Lecture 10: The Evolutionary Computation Zoo

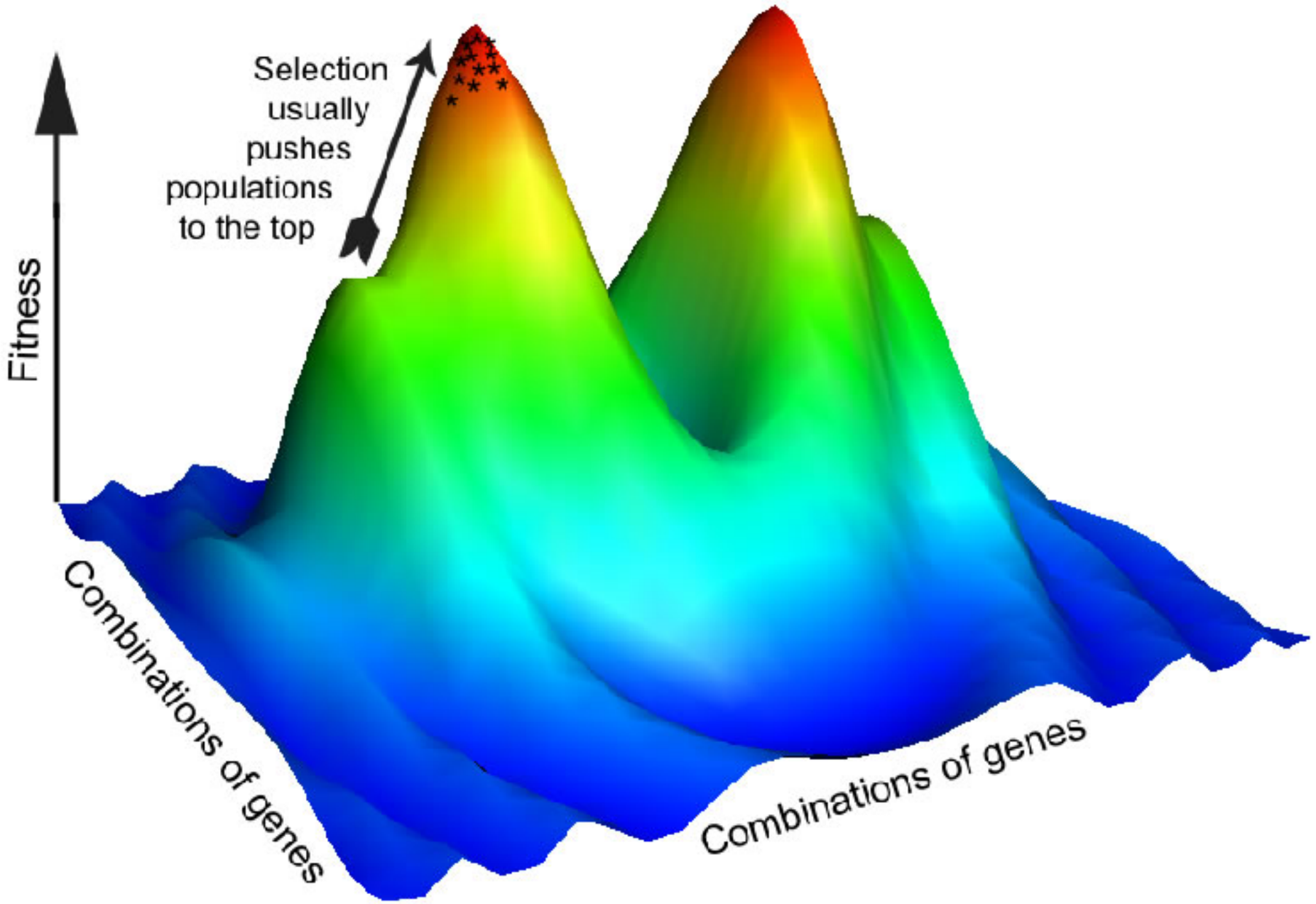
Artificial Intelligence
CS-GY-6613

Julian Togelius
julian.togelius@nyu.edu

Generic EA

```
BEGIN
   INITIALISE population with random candidate solutions;
   EVALUATE each candidate;
   REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
      1 SELECT parents;
      2 RECOMBINE pairs of parents;
      3 MUTATE the resulting offspring;
      4 EVALUATE new candidates;
      5 SELECT individuals for the next generation;
   OD
END
```

The fitness landscape



Simple μ+λ Evolution Strategy

- Create a population of μ+λ individuals
- Each generation
  - Evaluate all individuals in the population
  - Sort by fitness
  - Remove the worst λ individuals
  - Replace with mutated copies of the μ best

Generally, you need

- A solution representation
- Variation operators (mutation and/or crossover)
- A fitness (evaluation) function

Representation

- Individuals in the population are referred to as genotypes (e.g. arrays of doubles)
    - All variation and replication mechanisms are performed on the genotype
- What is evaluated by the fitness function is the phenotype (e.g. neural networks)
- Genotype-to-phenotype mapping should be fast, deterministic, preserve locality

Representations should be...

- Compact
- Expressive (represent all “interesting” parts of phenotype space)
- Human-understandable (?)
- Local (?)

Evaluation function

- Also called fitness function
- Returns a number (or ranking) of the evaluated individual (phenotype)
- You could have one or many
- Evaluation functions should be smooth, fast to evaluate, noise-free, accurate and relevant

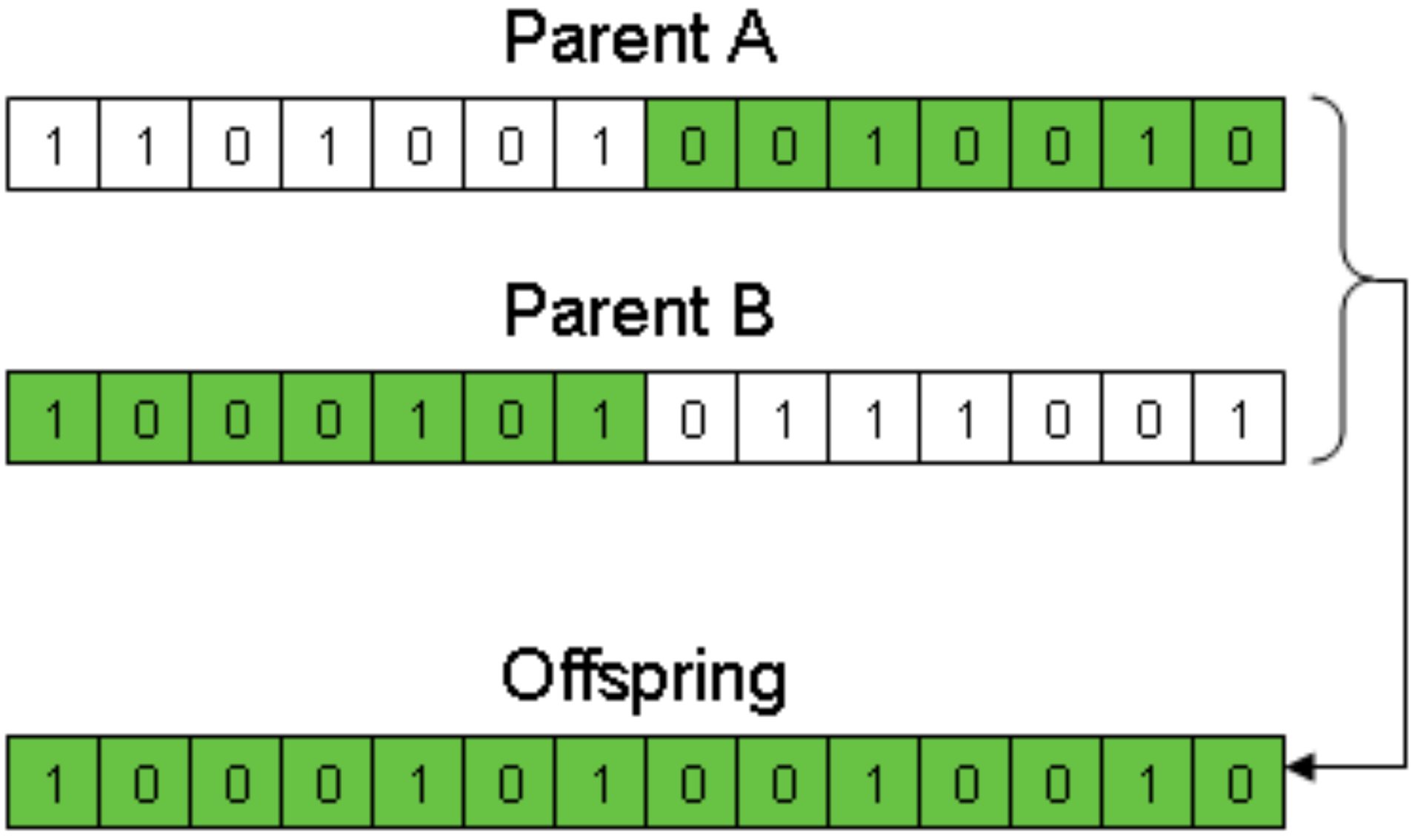
Parent selection

- Which individuals get to reproduce?
- Can choose this based on fitness...
- ...or fitness ranking
- Deterministic or stochastic
- How many to select?

Variation operators

- Should be appropriate to the representation
- Mutation
  - Should have the appropriate magnitude
    - e.g. Gaussian mutation: add values drawn from Gaussian distribution to an array of real numbers. Gaussian distribution is a random distribution with a mean and a standard deviation. e.g. in Java: random.nextGaussian ()
    - Or: add a value to all parts of the genome drawn from uniform distribution. e.g. Math.random () * some small value
- Recombination (crossover)
  - Only do this if you know what you are doing

Example: one-point crossover



Survivor selection

- Elitism?
  - Usually a good thing
- Who to replace?
  - Always kill the worst?
  - Diversity preservation?

Initialization

- Population should be diverse enough

- Should you start from results of previous evolutionary runs?

Termination condition

- When you're done?
- When you've waited long enough?
- Relevant factor to count: number of evaluations

Simple μ+λ ES (concrete)

- Create a population of μ+λ individuals (e.g. 50+50) represented as e.g. vectors of real numbers in [0..1]

- Each generation (until 100 generations)
  - Evaluate all individuals in the population
  - Sort by fitness (e.g. win rate or score for the game; higher is better)
  - Remove the worst λ individuals
  - Replace with mutated copies of the μ best (mutate through Gaussian mutation with mean 0 and s.d. 0.1)

Types of evolutionary algorithms

- Genetic Algorithms
  Representation: bitstring
  Mostly driven by crossover

- Evolution Strategies
  Representation: real values
  Mostly driven by (self-adaptive) mutation

- Genetic Programming
  Representation: expression trees

- Estimation of Distribution Algorithms, e.g. CMA-ES

- Others: Particle Swarm Optimization, Differential Evolution…

Taking a step back…
How would you actually use an evolutionary algorithm?

Uses of evolution:

- Evolve a route
- Evolve the parameters of the car
- Evolve a controller that can drive the car


MARIO: 1024
COINS
18
DIFFICULTY
10
TIME
144
WorldPause
false

Uses of evolution:

- Evolve level
- Evolve heuristic (for search algorithm)
- Evolve search algorithm
- Evolve path
- Evolve physics
- Evolve enemies


Pac-Man vs Ghosts


Pac-Man vs Ghosts

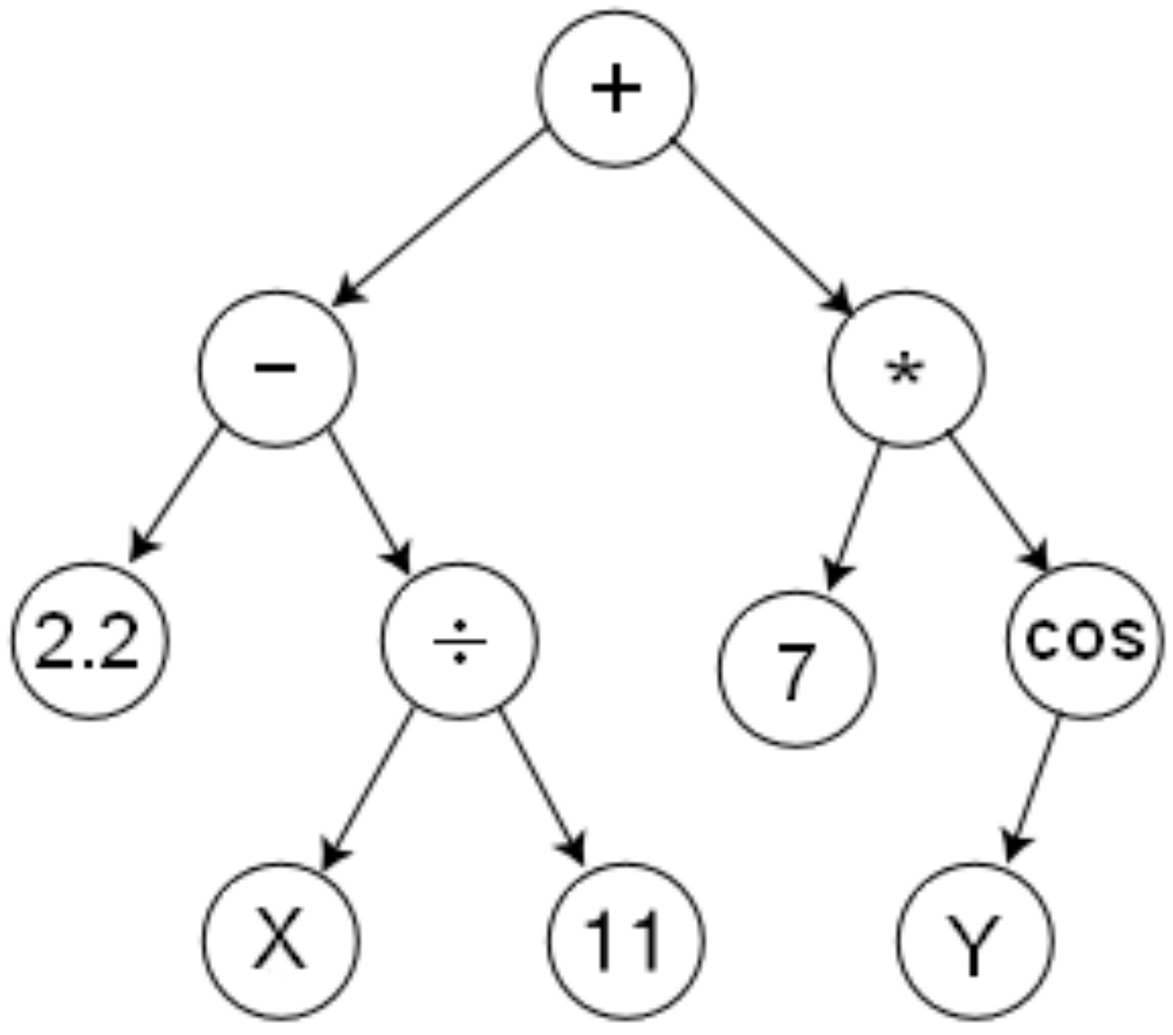
Uses of evolution:

- Evolve levels
- Evolve ghost behaviors
- Evolve controllers
- Evolve paths (every time step)

Uses of evolution:

- Evolve players
- Evolve board state evaluators
- Evolve piece positions
- Evolve pieces
- Evolve new rules

Genetic programming

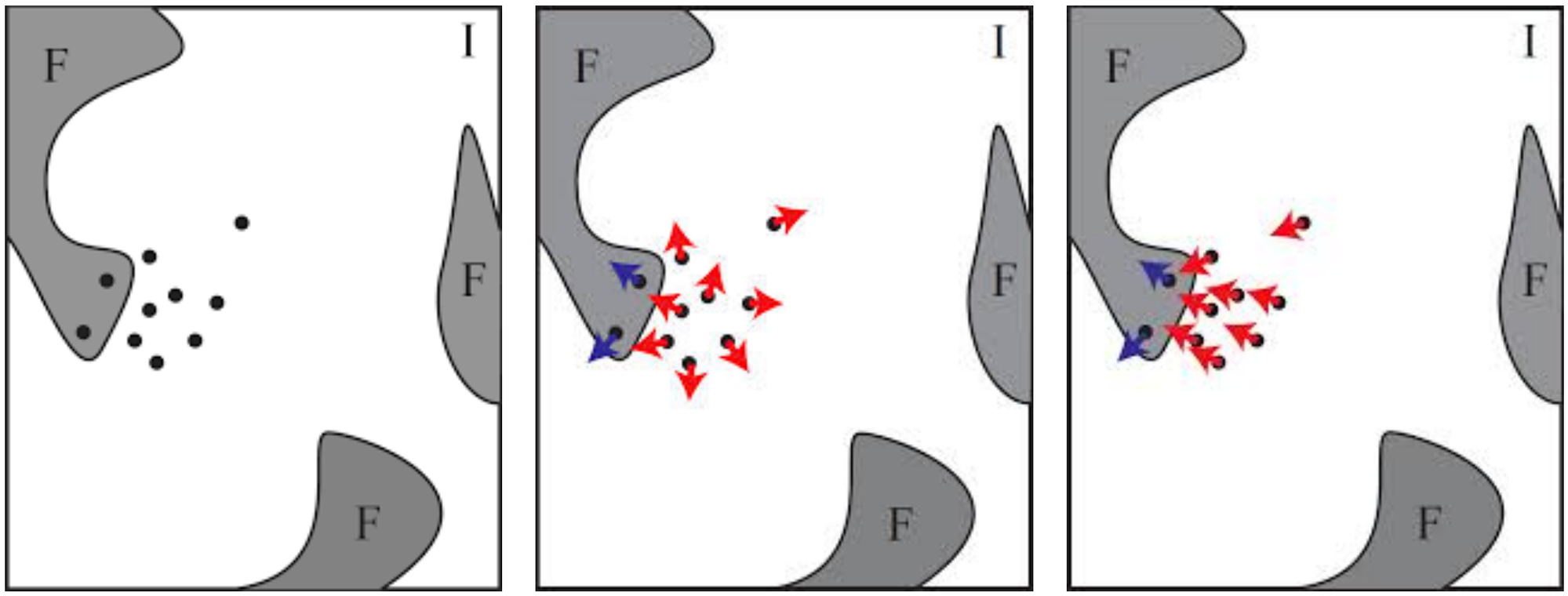


$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$

Evolution with constraints

- Sometimes there are certain features which *must* be part of the solution
  - e.g. car has four wheels, path passes through New Jersey, wing length less than 40m
- These could be encoded in the fitness function - but gets complicated
- Better, encode them as constraints

Feasible and Infeasible Areas



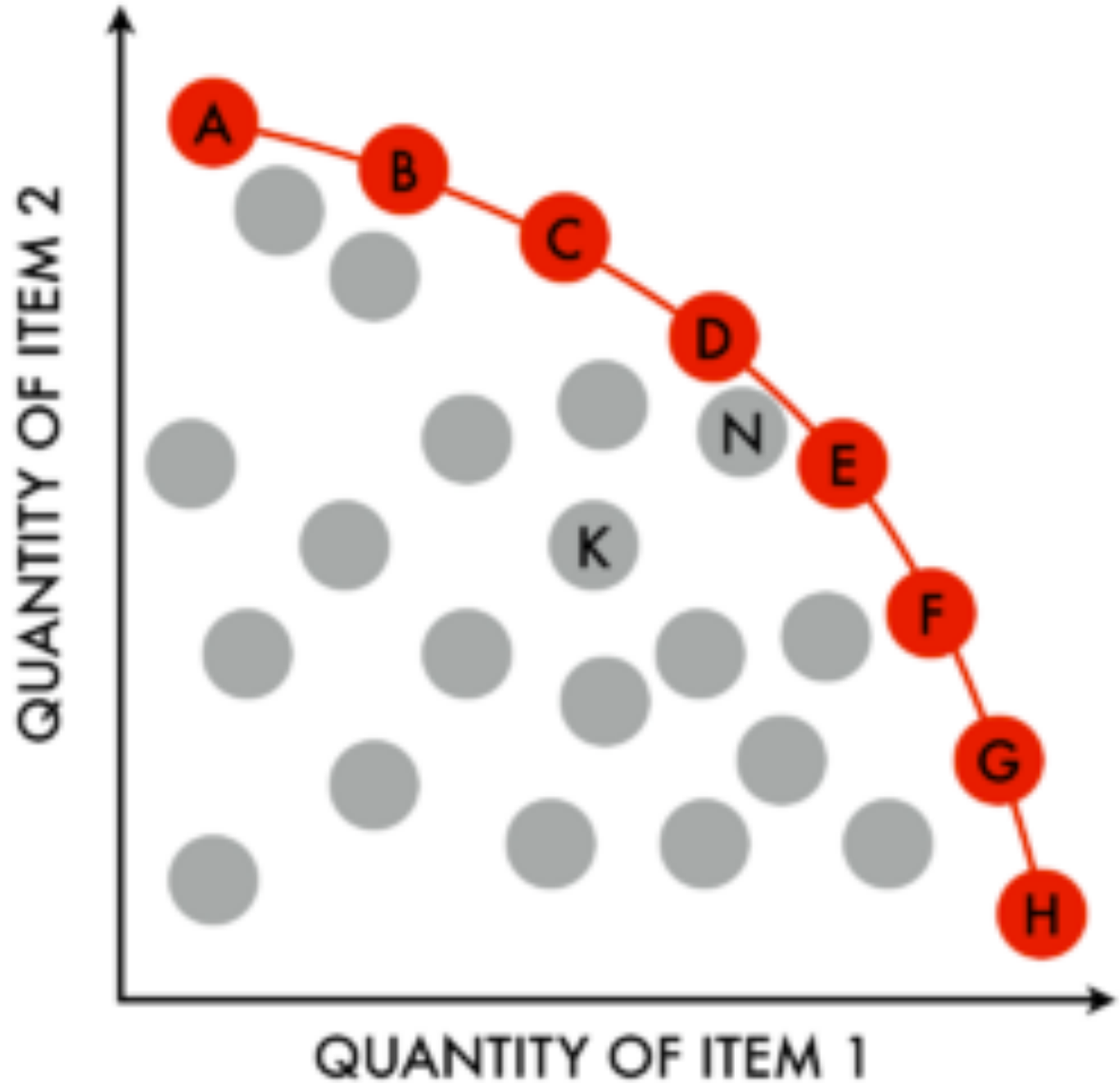
FI-2Pop algorithm

- Keep two populations: feasible and infeasible individuals
- Feasible individuals are selected for their fitness
- Infeasible individuals are selected for their proximity to feasibility
- Infeasible offspring of feasible parents are moved to infeasible population

Multiobjective evolution

- Often, you want to optimize for multiple fitness dimensions
    - For example, you want the car to be fuel-efficient, cheap and fast…
- These objectives are often *partially conflicting*
- A multiobjective evolutionary algorithm finds the *pareto front* of solutions given these objectives

Pareto fronts




© Reuters

Competitive coevolution

- Two (or more populations) with different fitness functions
  - e.g. predators and prey
- The fitness of one population depends on the other and vice versa
- Goal: start an *arms race*