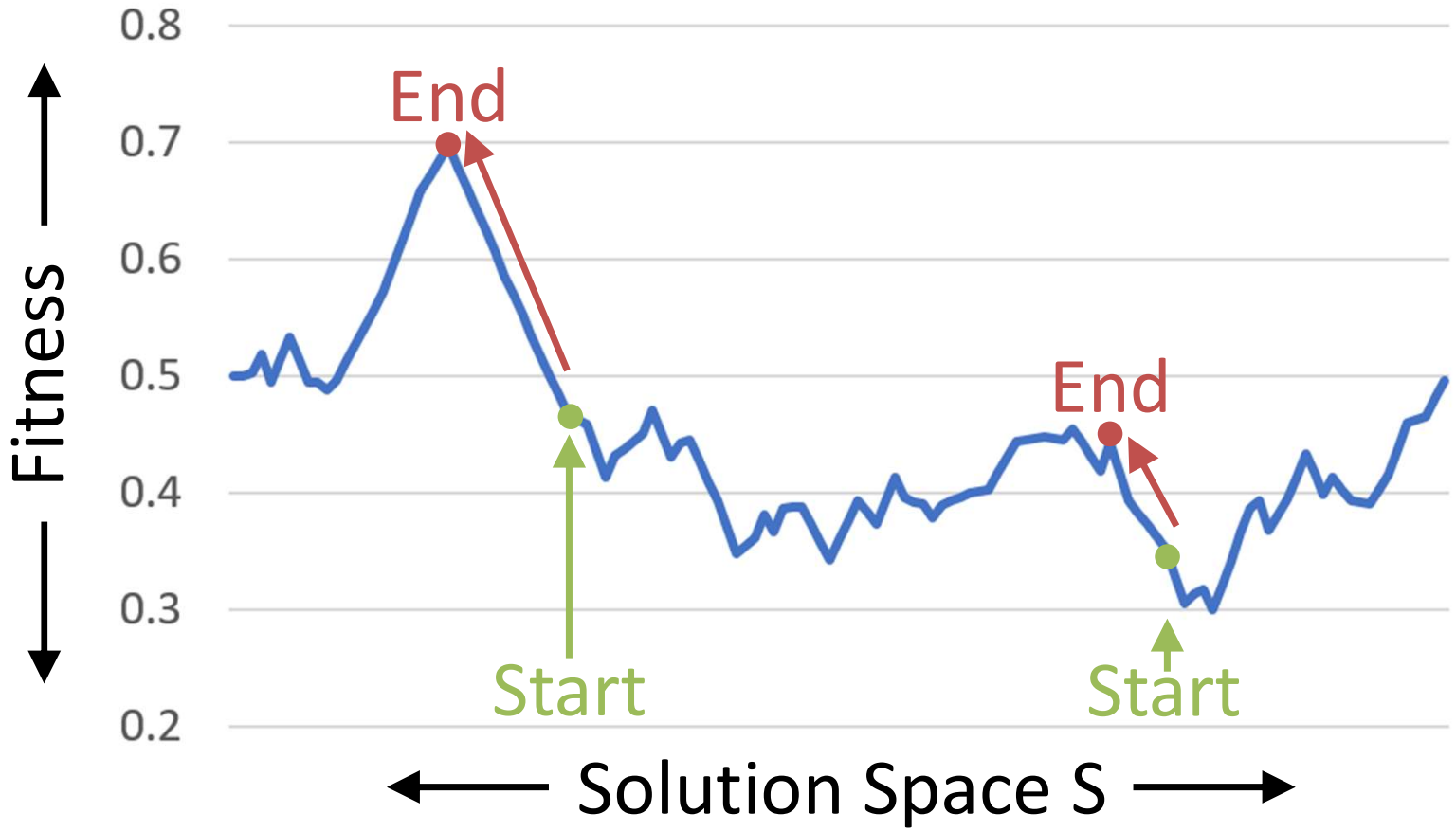
Individual-based Metaheuristics

In contrast to EC approaches, some metaheuristic approaches do *not* use a population of solutions; e.g.,

- Hillclimbing
- Simulated annealing
- Tabu search
- …

University of BRISTOL

Hill Climber



1. Make a random starting solution
2. Change it a little
3. If the new solution is better: keep it
   Else: discard it
4. If stuck: stop
   Else: go to 2

Consider: Is this a good diagram?

University of BRISTOL

Simple Hill Climbing

Repeatedly switch to the first better solution found in the local neighbourhood of the current solution

```
start with any solution p chosen from solution space S
repeat:
      choose_a_better_neighbour(p) => q
      if q is None return p, else q => p

def choose_a_better_neighbour(p):
      for each neighbour q of p:
            if q is better than p:
                  return q
      return None
```

Consider:

- *How do we pick the first solution p from S?*
- *What counts as the neighbourhood of a solution?*
- *In what order should we consider p's neighbours?*

University of BRISTOL

Steepest Ascent Hill Climbing

Repeatedly switch to the best solution found in the local neighbourhood of the current solution

```
start with any solution p chosen from solution space S
repeat:
      choose_the_best_neighbour(p) => q
      if q is p return p, else q => p

def choose_the_best_neighbour(p):
      p => fittest
      for each neighbour q of p:
            if q is better than fittest: q => fittest
      return fittest
```

Consider:

- *Does the order in which we consider p's neighbours still matter?*

University of BRISTOL

Simple Tabu Search

Hillclimb, but don't consider solutions on a (finite) list of previously visited solutions

```
start with any solution p chosen from solution space S
p => best_so_far ; [p] => tabu_list
repeat:
      best(neighbours_of_p_not_in_tabu_list) => p
      if p is better than best_so_far: p => best_so_far
      add p to tabu_list
      if tabu_list too long:
            remove oldest item
      if time_to_stop: return best_so_far
```

Consider:

- *What's the significance of the tabu_list max length?*

University of BRISTOL

Simulated Annealing

Hillclimb, but tolerate moves to worse solutions with a probability that is initially high, but decreases as more steps are made

```
start with any solution p chosen from solution space S

for step from 1 through max_step:
      pick_a_random_neigbour(p) => q
      if allow(fit(p), fit(q), temp(step)):
            q => p
return p

    def allow(p, q, T):
          return True if q>p or rand(1)<=exp(-(q-p)/T), else False
```

Consider:

- *The 'schedule' temp(step) needs to be defined*
- *It should return a high temperature for step 1*
- *And reduce it gradually as more steps are taken*

University of BRISTOL

The Explore/Exploit Tradeoff

- Recall: Breadth-first search *explores*, while depth-first *exploits*.
- Hill Climbers *exploit* current local knowledge of the search space – not even interested in fully exploring the local neighbourhood.
- Steepest Ascent Hill Climbers do explore the local neighbourhood fully before *greedily* exploiting the best local next step.
- Tabu is a little more *exploratory* than regular hill climbing. It keeps a record of solutions that have not year been fully exploited.
- Simulated Annealing starts off more *exploratory* (at high temps) but gets more *exploitative* over time (as temp falls).

University of BRISTOL

Going Populational

- All the algorithms that we've looked at work with only *one* hill climber moving across the solution space.
- By contrast, Genetic Algorithms maintain a *population* of solutions to help balance the explore/exploit trade-off.
- A population of solutions allows exploration to take place in *multiple* parts of the solution space.
- Competition & crossover allow success in one part of the space to be exploited by the rest of the population

University of BRISTOL

Example Questions

- What's the difference between a stochastic algorithm and deterministic algorithm? *[1 mark]*
- Why does the temperature of a simulated annealing algorithm start off high and get lower over time? *[2 marks]*
- If a hill-climber is stuck, what can you infer about the solution that it has found? *[3 marks]*
- Explain the explore/exploit trade-offs made by best-first search, tabu search & simulated annealing. *[8 marks]*

University of BRISTOL

Thank you!

The Genetic Algorithm

Seth Bullock

bristol.ac.uk

University of BRISTOL

A Standard GA

- Genetic algorithms share the same basic form:
  - Repeatedly: assess, breed, mutate a population of solutions
- But different types of GA vary in many different ways:
  - Selection scheme, genetic operators, population structure, etc.
  - ‘Steady state’ GA vs. ‘generational’ GA
  - + Many special tricks of various kinds...
- Here we present a very simple version
    - (and mention some of the most frequent variants)

University of BRISTOL

Terminology

- *Population* – the set of individual solutions that the GA is acting on
- *Individual* – a member of the population
- *Genotype* – a string of symbols from some alphabet that encodes a particular solution (sometimes: *Genome* or *Chromosome*)
- *Phenotype* – the actual solution encoded by a genotype
- *Genotype-Phenotype Mapping* – analogous to development
- *Genes* (also *Loci*) – chunks of genome, each taking a value: "*Allele*"
- *Fitness* – the value or quality of an individual solution phenotype
    - *Fitness Function* – returns the fitness of a solution

University of BRISTOL

Terminology

- *Selection* – choosing which current individuals reproduce
- *Parents* – individuals selected to reproduce
- *Offspring* – the new individuals that result from reproduction
- *Crossover* – the recombination of alleles from multiple parents
- *Mutation* – replacing offspring alleles with random alternatives
- *Fitness Landscape* – an 'evolutionary search space' organising all possible solutions according to the neighbourhood relationships that result from the GA's Genotype Structure & Genetic Operators, with landscape 'altitude' set by each solution's Fitness.

University of BRISTOL

Simple Genetic Algorithm

- A Simple GA:

```
initialise() => population
repeat:
    evaluate(population)
    select(population) => parents
    breed(parents) => offspring
    offspring => population
    if timed_out or good_enough:
        return best(population)
```

- Compare with a Hillclimber

```
initialise() => individual
repeat:
    evaluate(neighbours)
    select(neighbours) => chosen

    chosen => individual
    if timed_out or stuck:
        return(individual)
```

University of BRISTOL

Generate Initial Population and Evaluate

- We start with a population of individual random genotypes
  - Each is a random string of symbols from the genetic ‘alphabet’
  - A typical (classic) GA alphabet: {0,1} – binary genotypes
- To evaluate an individual we use the fitness function:
  - How well does the robot specified by the genotype behave?
  - How good is the wing design specified by the genotype?
  - How good is the exam timetable specified by the genotype?
  - How successful is the chess strategy specified by the genotype?

University of BRISTOL

Genotype-Phenotype Mapping

- The genotype is just a way to write down a solution:
  - #1: [Thin, Cheese, NoTomato, NoOlives, NoAnchovy]
  - #2: [Deep, Cheese, Tomato, Olives, Anchovy]
- To allocate fitness to genotypes, we must decode them.
- We must build ('develop') the *phenotypes* they encode:
- The *genotype-phenotype mapping* characterises this 'developmental' encoding relationship…

University of BRISTOL

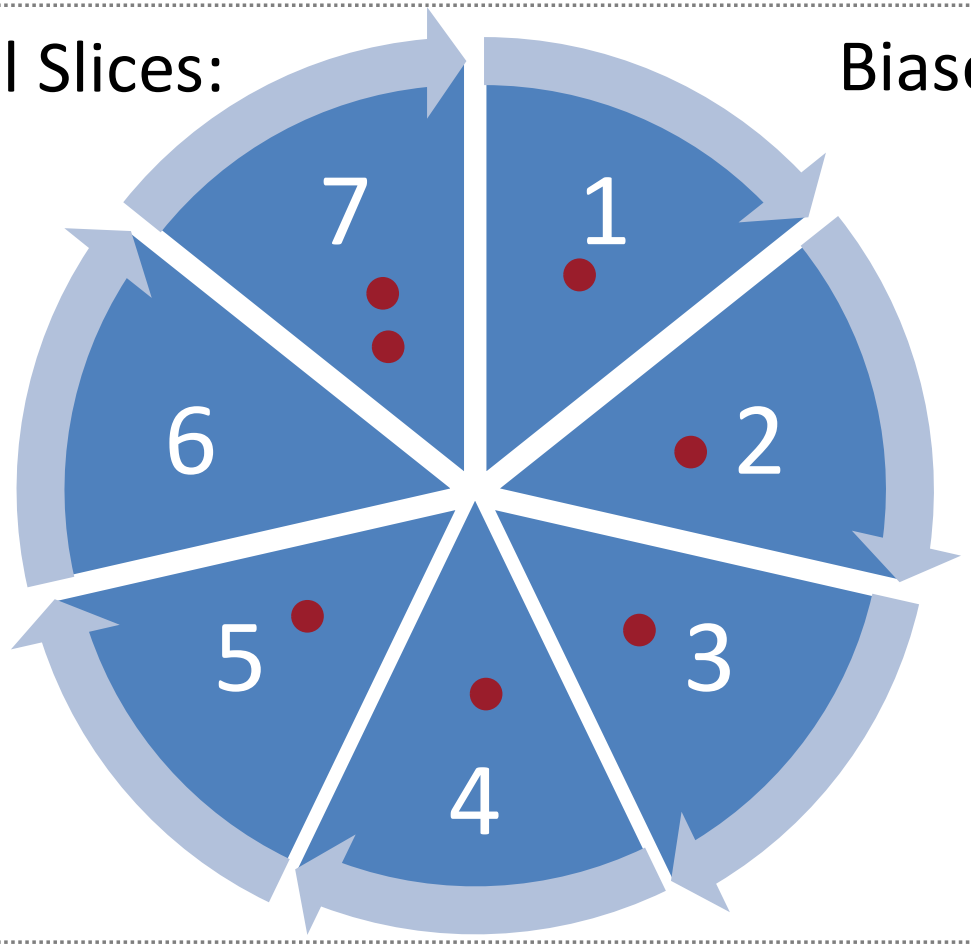
Selection

- A key idea is that we should tend to select the *fitter* individuals from the current population to become parents
- We can pick parents with a chance proportional to fitness:
  - $p_i \propto f_i / S$ where p_i is the chance that we pick individual i
  - S is the sum of fitnesses in the current population: $S = \sum_{j=1}^{N} f_j$
  - N is the number of individuals in the current population
- One way of doing this is using 'roulette wheel' selection:
    - Spin a wheel where: probability of landing in slice i = p_i

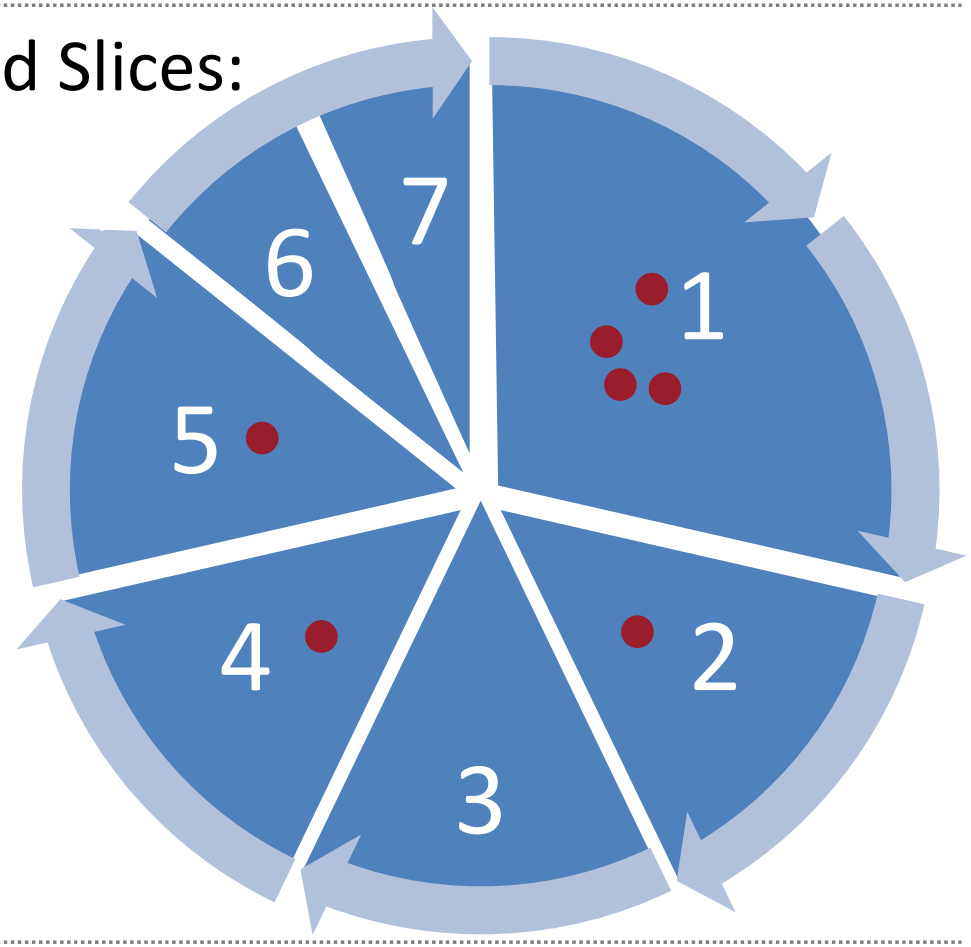
University of BRISTOL

Roulette Wheel Selection

Equal Slices:



Biased Slices:



University of BRISTOL

Roulette Wheel Selection

Run a roulette wheel that selects element i *with probability* p_i

```
start with a population: N individuals, each with a fitness score

sum(fitnesses(population)) => S
rand(0,S) => roll

0 => running_total
loop i from 1 to N:
      running_total + fitness(i) => running_total
      if running_total >= roll:
            return(i)
```

Consider:

- *Is it worth sorting the population by fitness before using the roulette wheel to pick N parents?*

University of BRISTOL

Rank-Proportionate Selection

- Alternatively we could pick each parent with a chance proportional to the *rank* of their fitness in the population
- One way of doing this is using 'tournament' selection:
  - Sample *k* (unique) individuals at random from the population
  - Pick a winner (e.g., the one with highest fitness) to be a parent
- To find *N* parents, run *N* independent tournaments.

Consider:

- *How is selecting on fitness* rank *different from selecting on* fitness*?*

Consider:

- *What is happening if k=1?*
- *What is happening is k=N?*

University of BRISTOL

Reproduction

- Having selected parents, we reproduce them:
  - we make a copy of their genes and put it into the next generation of the population
- Genetic operators are applied during reproduction:
  - Mutation: each offspring gene is replaced by a random allele with probability *m*
  - Crossover: the offspring genotype inherits genes from multiple parents
- Mutation is almost always used. One mutation per offspring is a typical rate.
- Crossover is sometimes not used at all ('asexual' reproduction).
- …or is only applied to some offspring (e.g., it occurs with probability *c*)
- Elitism: the fittest current individual is copied into next generation perfectly

Consider:

- *What if all parents were just copied perfectly?*

University of BRISTOL

Types of Crossover

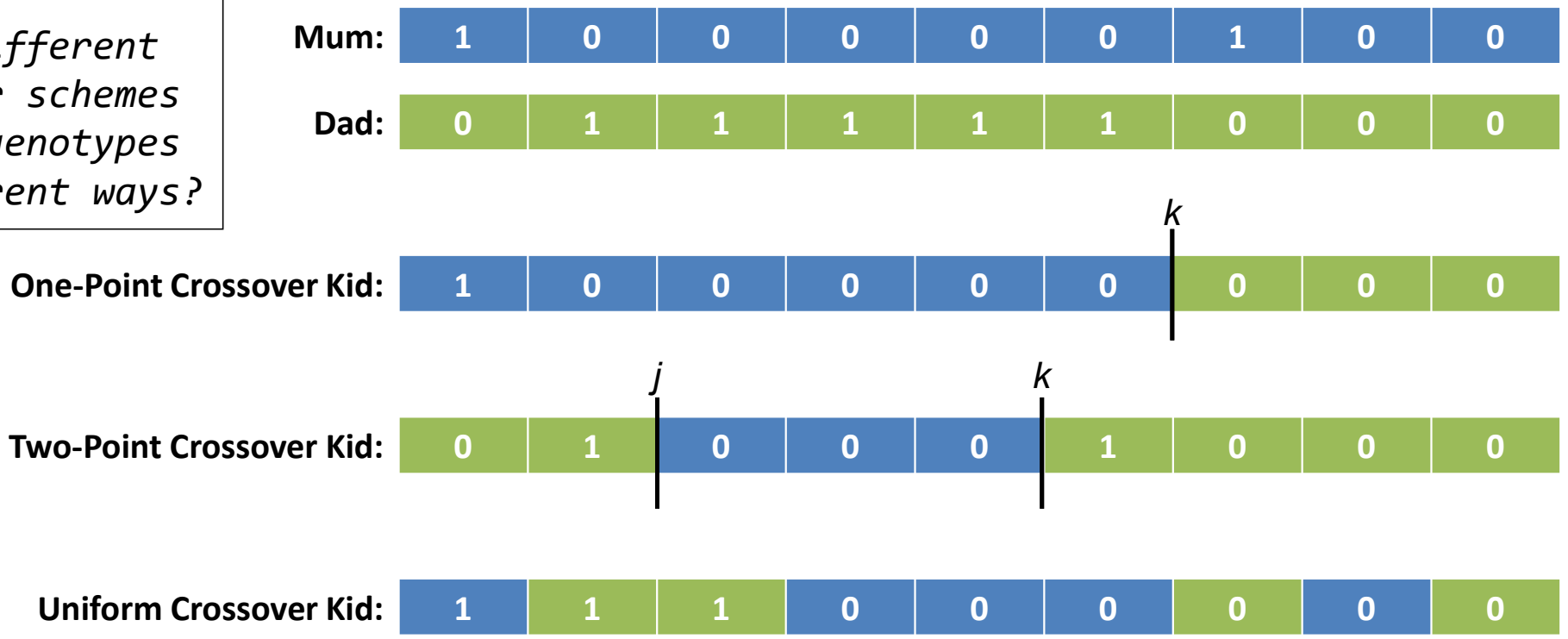
- One-Point: pick a point along the genotype, $k \in [0, L]$
  - The offspring inherits the first *k* genes from their mother's genotype and the remainder from their father's genotype
- Two-Point: pick two points along the genotype, $j, k \in [0, L]$
  - The offspring inherits genes *j* thru *k* from their mother's genotype and the remainder from their father's genotype
- Uniform: merge the genes from both parents at random
    - Each of the offspring's genes, is inherited from their mother with probability 0.5, else from their father.

University of BRISTOL

Crossover Schemes

Consider:

- *How do different crossover schemes disrupt genotypes in different ways?*



University of BRISTOL

Generational GAs vs Steady State GAs

- Typically the GA population size, *N*, is held constant...
- Two common ways to ensure this:
  - Generational Reproduction: ...we continue selecting and reproducing parents until we have a *N* new offspring. Then we discard the old population and replace it with the *N* new individuals.
  - Steady-State Reproduction: …as soon as we have generated *one* new offspring, we pick a member of the current population (either at random or biased toward the unfit) and kill it, replacing it with the new offspring.
- There are many different wrinkles on these basic schemes:
    - E.g., only ever let a new offspring displace a less fit individual

University of BRISTOL

Stopping

- Two simple stopping conditions:
  1. We've found the perfect solution, i.e., a genotype that achieves maximum fitness
  2. We've run out of time, i.e., our generation counter has reached max_generation
- A more subtle stopping condition:
  3. We think the GA has run out of steam and things are as good as they are going to get
- Checking for conditions 1 and 2 is easy. How do we check for condition 3?
- Fitness scores have *stagnated* – no improvement for many generations:
  - Current "best" genotype is (very) old…
  - The population is strongly converged…

Consider:

- *Might the population still be exploring new solutions?*
- *How could we know for sure?*

University of BRISTOL

Genome Structure

- Remember: Each genotype is made of symbols from the genetic 'alphabet'
  - {0,1} – bit strings; {A, C, G, T} – nucleic acids; {A, B, C, …Z} sentences
  - Or maybe parameters of some kind: {<integers>}, {<floats>}
  - Note that choice of alphabet has implications for mutation and crossover
  - Mutation = a bit flip; or a random letter; or a random perturbation
- And there could be constraints on some genes:
  - `pizza_radius` must be *positive* and *less than* `oven_width`
  - `num_robot_legs` must be *even*;
    - a 3-bit encoding of `exam_day` leaves `101`, `110`, and `111` unused?
    - illegal genotypes must be discarded... causing search *biases*

University of BRISTOL

Simple Genetic Algorithm (Recap)

- The Simple GA:

```
initialise() => population
repeat:
    evaluate(population)
    select(population) => parents
    breed(parents) => offspring
    offspring => population
    if timed_out or good_enough:
        return best(population)
```

- Simple Example

```
# generate N random bit strings
# employ a generational GA
#  fit. func. assigns f_i ϵ [0,1]
#  tournament selection, use k=3
#  bitflip mutation & 1-pt xover
#  elitism on: always copy best
#  gen > max_gen or fitness = 1
```

University of BRISTOL

Example Questions

- Name two different kinds of cross-over operator. *[1 mark]*
- Uniform crossover of two L-bit binary genotypes can result in how many different offspring genotypes? *[2 marks]*
- What is the difference between roulette-wheel selection and tournament selection? *[4 marks]*
- Define a good fitness function for evolving a university time-table schedule. Explain your answer. *[8 marks]*

University of BRISTOL

Thank you!

University of BRISTOL

Fitness Landscapes

Seth Bullock

bristol.ac.uk

University of BRISTOL

Fitness Landscapes

Seth Bullock

bristol.ac.uk

University of BRISTOL

Fitness Landscapes

This time:

- Visualizing search problems as fitness landscapes
- Kinds of landscape (unimodal, multimodal, deceptive...)
- How search operators interact with fitness functions
- How search moves across landscapes
- The need to both explore new areas and exploit existing knowledge while searching

University of BRISTOL

Search Space

- The set of possible genotypes is the GA's search space, *S*
- Given genotypes of length *L* from an alphabet of size *A*:
  - The space has *L* dimensions which each have *A* possible values
  - There are therefore A^L possible genotypes in *S*
- Each genotype is a *point* in the search space
- Genotypes are 'neighbours' if they differ by one mutation
- The *structure* of a search space influences how challenging it is for an algorithm to search it

University of BRISTOL

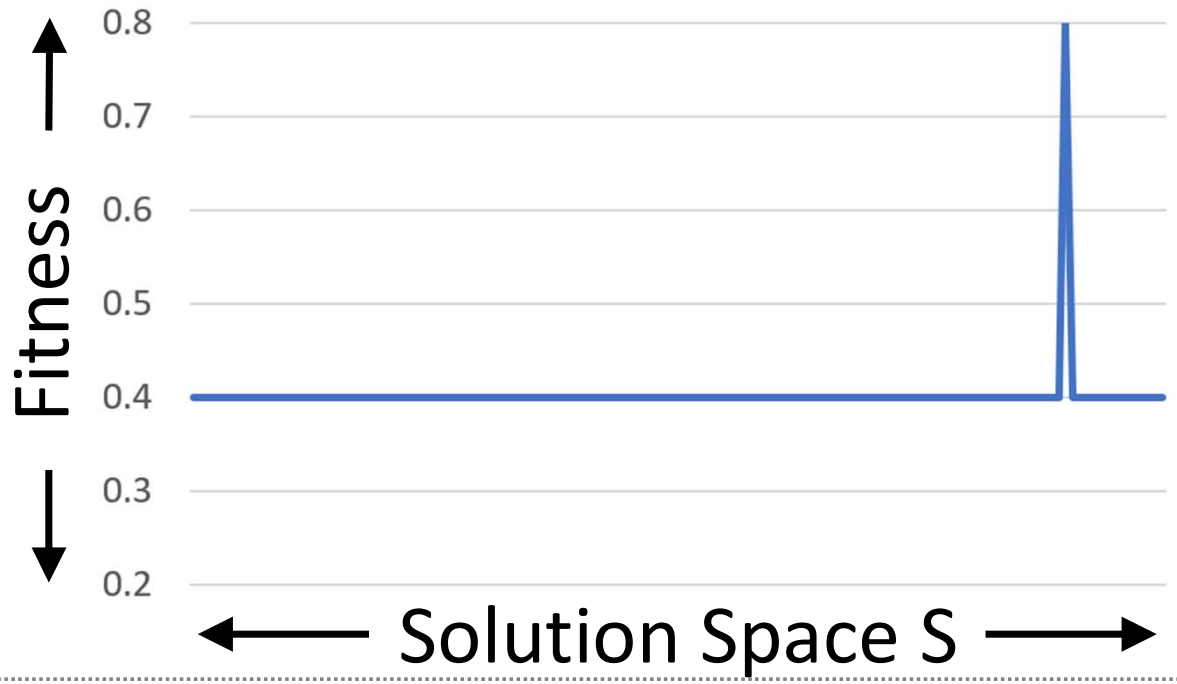
Fitness Landscapes

- A search space can be visualized as a *fitness landscape*
  - A plot of the fitness of all possible genotypes: fitness => height
  - Called a landscape because it can have peaks, valleys, ridges...
  - Introduced by biologist Sewall Wright in 1932
- We can think of search as movement across the landscape
  - A population of points move in discrete steps and jumps
  - The shape of the landscape influences this movement

University of BRISTOL

Needle-In-A-Haystack Landscape

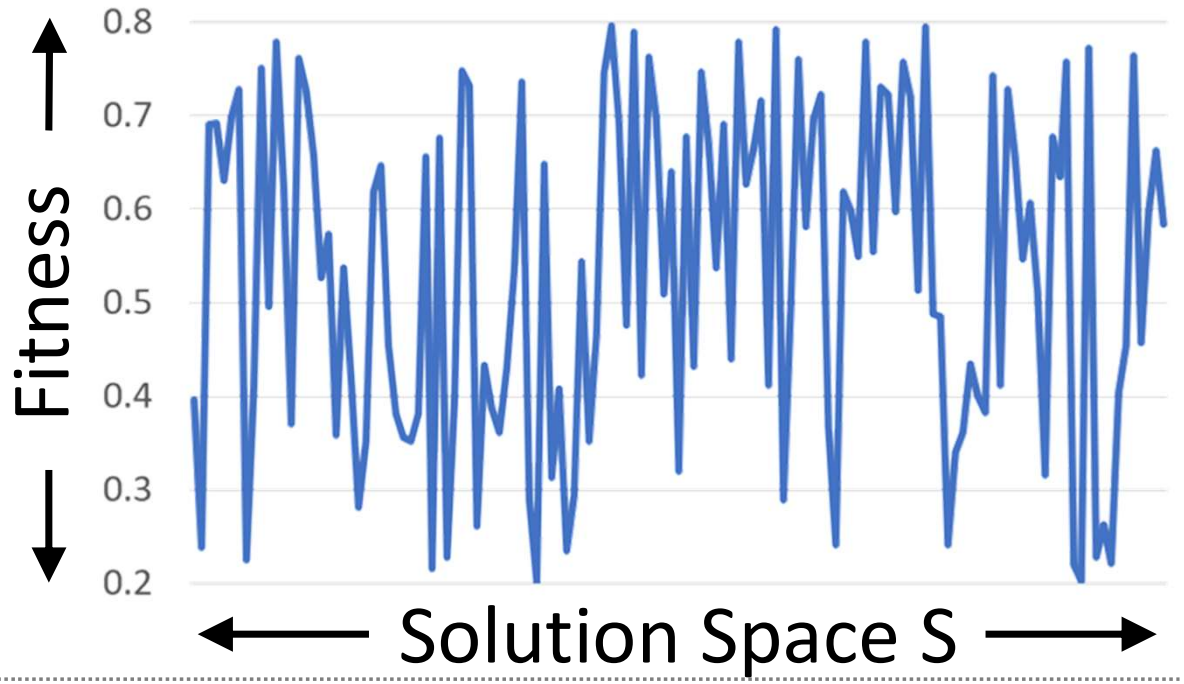
- Suppose all genotypes except one have the *same* low fitness
- This kind of problem is *hard* for any search algorithm:
  - The fitness function is very unhelpful; it has no structure to exploit
  - The fitness of suboptimal genotypes tells the algorithm no information about where the optimal genotype is



University of BRISTOL

Random Landscape

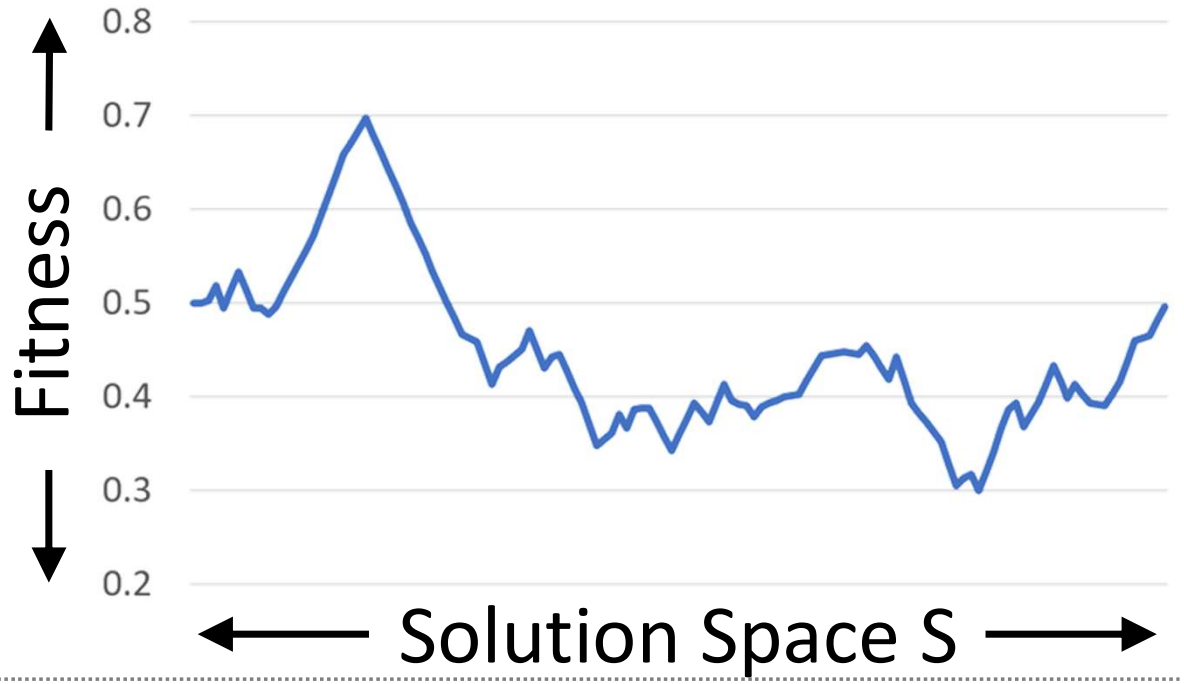
- Suppose all genotypes had a *randomly* assigned fitness
- Again, this problem is *hard* for any search algorithm:
  - The fitness function is still very unhelpful; it still has no structure to exploit
  - The fitness of suboptimal genotypes still tell the algorithm no information about the optimal genotype



University of BRISTOL

A Nicer Landscape

- Luckily, real-world problems do tend to have useful structure
- The algorithm's job is to be able to exploit this structure:
  - Local smoothness, i.e., nearby points have similar fitness = *correlation structure*
  - Ideally, higher peaks are also broader = *non-deceptive*
  - The problem can be divided into parts = *decomposable*



University of BRISTOL

Search Neighbourhoods

- What does ‘local’ mean in the term ‘local smoothness’?
  - It depends on how we represent and search the space, *S*
- We could say two points, *a* and *b*, in *S* are ‘local’ to each other if the ‘distance’ between their genotypes is beneath a threshold
  - What distance measure to use? Hamming distance? Euclidian distance?
- Alternatively we could define the ‘distance’ between *a* and *b* in terms of how easy it is for the algorithm to move between them.
  - *a* and *b* are *neighbours* if *a* can reach *b* in one genetic operation
    - (e.g., a single mutation)

University of BRISTOL

Thank you!

University of BRISTOL

Fitness Landscapes II

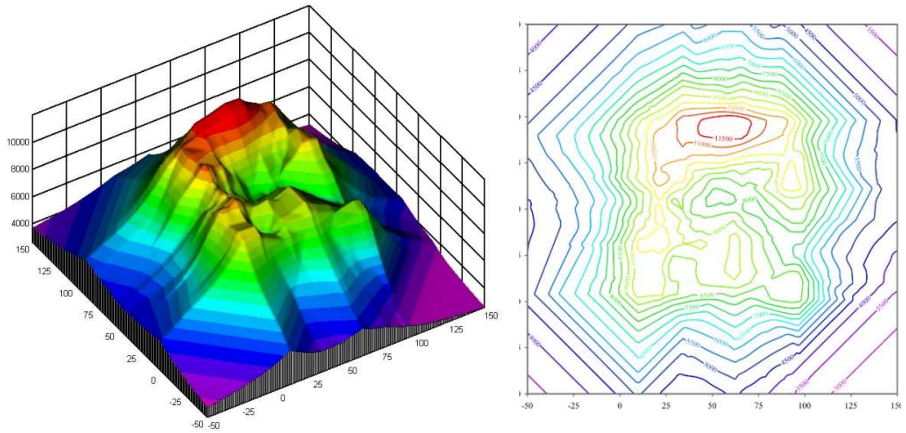
Seth Bullock

bristol.ac.uk

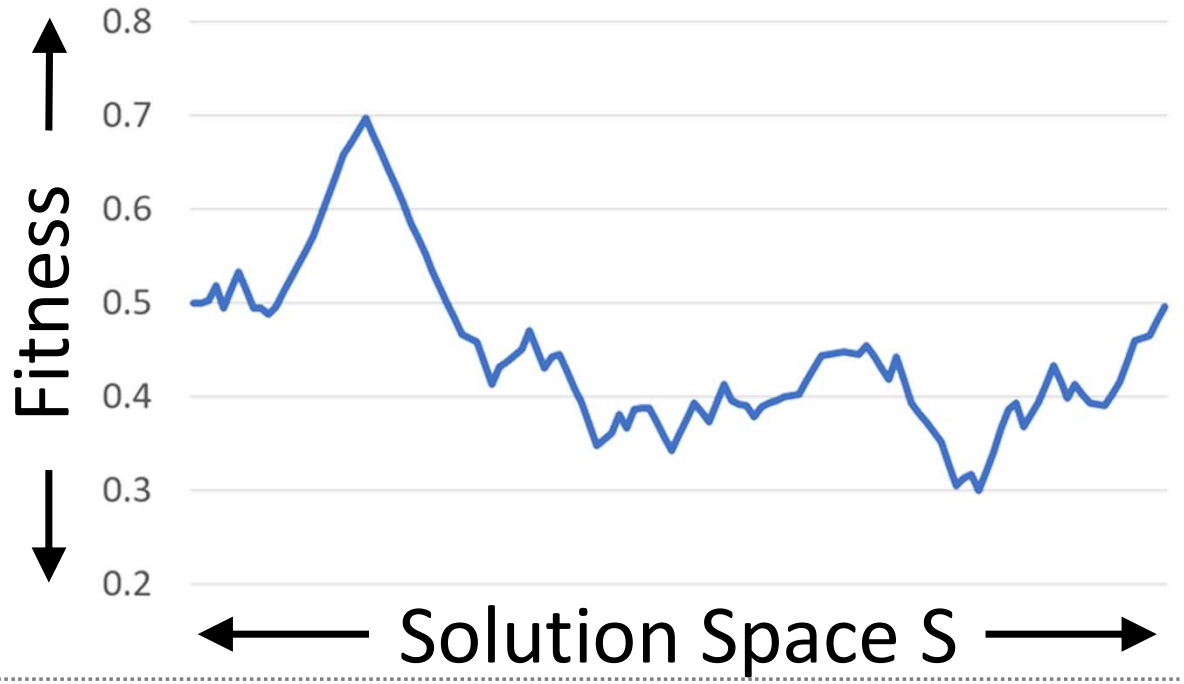
University of BRISTOL

Visualizing Neighbourhoods

- So far we have used a very poor diagram of a search space
- It represents *S* as a *line*; but *S* is typically *high-dimensional*
- Even a 2-D *S* is challenging:



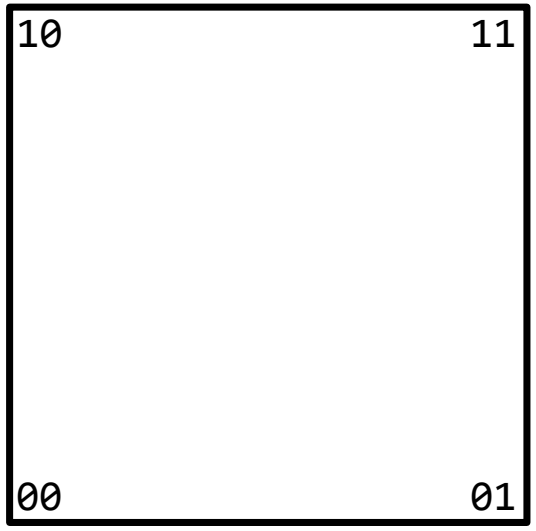
https://en.wikipedia.org/wiki/Contour_line



University of BRISTOL

Visualizing Neighbourhoods

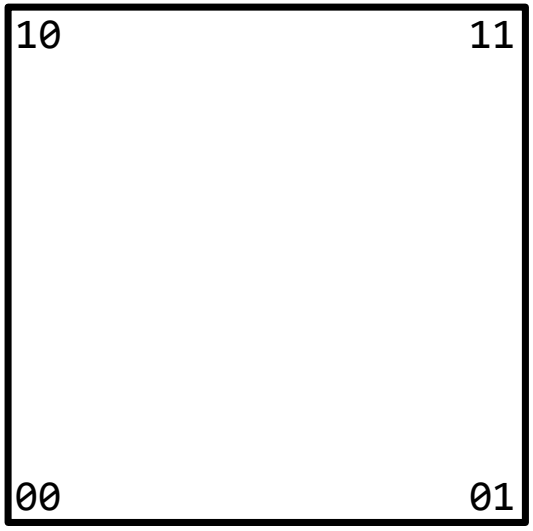
- In fact, it's even worse as many GAs search *discrete* spaces:
- Consider a binary genotype of length 1 or 2…

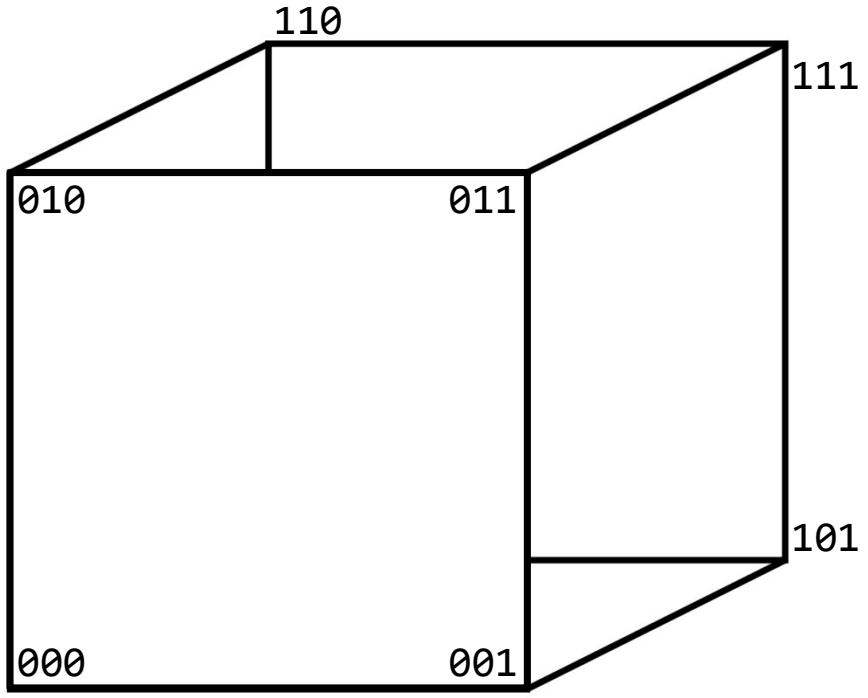


University of BRISTOL

Visualizing Neighbourhoods

- In one, two and three dimensions we are reasonably ok…

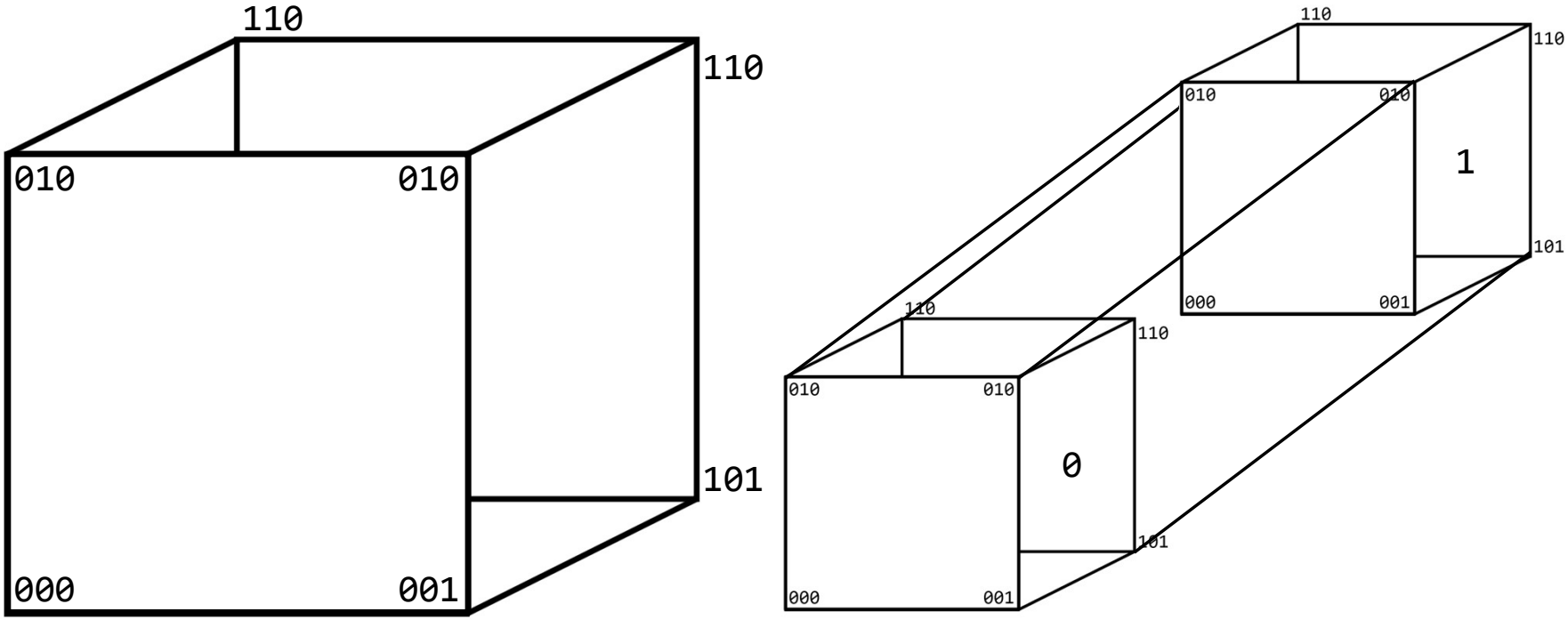




University of BRISTOL

Visualizing Neighbourhoods

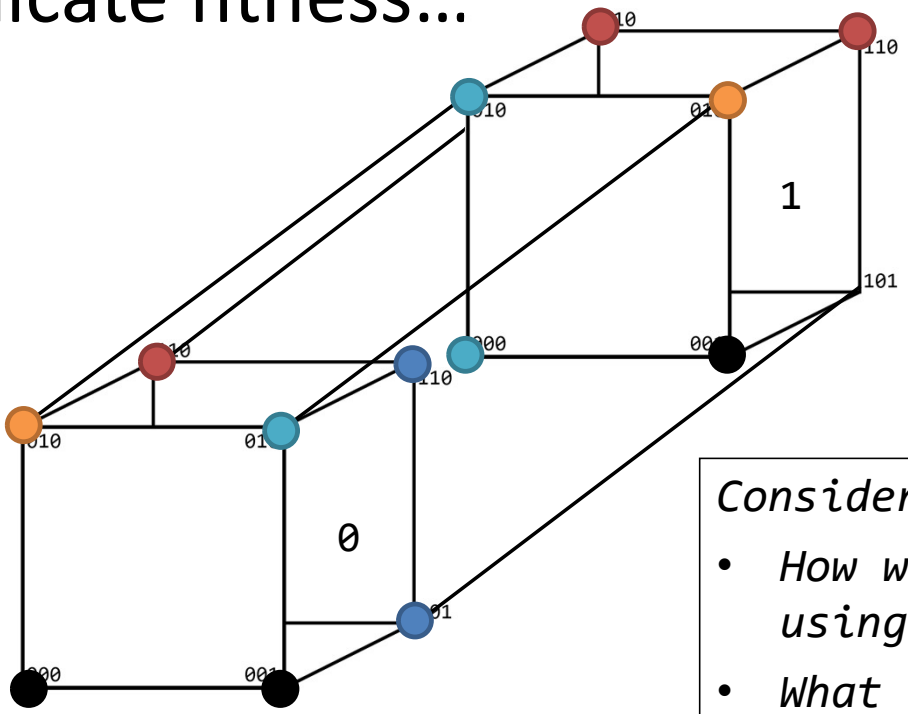
- …but a solution space of all length-*L* bit-strings implies that genotypes are located on an *L*-dimensional binary hypercube…



University of BRISTOL

Visualizing Neighbourhoods

- …and each visualized point also needs an extra dimension to indicate fitness…



- Comprehending the structure of real search spaces is challenging!

Consider:

- *How would this diagram change if we were using a ternary alphabet: {-1,0,1}*
- *What about genes with integer alleles?*

University of BRISTOL

Landscape Metaphors

- A real landscape is three-dimensional, continuous, defined, relatively smooth, and static or changing very slowly.
- But fitness landscapes don't share these properties.
- This means we have to be careful when we use "landscape" language to describe search spaces:
  - Peaks
  - Valleys
  - Ridges
  - Basins
  - Plateaus
  - Smoothness
  - Ruggedness
  - Climbing
  - Drifting

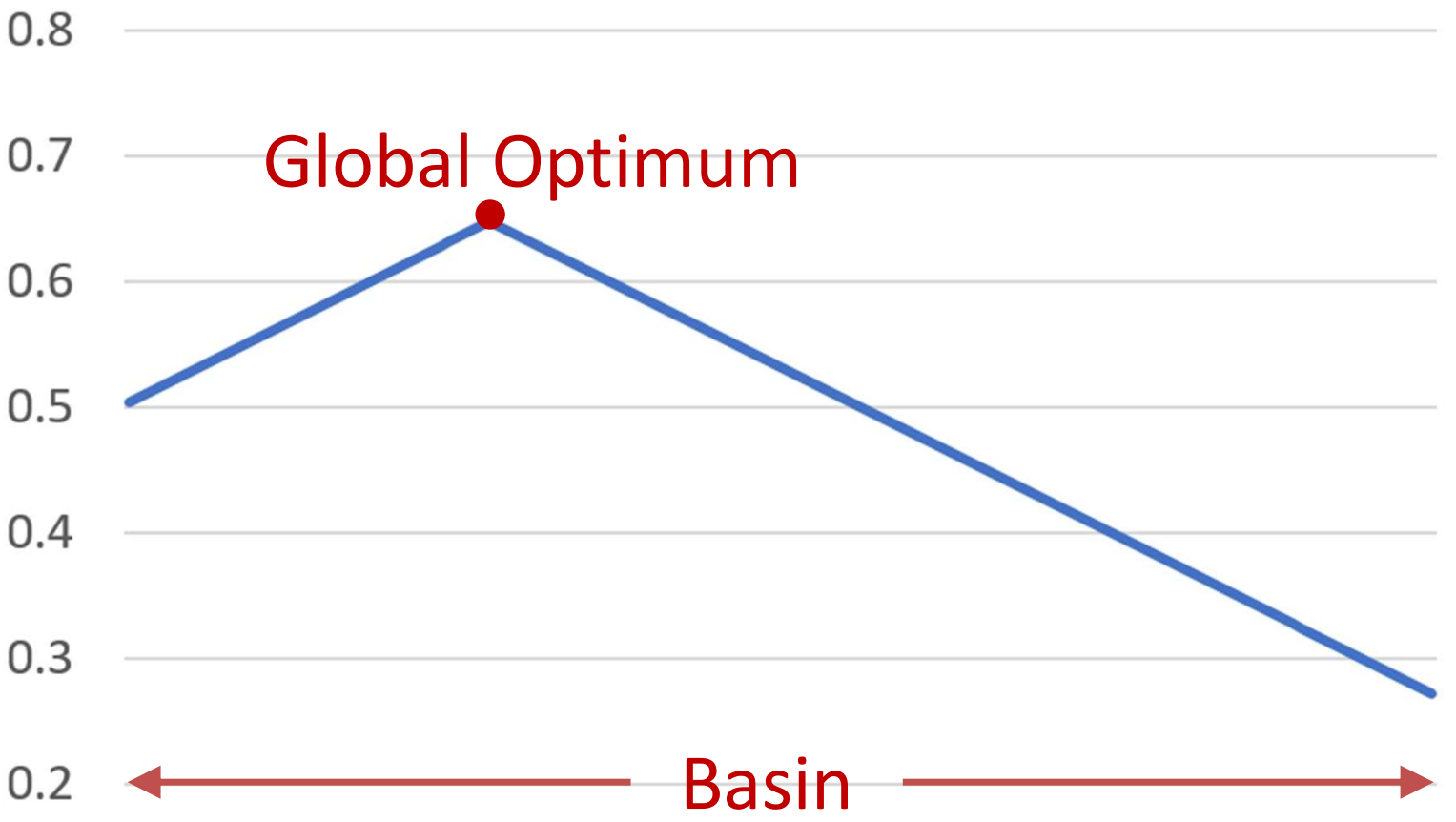
Modality

- A landscape with one global optimum solution (i.e., it has no local optima) is called *unimodal*
- A landscape with two optima is called *bimodal*
- A landscape with many optima is called *multi-modal*
- A set of points that lie on hill-climbing routes that terminate at one local optimum are said to lie within the *basin of attraction* of that local optimum.
    - A unimodal landscape has *one* basin of attraction

University of BRISTOL

Unimodal

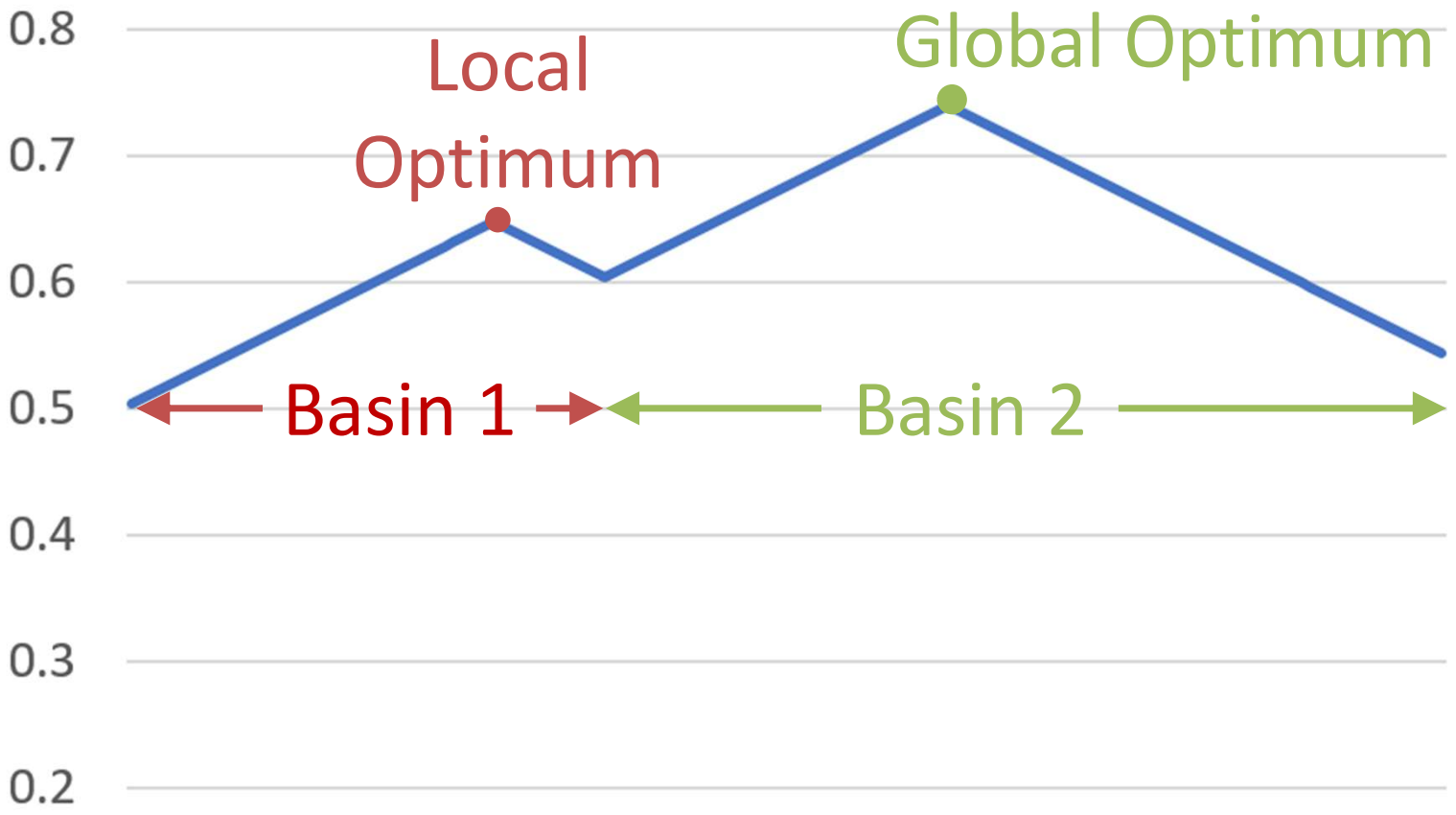
- Trivial problem
- EC is overkill
- Each gene has a *best* allele, independent of the alleles at all other genes



University of BRISTOL

Bimodal

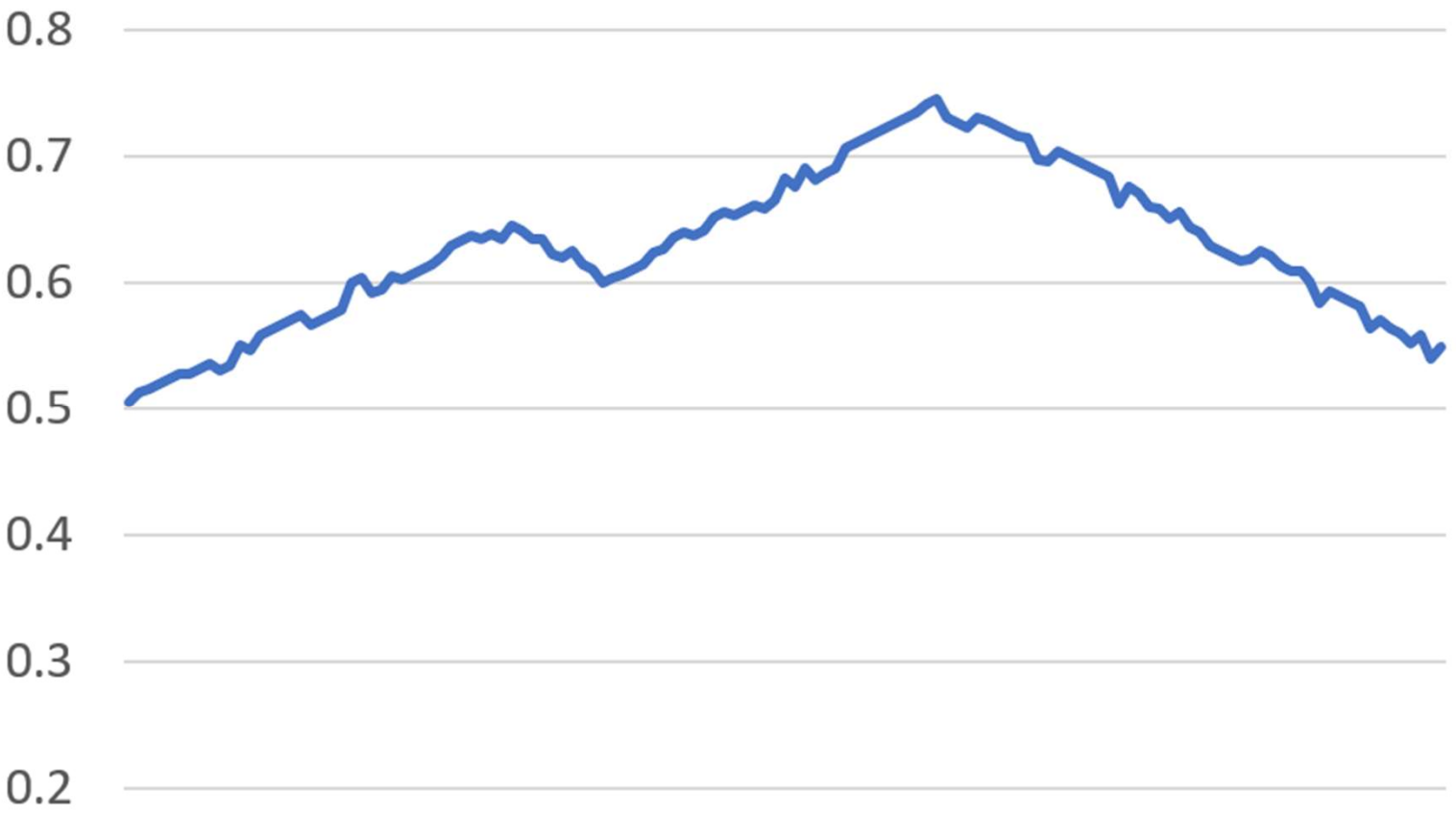
- The best allele for some genes *depends* on the alleles at some other genes.
- GA can't know which basin its population is in.



University of BRISTOL

Multimodal?

- Many basins of attraction for many optima.
- But isn't this landscape still bimodal really?
  - Depends on the algorithm…

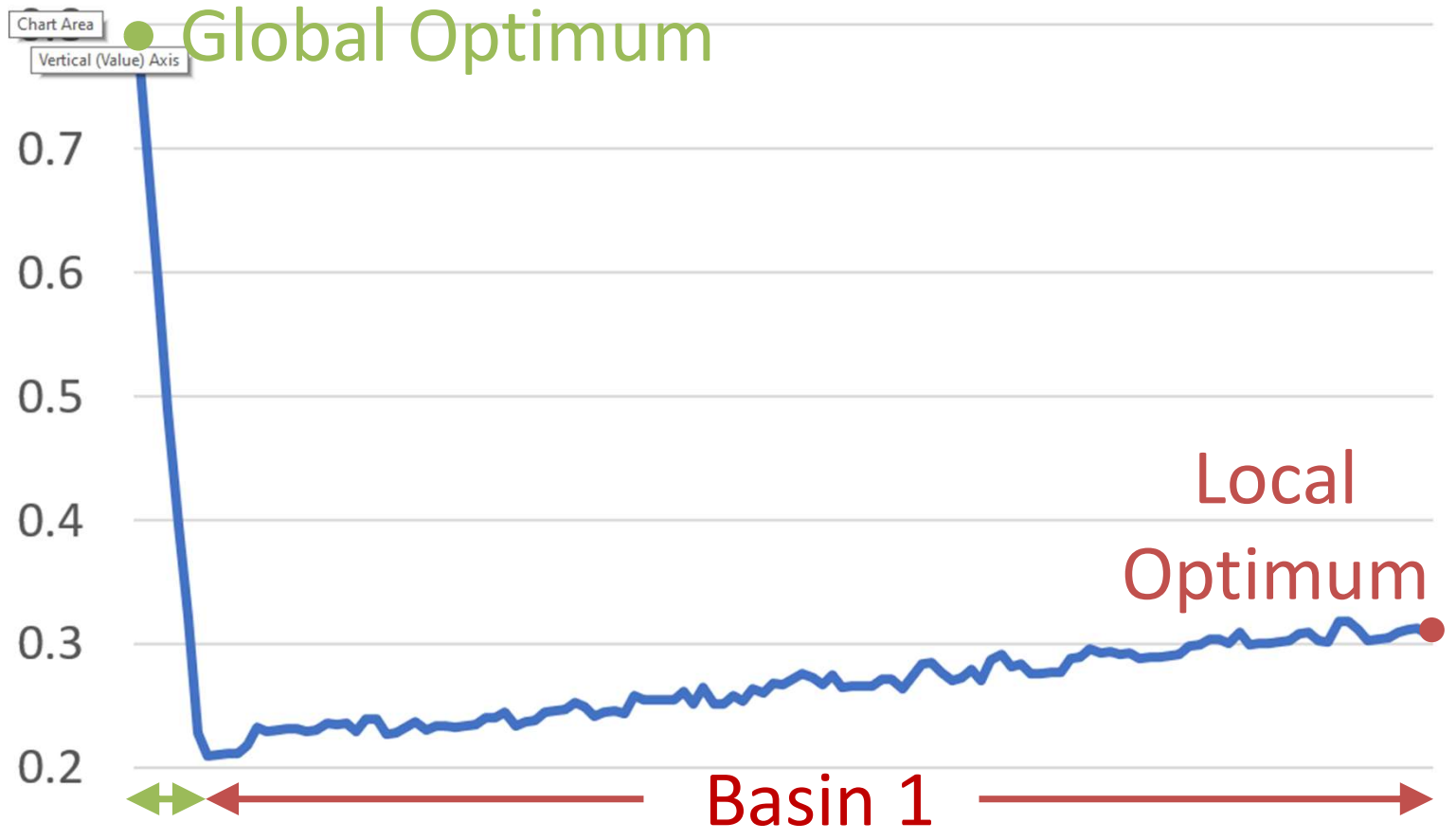


University of BRISTOL

Deception

For a deceptive landscape:

- The best optima have small basins.
- Consequently, global optima are hard to find



University of BRISTOL

Example Questions

- What size is the search space if genotypes each have G genes and each gene can be one of A alleles? *[1 mark]*
- Consider the search landscape drawn below. Which of the following properties does it have *[1 mark each]*
  - Neutral, Global, Deceptive, Unimodal, Rugged, Coevolutionary
- Tom's GA keeps getting stuck on local optima. He thinks that doubling the mutation rate might be a good idea. Is it? Explain your answer. *[5 marks]*

University of BRISTOL

Thank you!

University of BRISTOL

More Advanced GA Concepts

Seth Bullock

bristol.ac.uk

University of BRISTOL

Contents

This lecture covers a number of Evolutionary Computation concepts:

- Premature Convergence
- Neutrality
- Quasi-Species
- Epistasis
- Problem Modularity
- No Free Lunch
- Evolvability

University of BRISTOL

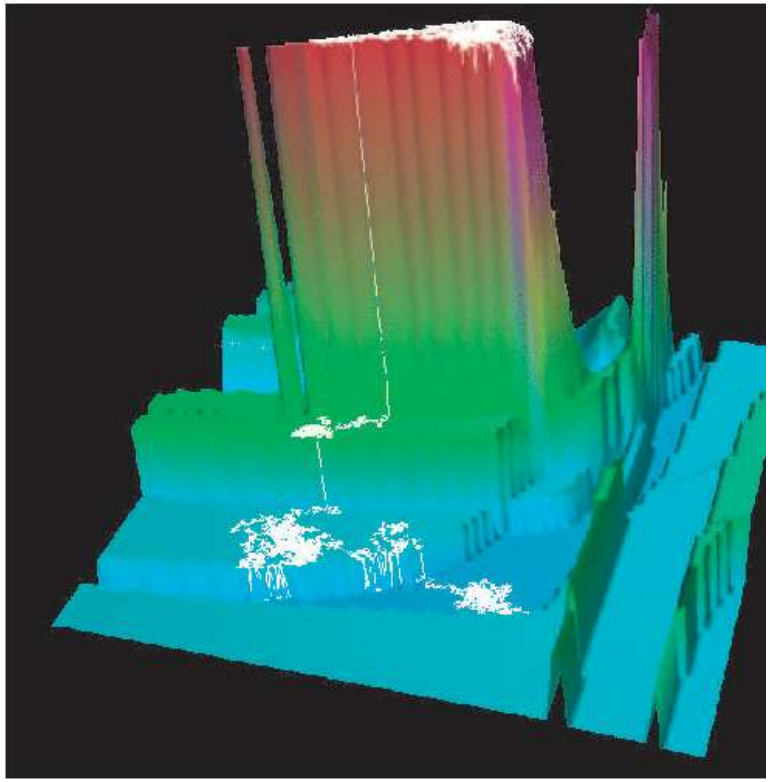
The Problem of Local Optima

- Hill-climbers get stuck on local optima; Exhaustive search does not
- GAs are less likely to get stuck on local optima than hill-climbers:
  - …because they maintain and evolve a *population of solutions*
  - …because genetic operators may generate a *wider range of neighbours*
  - These reasons exploit and rely on *diversity* in the evolving population
- Local optima are still a problem as *pop diversity is not guaranteed*
- Should a population of solutions end up *converged* within one basin of attraction they can find it difficult to escape
    - Getting stuck like this is called: *premature convergence*

University of BRISTOL

Neutrality

- Neighbouring genotypes with equal fitness are ‘selectively neutral’
  - …even if their phenotypes are different.
  - Evolution cannot choose between them…
  - …which results in ‘evolutionary drift’
- Neutrality is often easy to overlook and hard to visualize, but it can be key…
  - E.g., ‘neutral networks’ that percolate the search space may allow converged populations to escape “local optima”

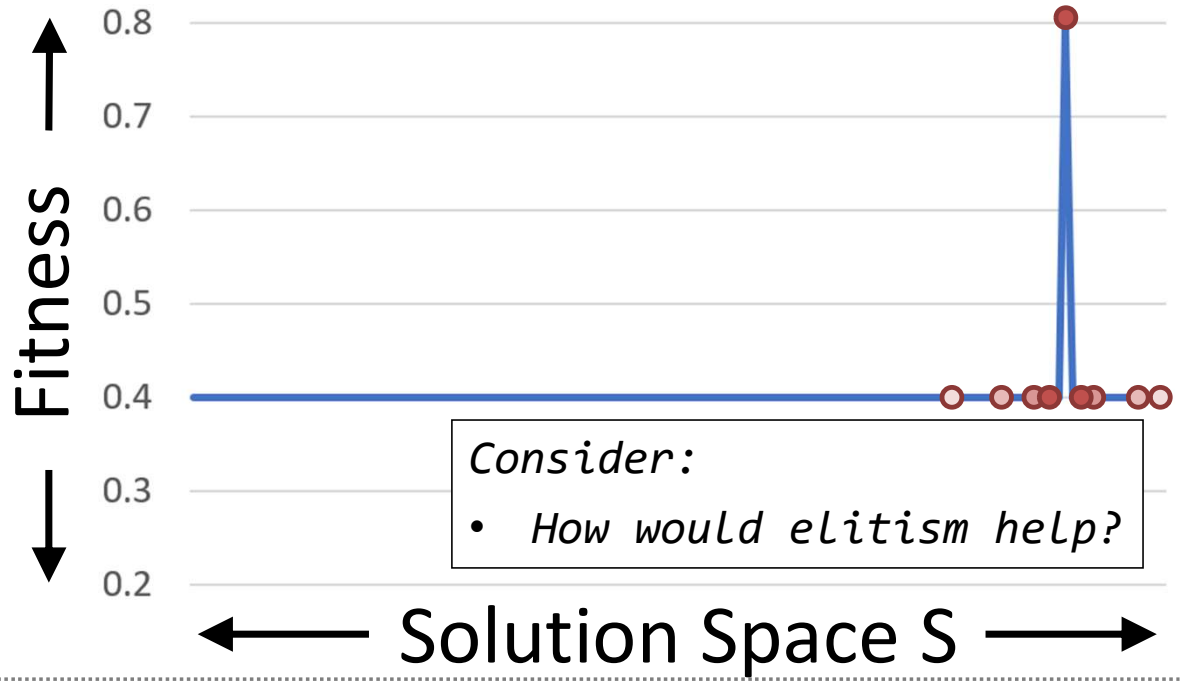


Barnett (2002).
Explorations in Evolutionary Visualisation

University of BRISTOL

The Invisible Needle

- Recall the “needle” fitness landscape we saw in a previous lecture
- The needle is hard to find because there’s no gradient to climb
- But even if we start the pop *on the needle*, we still might not find it…
  - Notice: it’s very rare for an offspring to be an exact copy of their parent
  - So a perfect solution will tend to have unfit offspring…



University of BRISTOL

The Quasi-Species Concept

- Manfred Eigen & Peter Schuster developed the *quasi-species* concept to explain what's happening here:
  - High GA mutation rates (much higher than for DNA) mean that the evolving population is a *cloud* of points on the landscape: a *quasi-species*
  - If the selection pressure on the population to reward good solutions…
  - …is overcome by the mutation pressure that constantly corrupts them…
  - …then the population effectively cannot even "see" the needle.
- Conversely: if an evolving population *is* able to find a good solution, *and stay there*, we can infer that the solution must be surrounded by other pretty good solutions – it is *robust*

University of BRISTOL

Epistasis

- If we're very lucky, fitness is a *linear function* of the gene alleles:
  - The contribution of each gene to fitness is *independent* of other genes
  - `1-max` is a linear fitness function: fitness = the number of 1s in a bitstring
  - Trivial to optimize: optimize each gene independently
- But almost always, fitness is a *non-linear function* of gene alleles:
  - The contribution of genes to fitness is *interdependent* to some extent
  - The best allele choice at one gene *depends* on the alleles at other genes
  - E.g., baby skull size ↔ size of birth canal; weapon range ↔ visual range
    - This is *epistasis*: more epistasis makes a search problem harder

University of BRISTOL

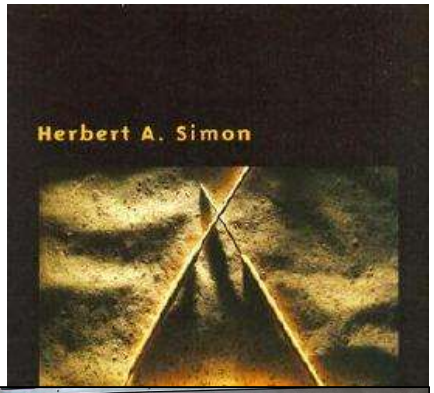
Epistasis

- The presence of epistasis ⇔ the existence of local optima.
  - Epistasis / local optima are different ways of describing the same thing:
  - Non-linearities in the mapping of fitness onto the search space
- Epistasis means we can't optimize each gene *independently*
  - Instead our algorithm must *co-optimize* the alleles of multiple genes
- How big is the set of interacting genes? (the 'order' of interaction)
- How many sets of interacting genes? How do they overlap?
- More generally, what is the *structural modularity* of the problem?

University of BRISTOL

Modular Decomposability

- Herb Simon introduces a nice way of thinking about problem modularity in his book *The Sciences of the Artificial* (1969): trying to crack a safe
  - A bank safe has *N* dials, each with 100 different settings
  - Only *one* setting is correct on each dial
  - *All* dials need to be correct to open the safe
  - (Recall that Dawkins uses the same example in his *Horizon* episode…)
    - How hard is it to open the safe?



University of BRISTOL

Modular Decomposability

The answer depends on the *modularity* of the safe

- For a safe with perfectly silent dials: we must search all 100^N possibilities
  - We can expect to have to check ½ of them => $100^N/2$ to crack the safe
  - No modularity; no useful problem structure; solution has strong epistasis
- But if each dial gives a little *click* only when it is at the correct setting:
  - We can expect to have to check ½ of the 100 possibilities *for each dial*
  - => $100N/2$ tries to crack the whole safe (much less time)
  - Full modularity; useful problem structure; solution has zero epistasis
    - The same as the “Methinks it is like a weasel” & 1-max problems

University of BRISTOL

Partial Decomposability

- But real problems tend to lie somewhere in between:
  - Partial modularity; useful problem structure; solution has some epistasis
  - Compare: “Methinks…” vs. evolving any correct English sentence
  - English sentences have modules (words) that depend on each other
- Watson (2006): consider a safe with dials that are sensitive to each other. For each dial, *n* settings (including the correct one) give a little click:
  - *n*=1 (full modularity); *n*=100 (no modularity); *n*=20 (some modularity)
  - We must try $n^N/2$ combinations (much less than $100^N/2$) to crack the safe
  - (After we find which are the *n* clicky settings on each dial: 100N tries)
    - What if *n* is halved for every dial that is in the correct setting?

University of BRISTOL

No Free Lunches

- Wolpert and Macready (1997): No Free Lunch for Optimization:
  - When algorithm performance is averaged across *all possible problems*:
  - *Any two optimization algorithms will exhibit equivalent performance*
  - *i.e., no optimization algorithm can do better than blind random search*
- The insight here is that every search algorithm other than random search has a *search bias*: it works on a hunch, gamble, or heuristic
- For every time the hunch pays off, there's a time where it doesn't
- Perhaps need to shift focus to *satisficing* rather than optimising…
- Algorithm quality depends on the *problem structure* that it faces

University of BRISTOL

Learning How to Search

- Recall that the structure of *S* (including it's modularity) is influenced by our choice of *representation* and *genetic operators*
- So, we can restructure the space by making different choices
- Good choices could be the difference between success and failure
  - Might it be possible to learn which operators to use when?
  - Or to learn a good genetic representation?
  - ..either for a whole class of different, related problems (e.g., timetables)?
  - ..or *online* for one problem *as we are trying to solve it*
    - Could a GA implement evolution that gets more + more powerful?

University of BRISTOL

The Evolution of Evolvability

- Natural evolution has got more powerful over time:
  - cf. the 'major transitions' in evolution (Maynard Smith and Szathmáry)
  - E.g., Single=>Multi-cellularity; A-sex=>Sex; Non-Social=>Social/Cultural
- In fact, Dawkins' paper at the first Artificial Life conference is on what his *Biomorphs* tell us about "the Evolution of Evolvability".
  - 'The ability of a population to generate *adaptive* genetic diversity'
  - (Here *adaptive* means 'well adapted' to the problem at hand)
  - i.e., evolvability is: how good is a population at evolving
    - A poor GA has low evolvability; Nature exhibits high evolvability

University of BRISTOL

The Evolution of Evolvability

- More subtle mechanisms in nature:
  - DNA repair – maintains some parts of the genome more carefully
  - Chromosome Organisation – nearby genes less likely to be disrupted
- Genetic Algorithms can look to exploit similar tricks:
  - Analyse how fitness varies in the current population – build a model
  - Use this model to predict what type of genotypic variation will be best
  - Effectively learning the structure of the problem during evolution
- E.g., “How can evolution learn?” (Watson & Szathmáry, 2015)
  - http://www.bbc.com/earth/story/20170301-life-may-actually-be-getting-better-at-evolving

University of BRISTOL

Example Questions

- Complete the missing fitness entries in the two tables of genotypes and their associated fitness values, below, such that there is (i) no epistasis, (ii) some epistasis. *[2 marks]*
- Give an example of two aspects of a real-world problem that are linked epistatically. Explain your answer. *[4 marks]*
- A problem comprises two complex but almost independent sub-problems. How should a GA's genotype be structured, and what type of crossover should be used? *[6 marks]*

University of BRISTOL

Thank you!

University of BRISTOL

Coevolution

Seth Bullock

bristol.ac.uk

University of BRISTOL

Fitness Function Design Is Hard

- As we have seen, a GAs success or failure is often down to our design decisions:
  - Our choice of genetic representation of the phenotype space
  - Our choice of genetic operators and selection scheme
  - Our design of a fitness function
- The third of these is particularly trying:
  - Many fitness functions are hard for GAs to make progress on due to local optima, ruggedness, neutrality, deception, etc.

University of BRISTOL

Coevolution

- But what if we could skip that tricky bit?
- Real evolving populations *do not consult a fitness function* to find out how many offspring each individual is allowed.
  - They just get on with it
  - Organisms co-operate & compete with the other organisms in their environment, and have offspring as part of this activity
- Coevolution: *when multiple populations of evolving organisms influence each other's evolution*

University of BRISTOL

Coevolution

- Coevolution = An automatic fitness function:
  - There is no fitness function designer in nature
- Coevolution = An implicit fitness function:
  - Instead of being set an *explicit* objective target by the fitness function, real organisms are assessed *relative* to their current environment
- Coevolution = A dynamic fitness function:
  - Instead of being assessed against a *fixed* target, real organisms cope with environments that change/evolve over time

University of BRISTOL

Example

- We would like to evolve a robot goalkeeper for RoboSoccer
  - Designing a fitness function has proven to be quite hard
- Instead, we decide to *coevolve* a population of goalkeeper robots against a population of robot penalty takers
  - We start off with random penalty takers and random goalkeepers
  - As soon as penalty takers start to be able to beat random goalkeepers..
  - ..there is pressure on the goalkeepers to get better..
  - ..putting pressure on the penalty takers to get better.. ..and so on.
    - If things work out, coevolution generates a good solution without us having to define a 'good' strategy via an explicit fitness function

University of BRISTOL

Coevolutionary GA

- Co-operative Coevolution:
  - Each population deals with a separate part of the overall problem
  - A solution combines individuals from each population
  - Appropriate when the overall problem is highly structured
- Competitive Coevolution:
  - One population of “solutions” to the problem that we are interested in coevolves against a population of “challengers” that must be defeated
  - Solutions are fit if they defeat many challengers.
  - Challengers are fit if they defeat many solutions.
    - Reminiscent of natural host-parasite or predator-prey coevolution.

University of BRISTOL

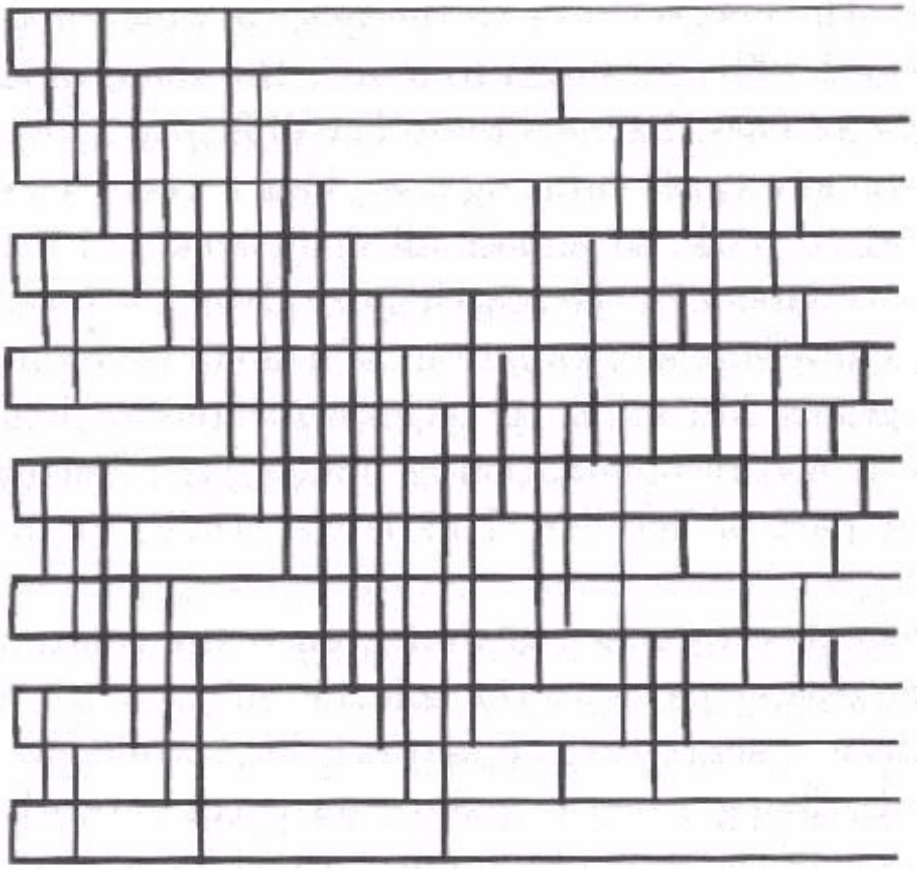
Examples

- The first coevolutionary GA was due to Hillis (1990) and was applied to solving a list-sorting problem.
  - What's the most efficient way to sort a list of length *n*?
  - Hillis pitched a population of 'hosts' (sorting networks) against a population of 'parasites' (each a set of lists to be sorted)
  - He initialised his host population with a reasonable solution and initialised his parasite population with random lists
  - Hosts rewarded for sorting parasite lists; parasites for being unsorted
  - In a few hours, he was able to coevolve a sorter that used 61 'gates'
    - Engineers had taken years to achieve a sorter with 60 gates

University of BRISTOL

List Sorting Example

- History of *n*=16 sorting networks
  - Batcher & Knuth (1964) **63** gates
  - Shapiro (1969) **62** gates
  - Green (1969) **60** gates (best known)
  - Hillis (1990): Evolved **65** gates…
    Hillis (1990): Coevolved **61** gates…
- Used in telecoms switches
- Now also relevant to GPUs
  - Saving even one gate can make a huge performance difference



University of BRISTOL

Why Did Regular Evolution Fail?

Evolution (~65K hosts for 5K generations) had problems:

1. Many local optima: reasonable sorters are surrounded by rubbish
2. Exhaustive assessment is prohibitively slow
   - There are 16! different lists to consider…
   - (Even reducing to 2^{16} binary lists is still a lot to check for each sorter)
3. But assessing against a random sample of lists is also inefficient:
   - Most randomly generated lists are sorted by reasonable sorters
   - Really challenging lists are rare, so there's a lack of pressure to improve

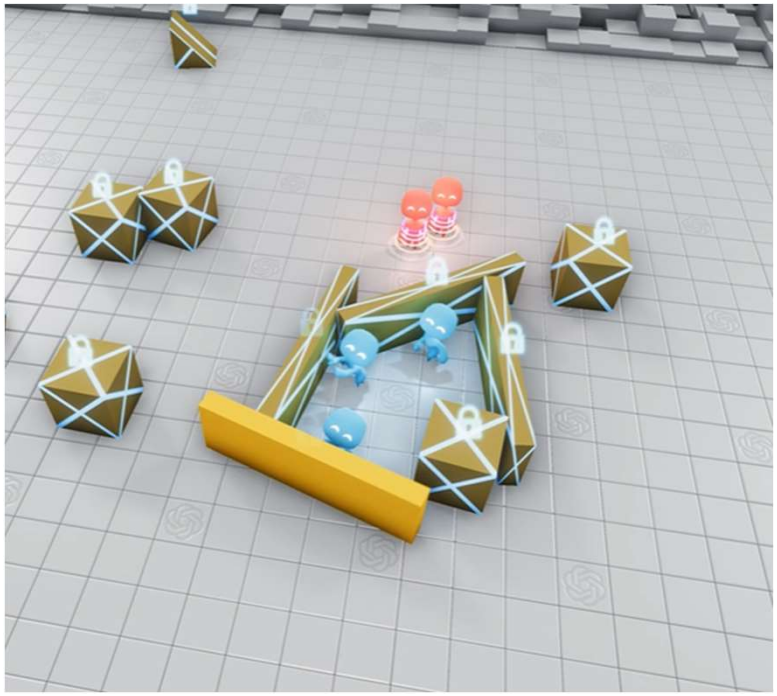
University of BRISTOL

Why Was Coevolution So Much Better?

- Coevolution solved these problems automatically:
  - Instead of random lists, sorters needed to sort sets of lists that were selected to become harder and harder.
  - If sorters got stuck on a local optimum, parasites were under pressure to punish that strategy by finding the lists that it couldn’t sort.
  - When improvements in the sorter population meant selection pressure got weaker, parasites were selected to get still harder.

Recall

From Karl Sims' (1994) *Blockies* to OpenAI's (2019) *Hide & Seek:*



University of BRISTOL

Thank you!

University of BRISTOL

Coevolution II

Seth Bullock

bristol.ac.uk

University of BRISTOL

Problems

- Coevolutionary arms races are supposed to *escalate* to increasing levels of sophistication… but sometimes that doesn't happen:
  - Stagnation: populations reach a poised state in which there is a turnover of individuals in the population but no phenotypic progress
  - Cycling: populations repeatedly cycle through the same weak strategies as if they were playing a game of scissors-paper-stone (the Red Queen)
  - Disengagement: every host is beaten by every parasite, all selection pressure is extinguished and coevolution effectively ceases (see Lab 2)
  - Measuring Progress: without a fitness function it can be difficult to assess whether genuine progress is being made by coevolving populations.

University of BRISTOL

Diversity

- To a large extent the problems with coevolution are familiar:
  - Our old foe: the loss of genetic and phenotypic diversity
- How can we stop the populations from converging and becoming vulnerable to stagnation, cycling, disengagement?
- Three different ideas:
  - Population structure
  - Fitness sharing
  - Reduced virulence

University of BRISTOL

Population Structure

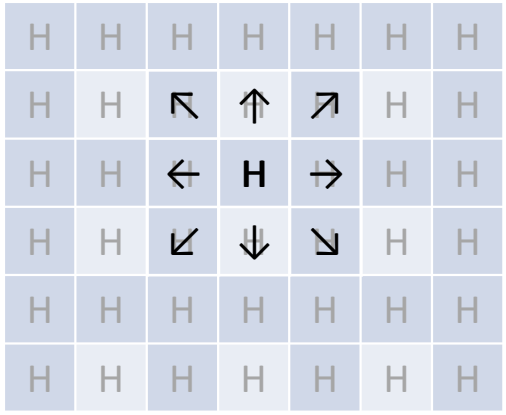
- The natural world is not randomly mixed up like an evolving soup.
- Populations are structured such that not every organism can interact with or compete directly with every other organism
- This is why island ecologies can be so distinctive and interesting
- GAs can do something similar
  - split populations into quasi-isolated sub-populations called ‘demes’
  - spread a population over a space and only allow local interactions

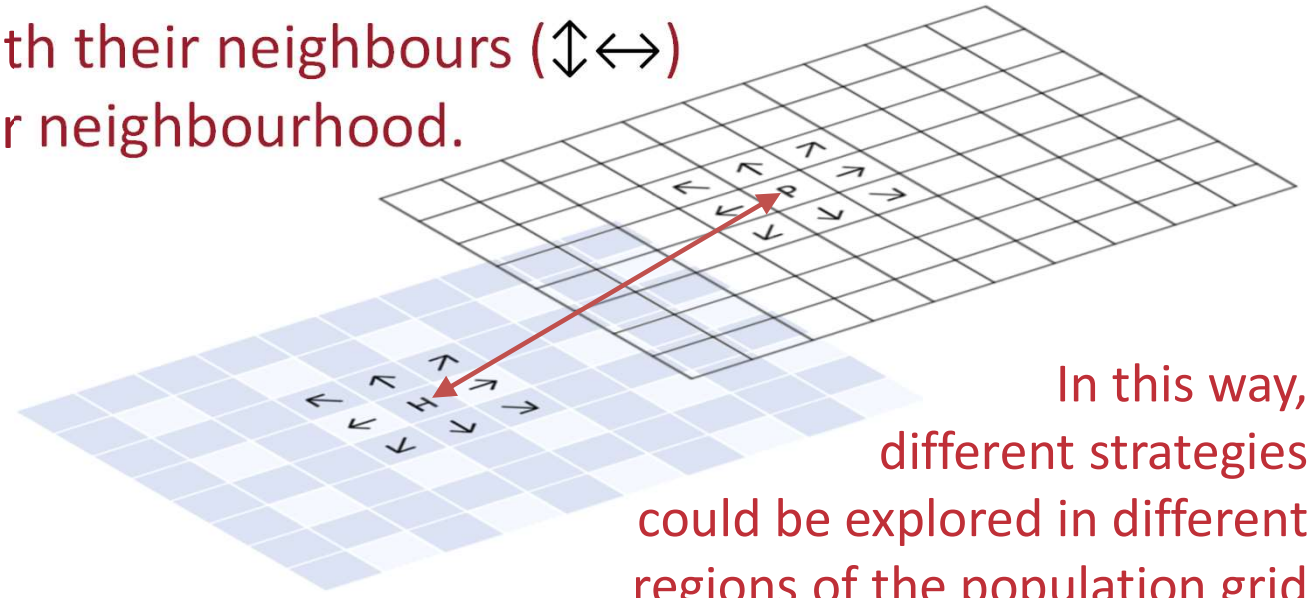
http://earthporm.com/alien-island/ Ben Shread © MoD/Crown Copyright

University of BRISTOL

Population Structure

- Hillis employed this idea in his list sorting work:
  - Each host was placed on a grid. Parasites were placed in a different grid.
  - Each host played the parasite at the same location in the other grid (↔)
  - Individuals competed with their neighbours (↕↔) to leave offspring in their neighbourhood.



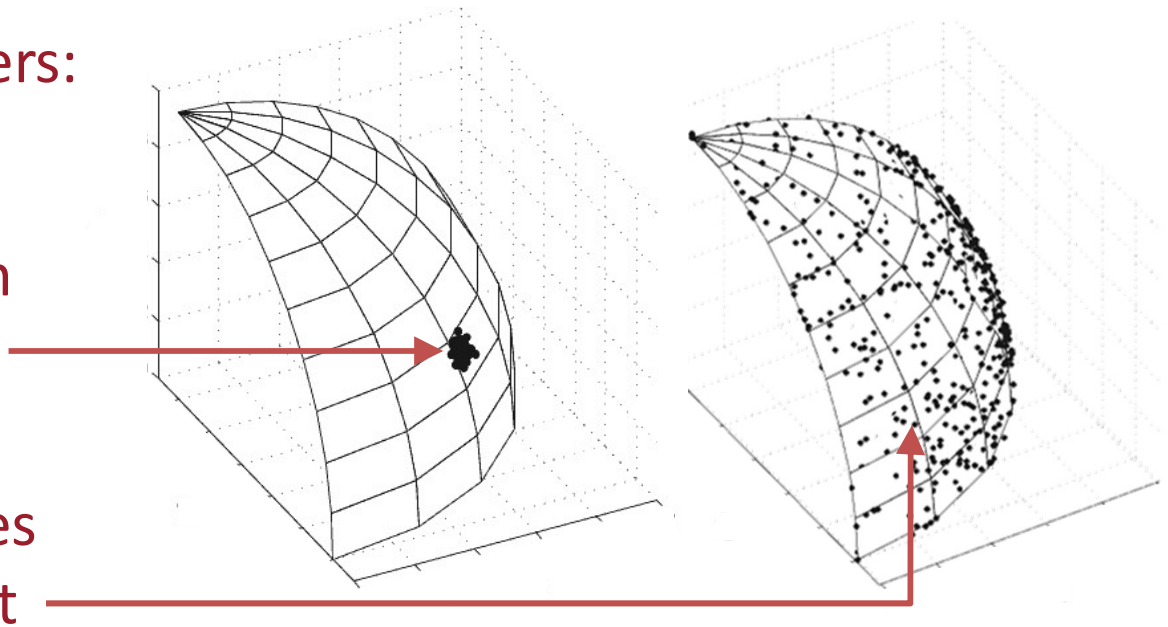


In this way, different strategies could be explored in different regions of the population grid

University of BRISTOL

Fitness Sharing

- A more deliberate way of maintaining diversity is to punish solutions that are similar to other solutions in the population:
  - Define a similarity measure (e.g., genetic hamming distance < threshold)
  - If solution *i* is similar to *n* others:
    - $f'_i = f_i / \sum_{j=1}^{j=n} \mathrm{sim}(i,j)$
  - Now, instead of converging on one region of this manifold of equally fit solutions…
    - …fitness sharing pushes solutions to spread out



University of BRISTOL

Reduced Virulence

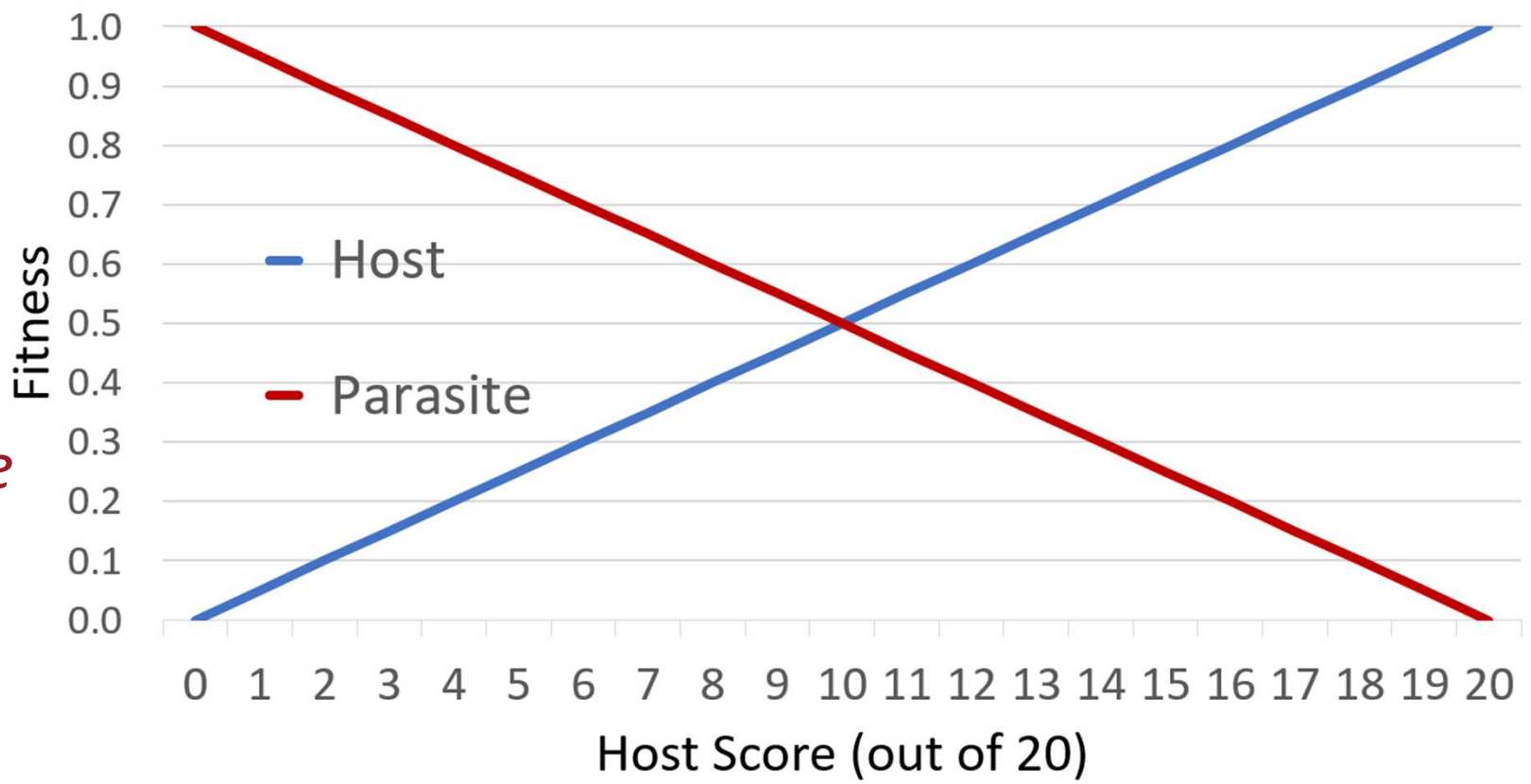
- Natural parasites / viruses often reduce their *virulence* over time
  - …it's not in their interest to kill their host quickly.
  - Consider the common cold or a tapeworm vs. extremely virulent Ebola
- But parasites in a coevolutionary GA are often rewarded for being *maximally damaging* to the hosts that they are assessed against…
- Our initial random robot goalkeepers won't benefit from facing penalty takers that smack the ball into the top corner every time.
- It would be better to start off with shots that are easy to save, and then get gradually harder…

University of BRISTOL

(Normal) Maximal Virulence

Normally, when a parasite plays against a host:

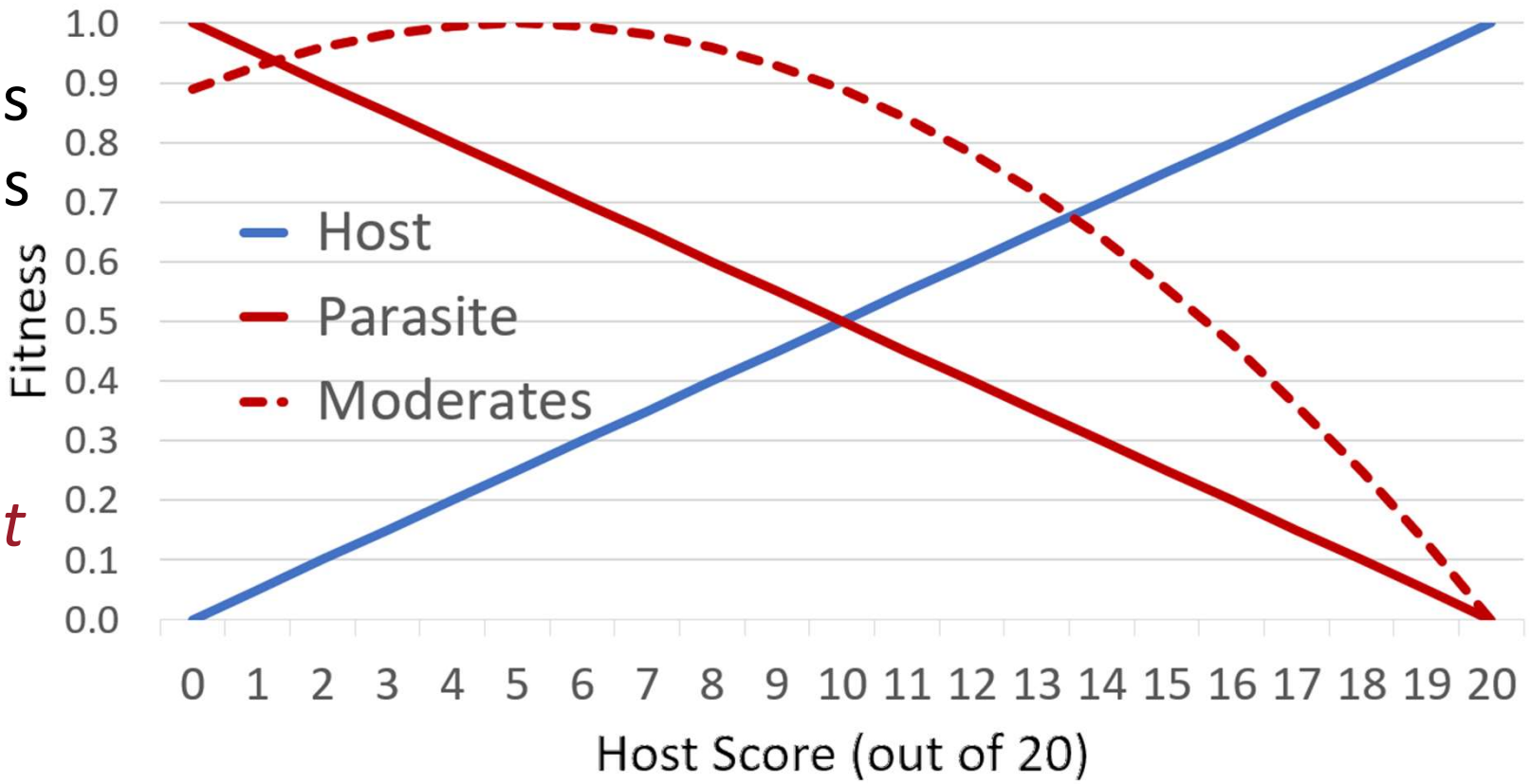
- If the *Host* scores *x* …
- … the *Parasite* scores 1-*x*



University of BRISTOL

Reduced Virulence

A ‘moderate virulence’ fitness function rewards parasites most for defeating hosts *most of the time, but not all of the time*…



University of BRISTOL

Coevolution

- More on these ideas in this weeks' GA Lab
- Ahead of the lab, please take a look at
  - Cartlidge, J. & Bullock, S. (2004). Combating coevolutionary disengagement by reducing parasite virulence. *Evolutionary Computation*, **12**(2), 193-222.
  - This lays out the details of the virulence idea, and introduces a simple game that we will be using to explore it.
  - It's a long paper, so don't feel you need to deep read all of it. ☺
    - Just focus on sections 1-4 (15 pages)

University of BRISTOL

Free Lunches?

- Recall Wolpert & Macready's (1995) No Free Lunch theorem
  - No search alg. can do better than random across *all* problems.
- More recently, in 2005, the same authors claim that *coevolutionary* free lunches are possible…
  - By using a separate (non-random) population to steer its search, a search algorithm may be able to always rule out some bad solutions, and thus out-perform random search…
- Check this stack exchange post for some intuitions…

University of BRISTOL

Example Questions

- Give an example of coevolution from nature. *[1 mark]*
- Name 3 problems that beset coevolutionary GAs. *[2 marks]*
- Why did Hillis embed his populations on a grid? *[4 marks]*
- What is premature convergence in a GA population. How do coevolutionary GAs hope to avoid it? *[4 marks]*
- Explain why a coevolutionary GA might cycle through the same poor solutions over and over again, rather than finding a better general purpose solution. *[6 marks]*

University of BRISTOL

Thank you!

University of BRISTOL

Artificial Life

Seth Bullock

bristol.ac.uk

University of BRISTOL

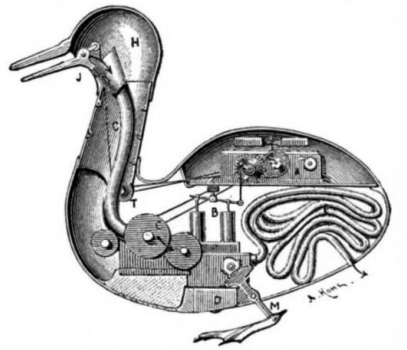
Alife and AI

- Artificial Life is a sister discipline for Artificial Intelligence
- Both take a functionalist approach to a mysterious subject:
  - AI: an artificial system can be *intelligent* if it is organised right
  - Alife: an artificial system can be *alive* if it is organised right
- Like AI, Alife combines a Strong project and a Weak project
  - Strong Alife: creating life inside a computer or in a robot
  - Weak Alife: understanding life by building life-like systems
    - Alife's pithy strapline: "The study of life-as-it-could-be"

University of BRISTOL

History

- Chris Langton named + founded Artificial Life in 1987
- But Alife has been going on for much longer than AI
  - Frankenstein, Golem, various ‘Automata’ and ‘Living Machines’:
  - Vaucanson’s Duck (1739): quacking, drinking eating, “excreting”
- The biggest names in CS were in fact Alifers…
  - Babbage: Evolutionary Models of Miracles
  - Turing: Models of Biological Morphogenesis
    - Von Neumann: Self-Reproducing Machines



INTERIOR OF VAUCANSON'S AUTOMATIC DUCK.
A, clockwork; *B*, pump; *C*, mill for grinding grain; *F*, intestinal tube; *J*, bill; *H*, head; *M*, feet.

University of BRISTOL

Self-Reproducing Machines

- How could a machine make a more complex copy of itself?
- This conundrum sounds impossible – paradoxical
- Von Neumann arrived at a two-part solution in 1948:
  1. Information encoding the structure of a machine …DNA
  2. Machinery that uses the information to make a new machine + a new copy of the information …cellular machinery
- In doing so, JVM predicted the contents of our cells…

…before the 1952 discovery of the DNA double helix.

University of BRISTOL

Universal Constructors

- But John Von Neumann went further than theorising and actually designed the first self-replicating machine!
- The first “Cellular Automaton” (CA):
  - A grid of cells, each containing a value
  - All values are updated according to a set of rules applied to each cell grid
    - Careful design of the rules plus the initial state of the grid, and…

University of BRISTOL

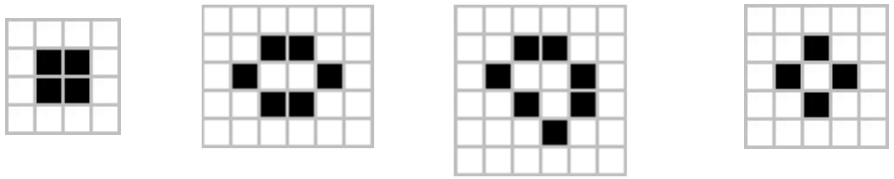
Conway's Game of Life

- You may be familiar with the most famous CA: Game of Life
- John Horton Conway (1937-2020) created the *Game of Life* in an effort to make a CA that had interesting behaviour:
  - Proliferating, repeating patterns; universal construction; etc.
- He settled on the following simple rules:
  - Each cell has a binary state – it can be alive or it can be dead
  - A live cell with >3 living neighbours or <2 living neighbours dies
    - A dead cell with three living neighbours becomes alive

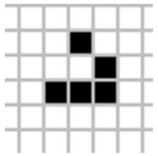
University of BRISTOL

Conway's Game of Life

- Some configurations are static:



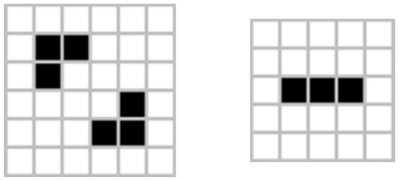
- Some oscillate on the move:



  - A "glider"

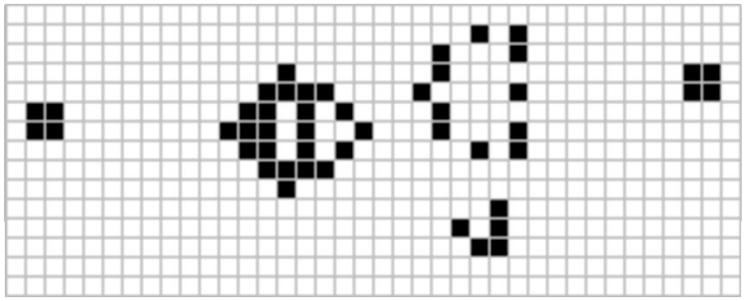
- Some destroy each other when they collide in the right way…

- Some oscillate in place:



  - A "beacon" and a "blinker"

- Some *generate* other structures: e.g., a *glider gun*

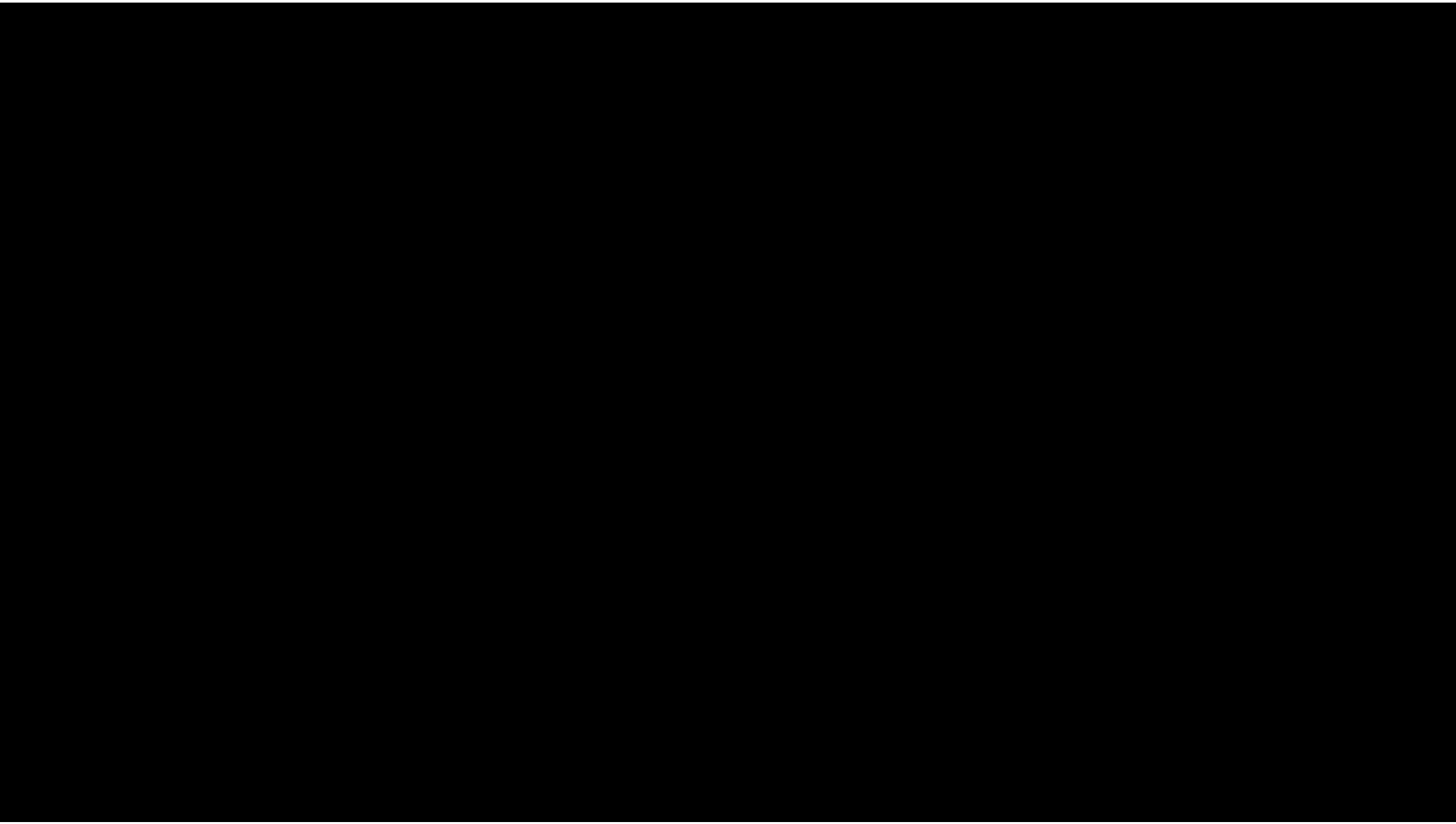


  - Amazingly, the Game of Life is provably Turing Complete…

Life in Life

Phillip Bradbury

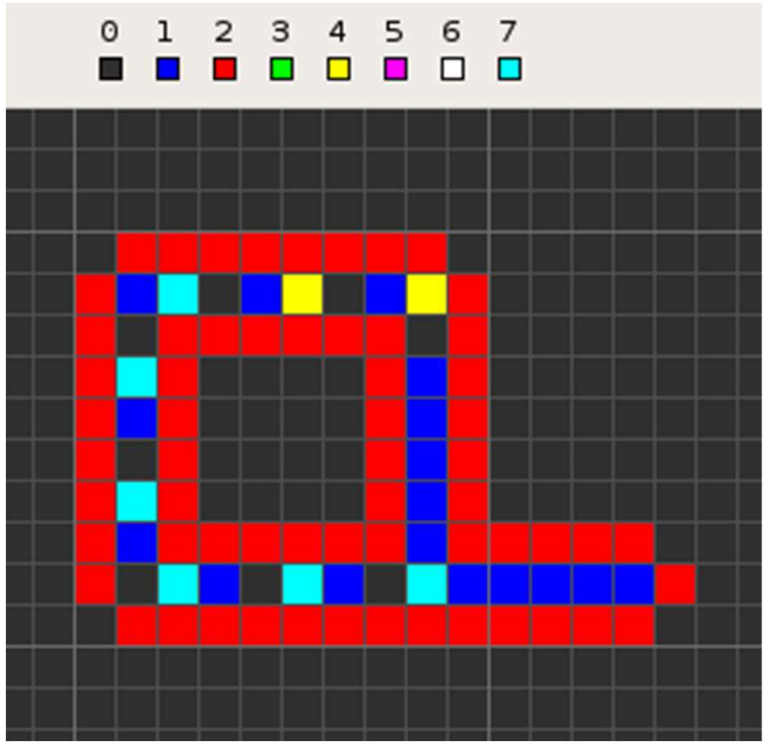
Roger Pincombe



University of BRISTOL

Langton's Loop

- Langton studied CAs + made a more simple self-replicator:

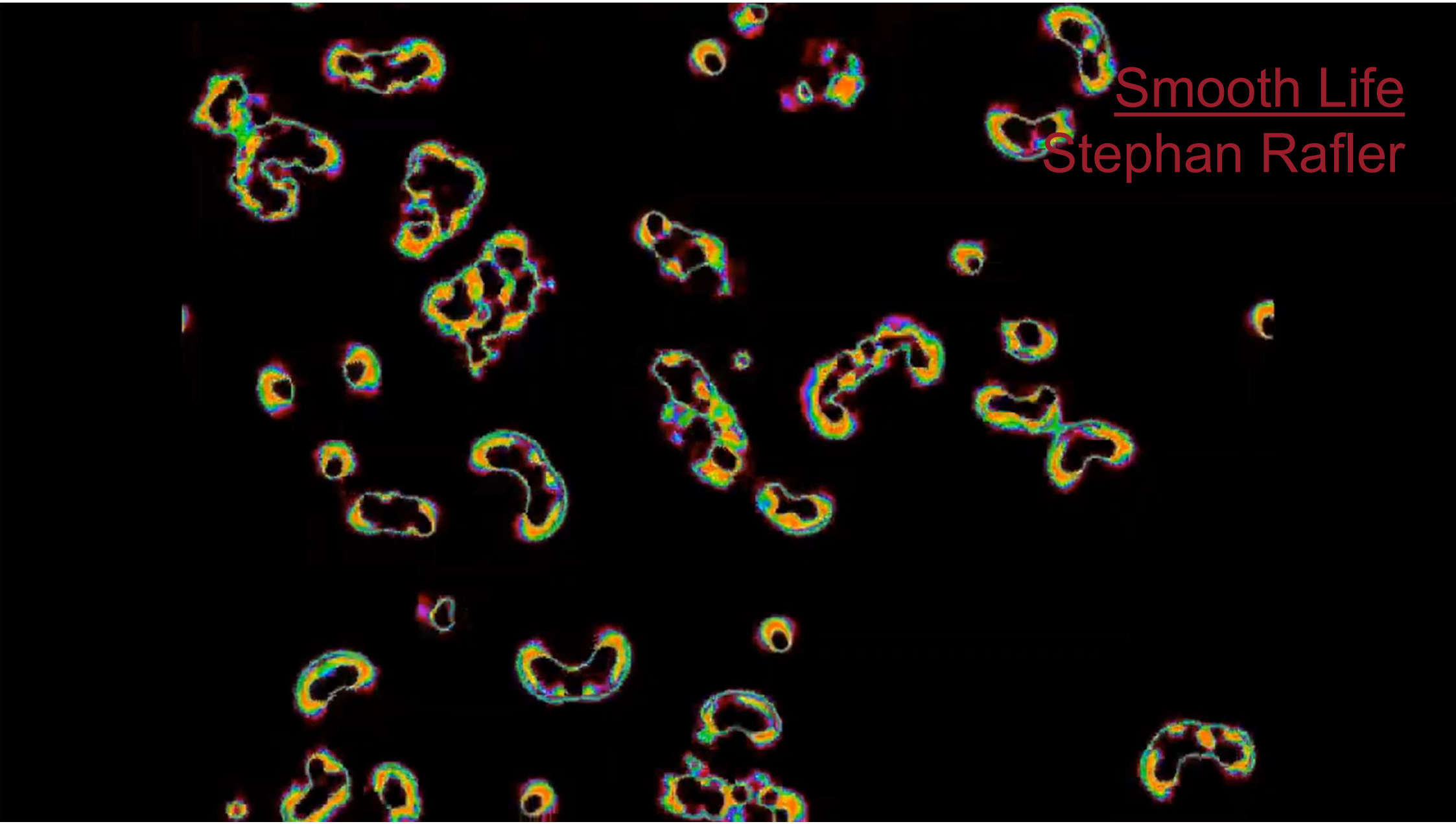


- Some CAs are boring, dead, inert, fixed;

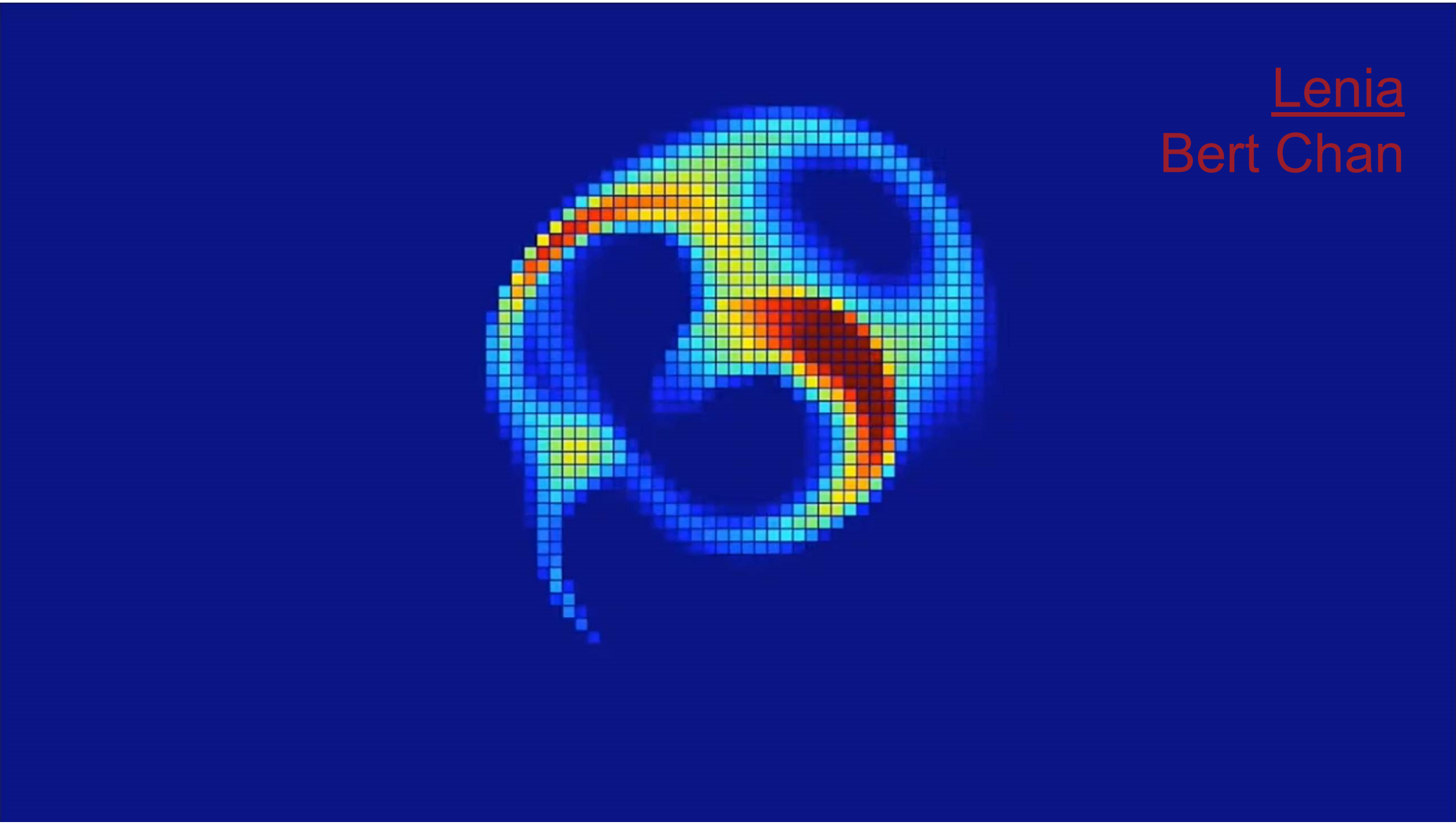
…some just repeat; some are messy chaos;

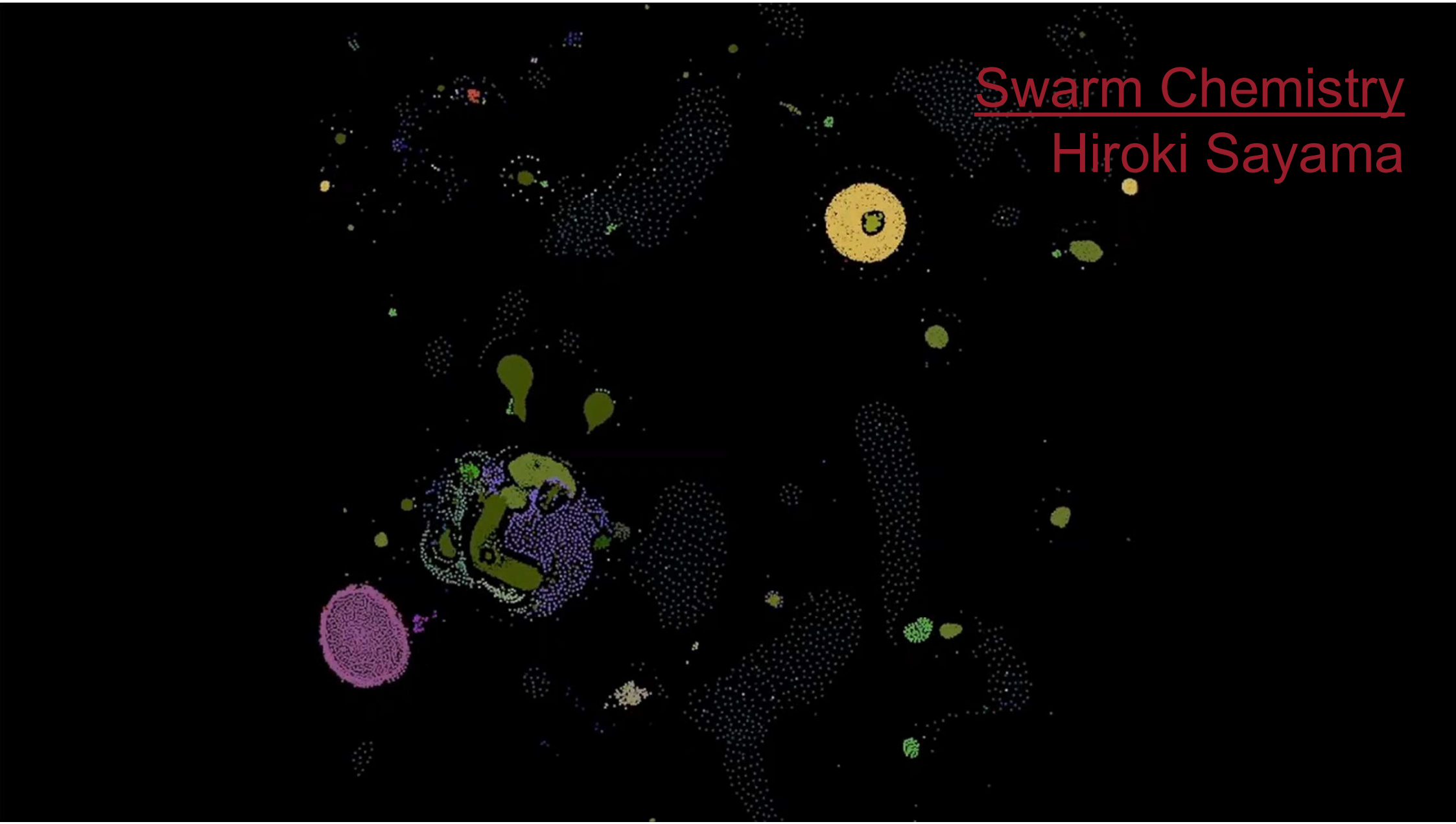
…and some are 'interesting' for a long time

- What is the physics of this complexity?
- Self-replicators + some kind of mutation = the possibility of *evolving replicators...*
- But how can truly 'open-ended evolution' be achieved in artificial systems?


Smooth Life
Stephan Rafler


Smooth LifeL
(Tim Hutton)

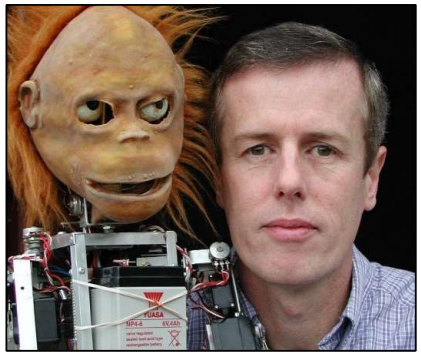

Lenia
Bert Chan

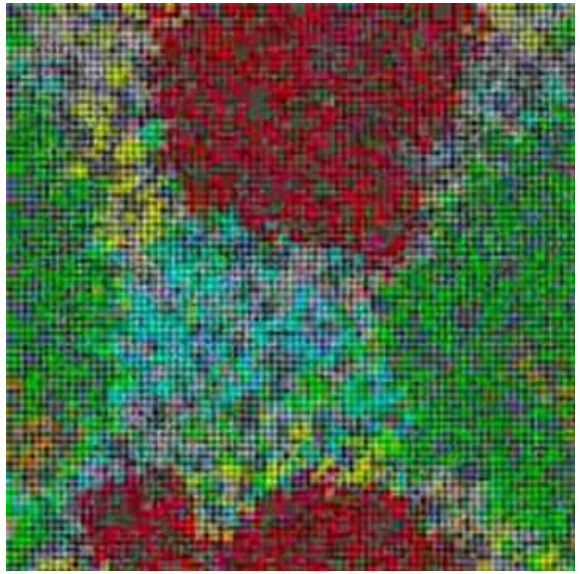

Swarm Chemistry
Hiroki Sayama

University of BRISTOL

Tierra to Avida

- Other Alifers explored the evolution of “digital organisms”:
  - Tom Ray’s *Tierra*: from replicators to parasites to hyperparasites
  - Chris Adami’s *Avida*: a kind of digital evolutionary laboratory
  - Alastair Channon’s *Geb*; Yaeger’s *Polyworld*.
  - Steve Grand’s video game: *Creatures*

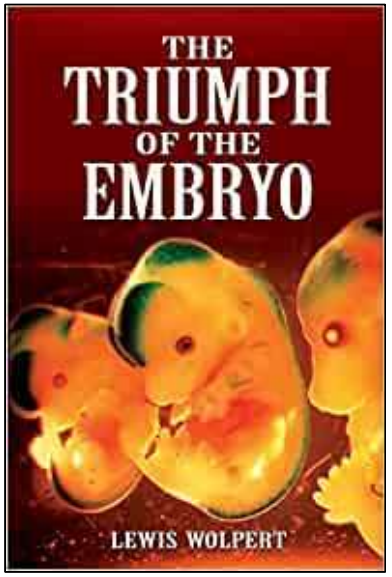




University of BRISTOL

From Evolution to Development…

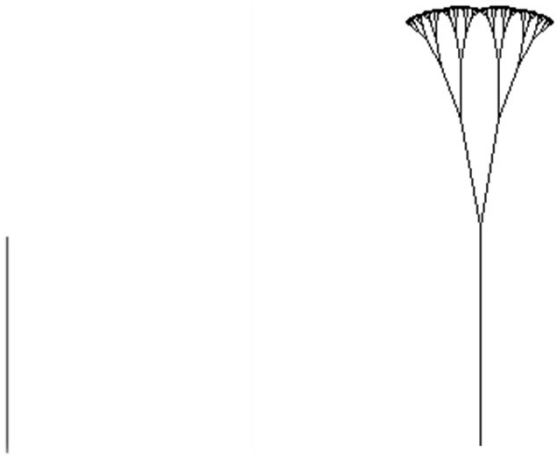
- Real biological lifeforms *develop* over time: morphogenesis
- Understanding how exactly an egg grows into a person is still one of the most profound scientific challenges
- Since before Turing: *growing* an AI has looked far more feasible than hand-crafting one
  - How can artificial systems grow and develop?
  - How can this development be steered or guided?



University of BRISTOL

L-Systems

- Lindenmayer Systems, Aristid Lindenmayer (1925-’89)
  - A model of plant development based on grammar rewriting
  - In language: S -> NP+VP; VP -> V+NP; etc., etc.
  - In L-Systems: I -> Y
  - More complex rules:

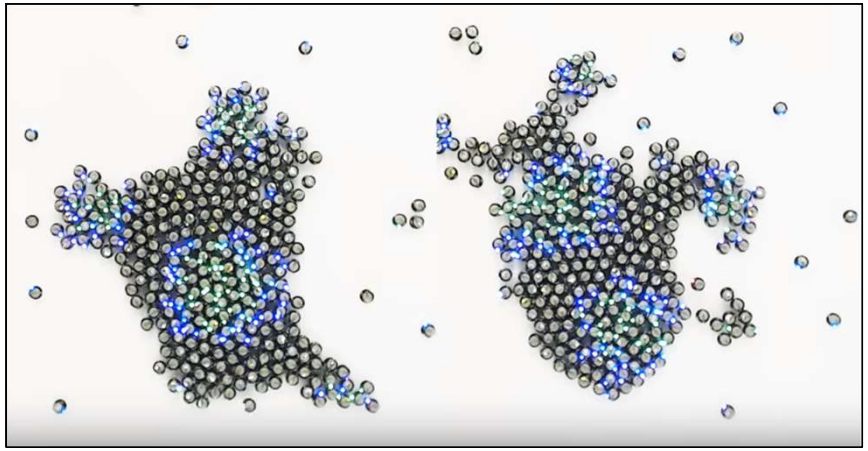


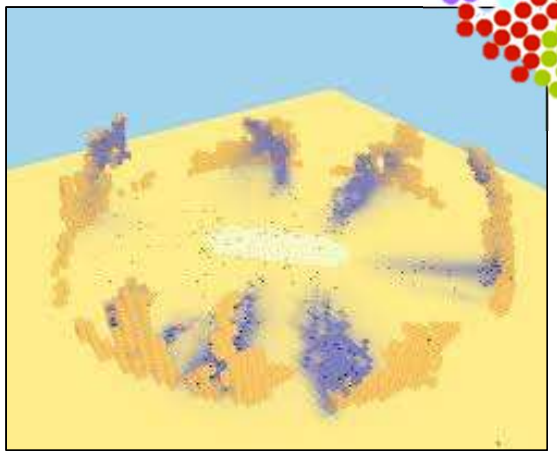
Przemysław Prusinkiewicz, A. Lindenmayer (1990). *The Algorithmic Beauty of Plants*

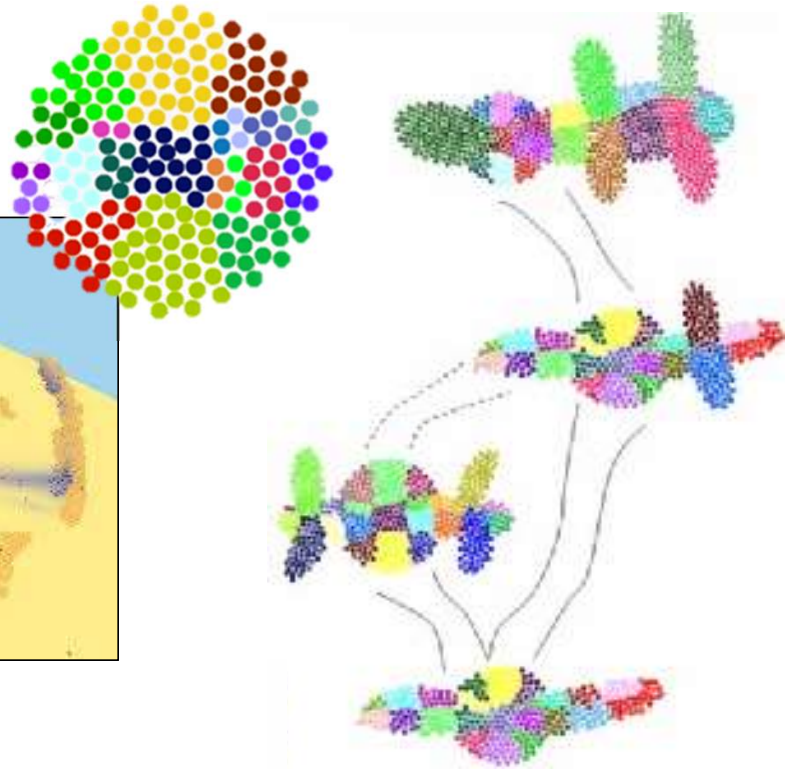
University of BRISTOL

Artificial Morphogenesis

- Morphogenesis is still a growing area of research in Alife: 😂
  - "Morphogenetic Engineering"
  - Swarm Construction

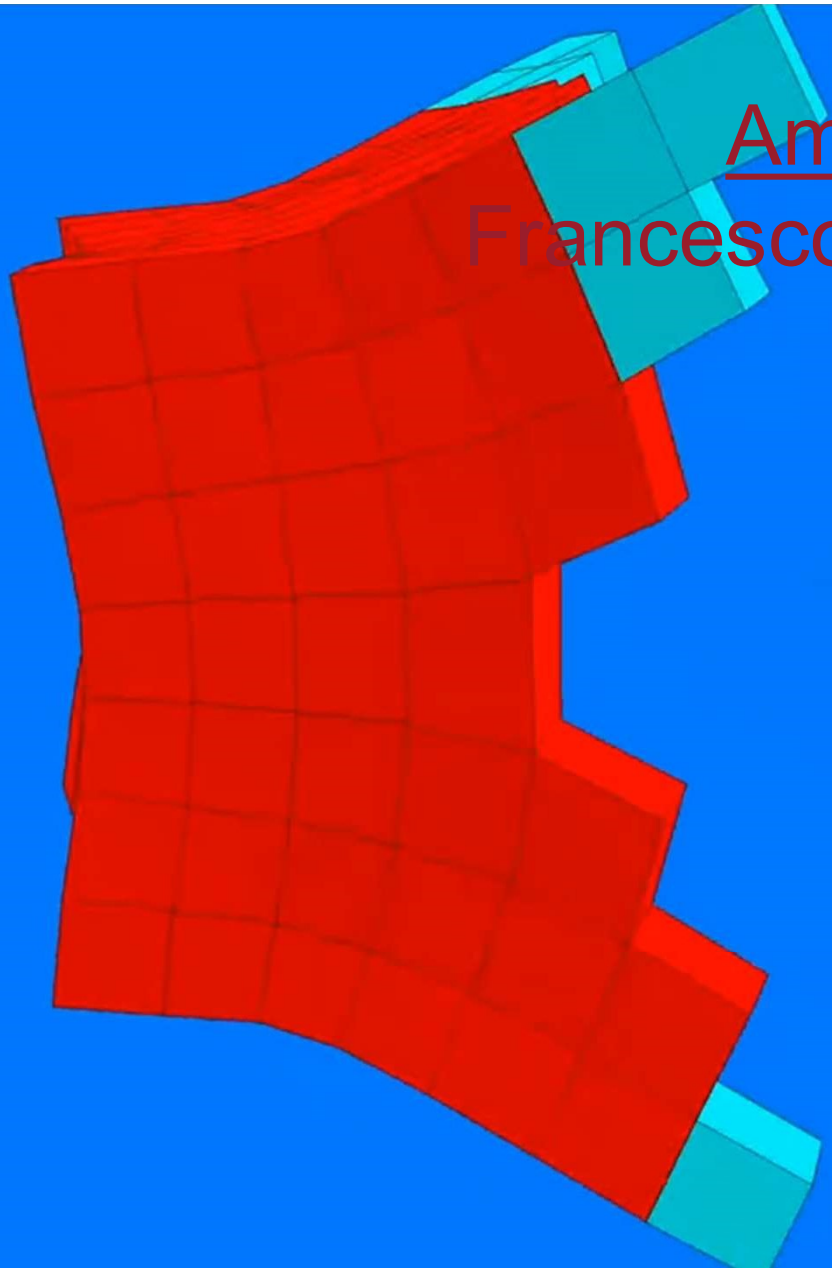






- Evo-Devo Soft Robotics

1



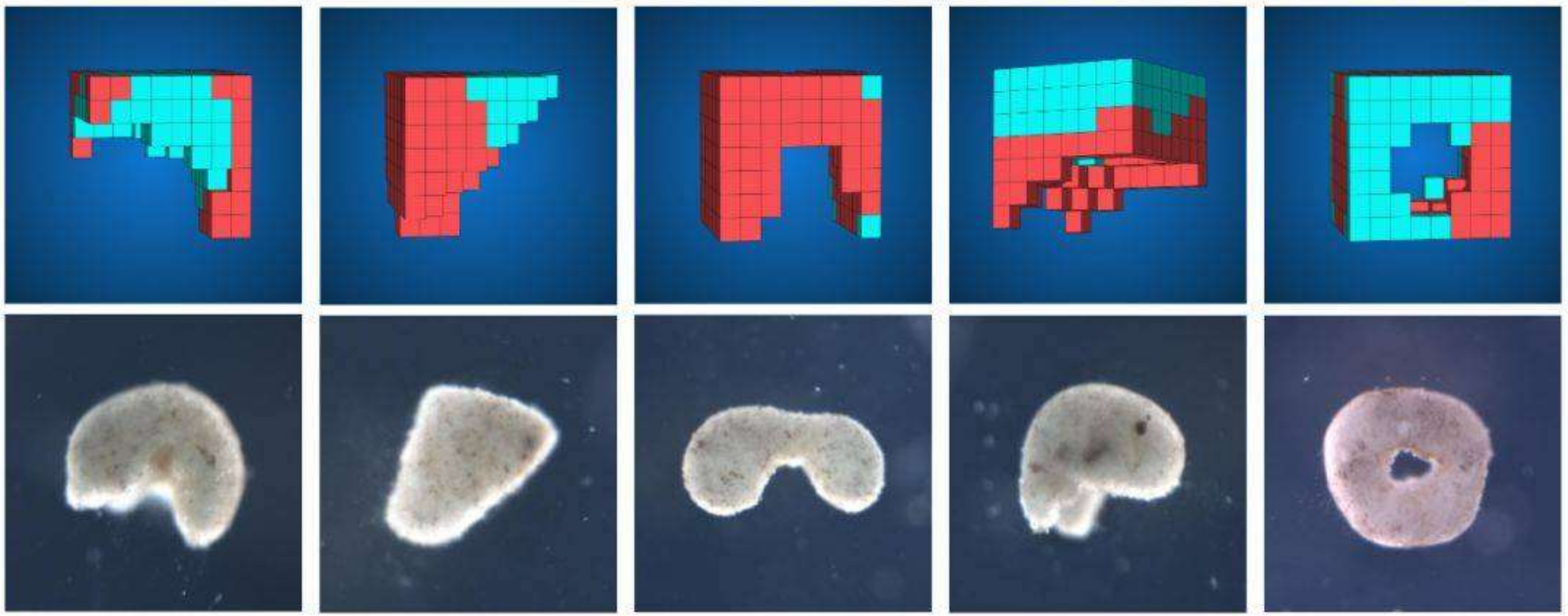
Amphibians

Francesco Corucci

University of BRISTOL

Xenobots

Robotics + soft-body simulation + evo-devo + bio-chemistry…



Josh Bongard (Vermont), Michael Levin (Tufts) and collaborators

University of BRISTOL

Types of Alife

Alife combines many disciplines

- Computer Science
- Robotics
- Biology
- Physics
- Chemistry
- Philosophy
- Linguistics

+ more

… and considers different life forms:

- Soft Alife: digital life
- Hard Alife: robotic life
- Wet Alife: biochemical life

- Whereas AI has focussed on:
  - Reasoning, Planning, Logic, etc.
- Alife has focussed on:
  - (Co-)Evolution, Development, Learning, Self-Organisation, Emergence, Origins of Life, etc.

University of BRISTOL

Open Problems

In 2000, a group of nine prominent Alife researchers authored a journal paper: "Open problems in Artificial Life": 14 challenges that are still live + relevant…

- How does life arise from the non-living? (5 problems)
  - Achieve the transition to life in an artificial chemistry.
  - Simulate a unicellular organism over its entire life cycle.
- What are the potentials and limits of living systems? (5 problems)
  - Determine what is inevitable in the open-ended evolution of life.
  - Develop a theory of information processing, flow and generation for evolving systems.
- How is life related to mind, machines, and culture? (4 problems)
  - Demonstrate emergence of intelligence in an artificial living system.
    - Establish ethical principles for artificial life.

University of BRISTOL

Women in Alife!

bristol.ac.uk

University of BRISTOL

Further Reading

- Langton, C. (1989). Artificial life, in *Artificial Life*, Addison-Wesley.
- Lehman, Clune, Misevic, Adami, et al., [52 authors!] (2020). The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artificial Life*, 26:2, 274-306.
- International / European Alife Conference proceedings at alife.org
- The *Artificial Life* journal at MIT Press | Lana Sinapayen's Prezi 🔥
- Hiroki Sayama's PyCX Python package and associated text book
    - NetLogo Life, Alife Virtual Seminars, Alife Papers

University of BRISTOL

Example Questions

- Which cells in the Game of Life grid shown below will survive to the next time step, and which will not? *[2 marks]*
- How did Von Neumann's insight into self-replicating machines anticipate the DNA molecule? *[4 marks]*
- Why might it be more feasible to grow a general purpose AI rather than code one by hand? *[4 marks]*
- Apply two iterations of the following L-system rules to the starting shape shown below. *[3 marks]*

University of BRISTOL

Thank you!