Logistics

- Project 1 (Dynamical Systems) has been turned in (Congratulations).

- You should have received a email from Bianca with your two review assignments (Check with us immediately if you have not).

- Reviews are due Friday, 2-17.

Logistics

- Stability analysis
  - Linear systems: take the eigenvalues of the matrix defining the system.
  - Non-linear systems: linearize by calculating the Jacobian…

    … then take the eigenvalues.

Those who figured that out despite my inadequate example (linear only in the old slides) get extra credit.

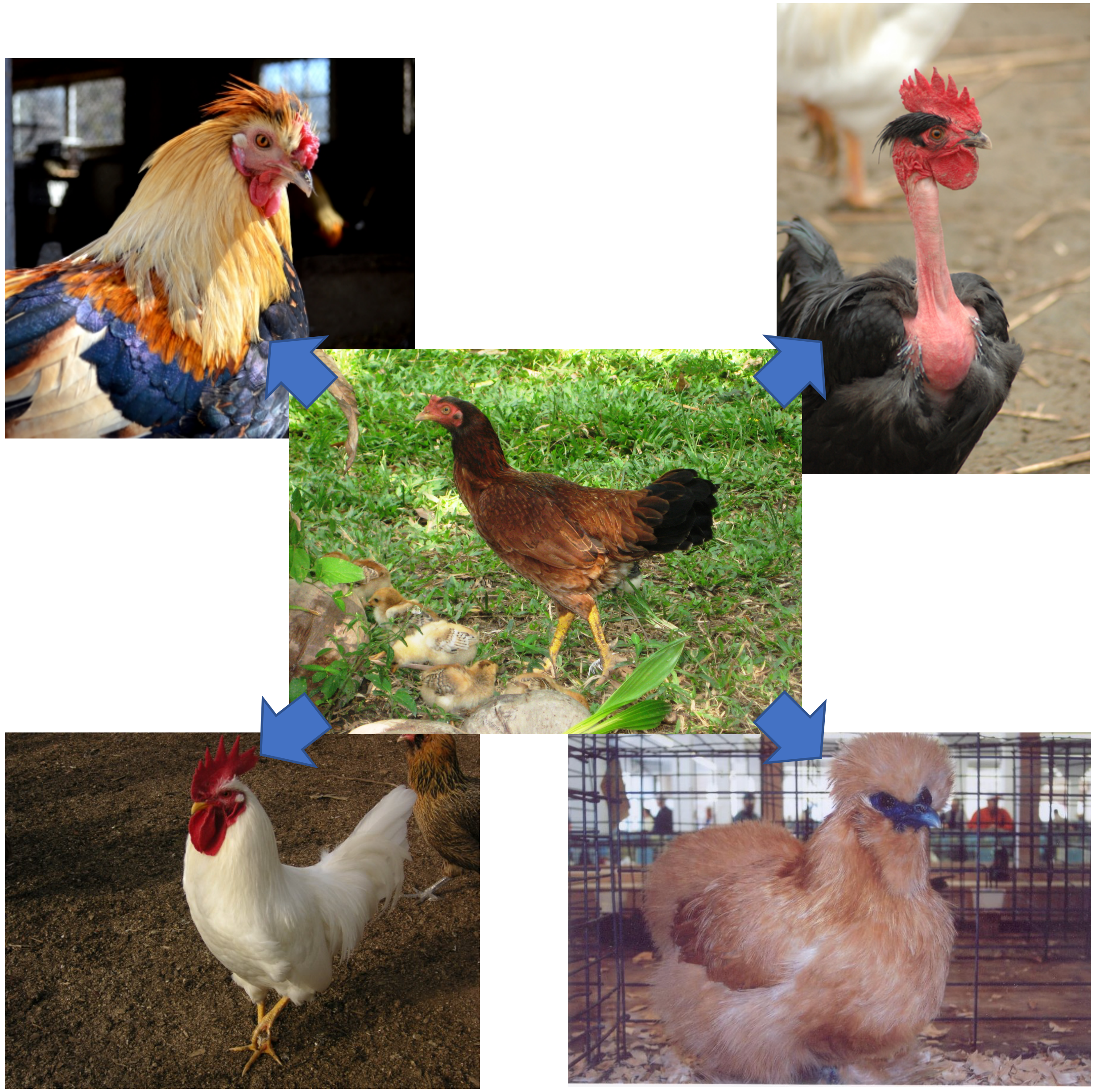
No one will lose points on the stability analysis in part 8 of Project 1.

Genetic Algorithms

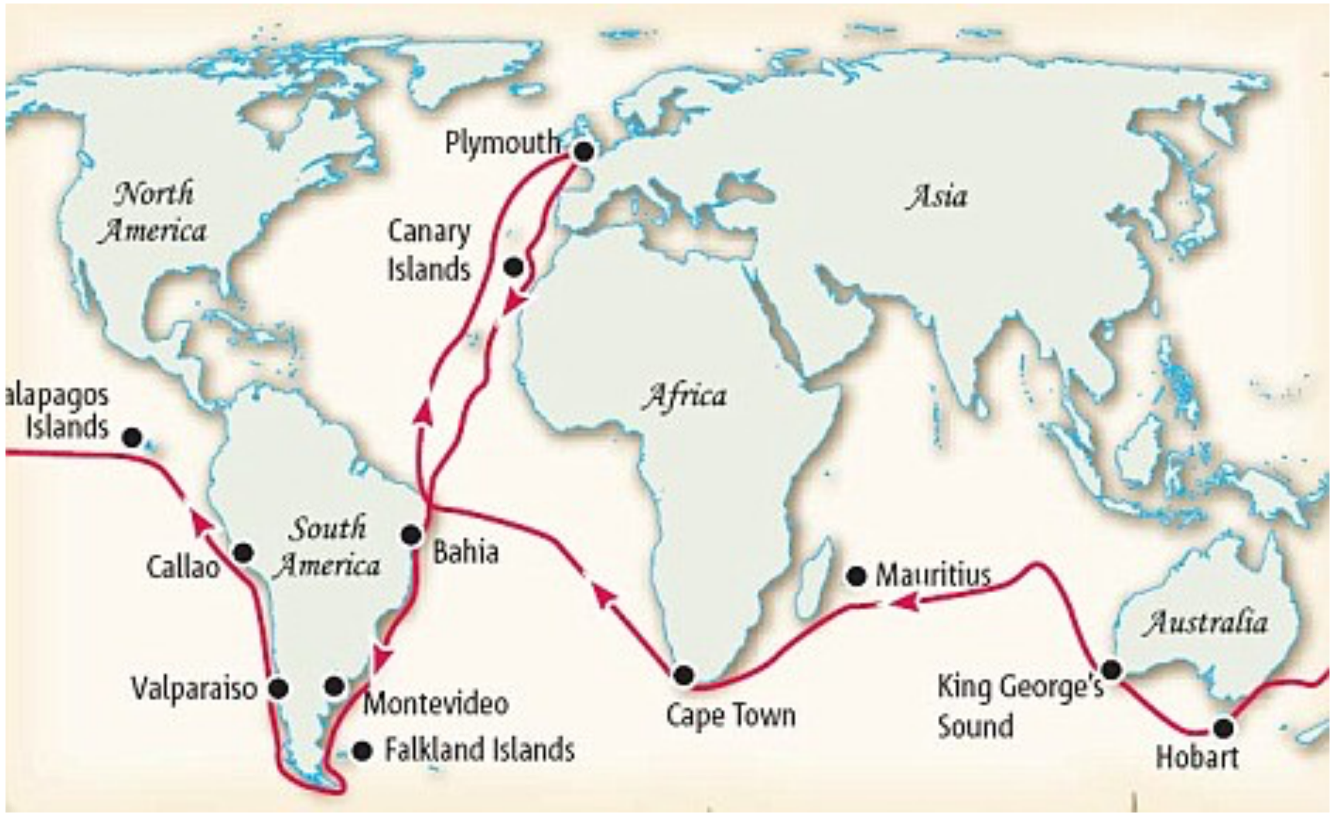
Complex Adaptive Systems

Biological Evolution

- New species develop from older species
- Old idea (versions since Anaxamander and Empedocles)
- Breeding of plants and animals



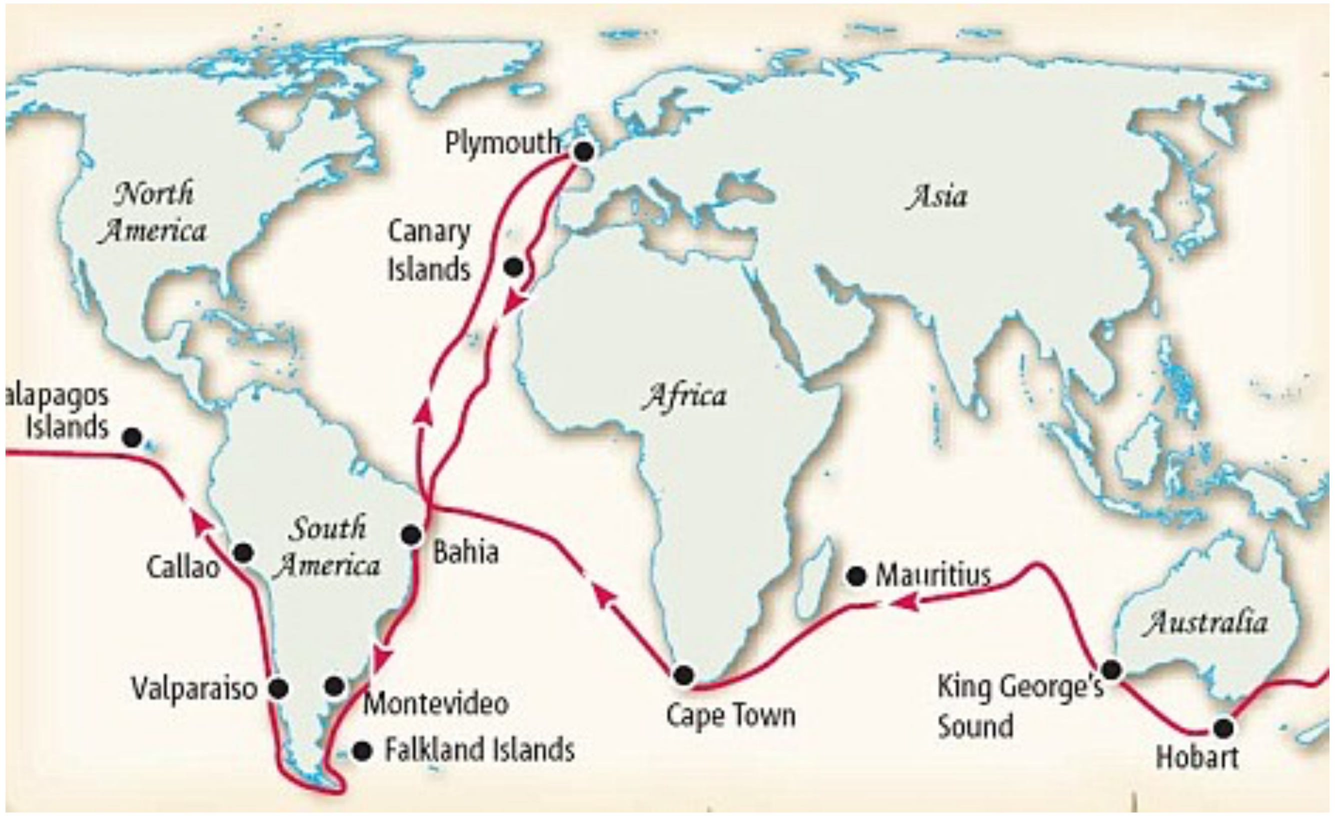
Biological Evolution: Charles Darwin



Voyage of the Beagle
5 years starting in 1831
Darwin trained for priesthood and medicine
but joined the Beagle as a naturalist.

Charles Darwin

Biological Evolution: Charles Darwin



Charles Darwin

Was able to observe:

- geological evidence of the earth’s age
- fossils of large mammals in South America
- diversity of adaptive living forms on the Galapagos

Darwinian Evolution

It is interesting to contemplate a tangled bank, clothed with many plants of many kinds, with birds singing on the bushes, with various insects flitting about, and with worms crawling through the damp earth, and to reflect that these elaborately constructed forms, so different from each other, and dependent upon each other in so complex a manner, have all been produced by laws acting around us.

--- Charles Darwin

“Nothing in Biology makes sense, except in the light of evolution.”
T. Dobzhansky

Darwinian Evolution

- Charles Darwin, 1859, The Origin of Species

3 key requirements:

- Exponential growth of populations
- Struggle for existence: Limited Capacity for any population
- Variable, heritable survival and reproduction

Darwinian Evolution

- The unity of life: all species have descended from other species
- Builds on Malthus, *An Essay on the Principle of Population,* 1798
- Domestic breeding shows hereditary modification is possible
- Fitness is a characteristic of individuals
- Natural Selection operates on populations
- Fitness is defined only for a particular environment Environments always change
- Species form the selective environments of other species

Darwinian Evolution

Natural selection

– is often slow, but arms races result in complex, wonderful, bizarre (and stupid) things

– can lead to cooperation

– (largely) based on the fitness of reproductive individuals

- Natural selection is not

– learned behavior passed on

– is not goal directed beyond producing sucessful offspring

Darwinian Evolution

Natural selection

– is often slow, but arms races result in complex, wonderful, bizarre (and stupid) things

– can lead to cooperation

– (largely) based on the fitness of reproductive individuals

- Natural selection is not

– learned behavior passed on

– is not goal directed beyond producing sucessful offspring

Darwinian Evolution

Heritability:

- Darwin did not know a mechanism for this.

Are offspring merely the average of their parents?

Darwinian Evolution

Heritability:

- Darwin did not know a mechanism for this.

Are offspring merely the average of their parents?

Darwinian Evolution

Gregor Mendel
Austrian Monk and Botinist

Heritability:

- Darwin did not know a mechanism for this.

Are offspring merely the average of their parents?

Mendalian Genetics

Heritability is:

Discrete!

Not just an averaging.

Probabilistic but
Well defined and predictable.

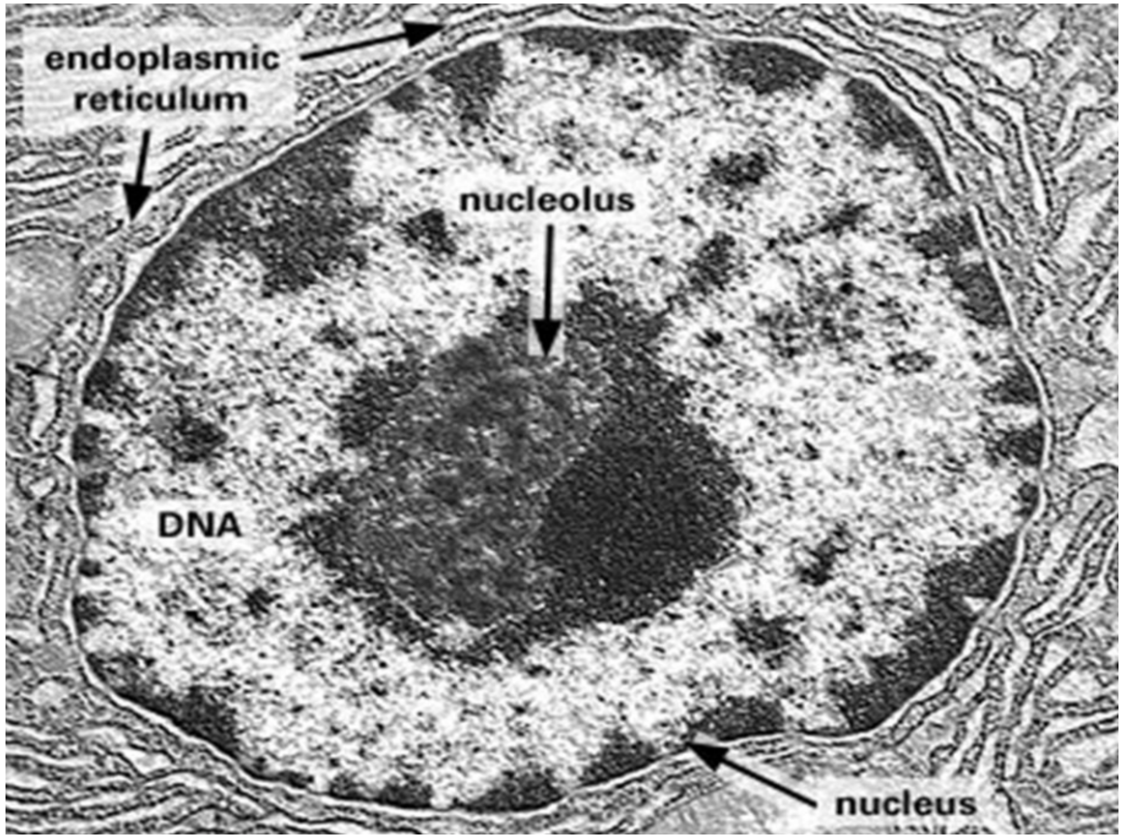
Gregor Mendel
Austrian Monk and Botinist

Genetics

Still no mechanism.

Gregor Mendel
Austrian Monk and Botinist

Genetics
DNA



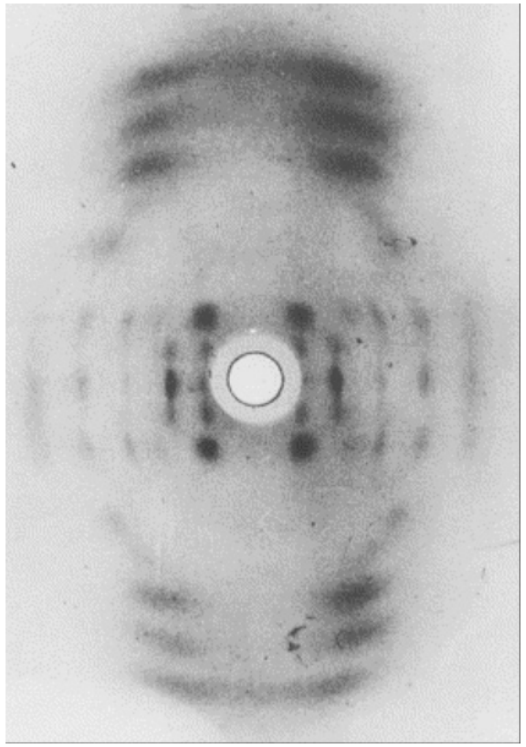
Friedrich Miescher
Swiss Physician

1869 Isolated a new protein from the nucleus of cells.

Thought it was related to heredity but too simple

Genetics

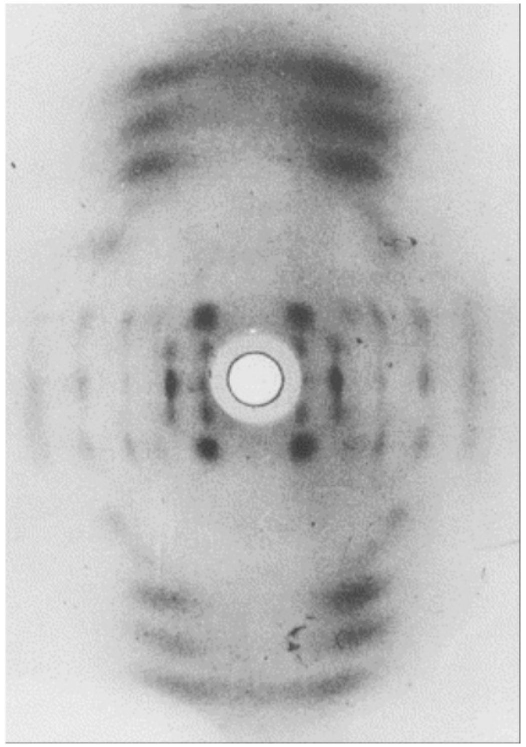
DNA.
Crystalography



Rosalind Frankin, Cambridge
Invented the technique

Genetics

DNA.
Crystalography

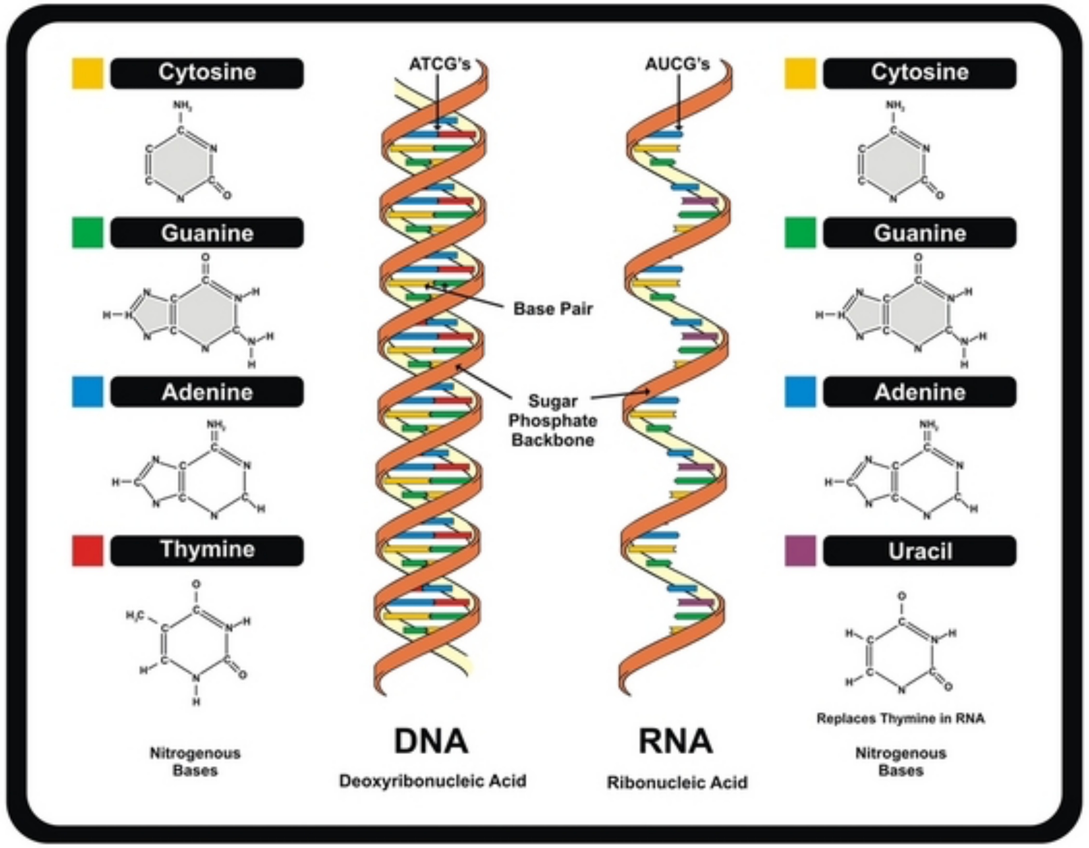


Rosalind Frankin
Invented the technique

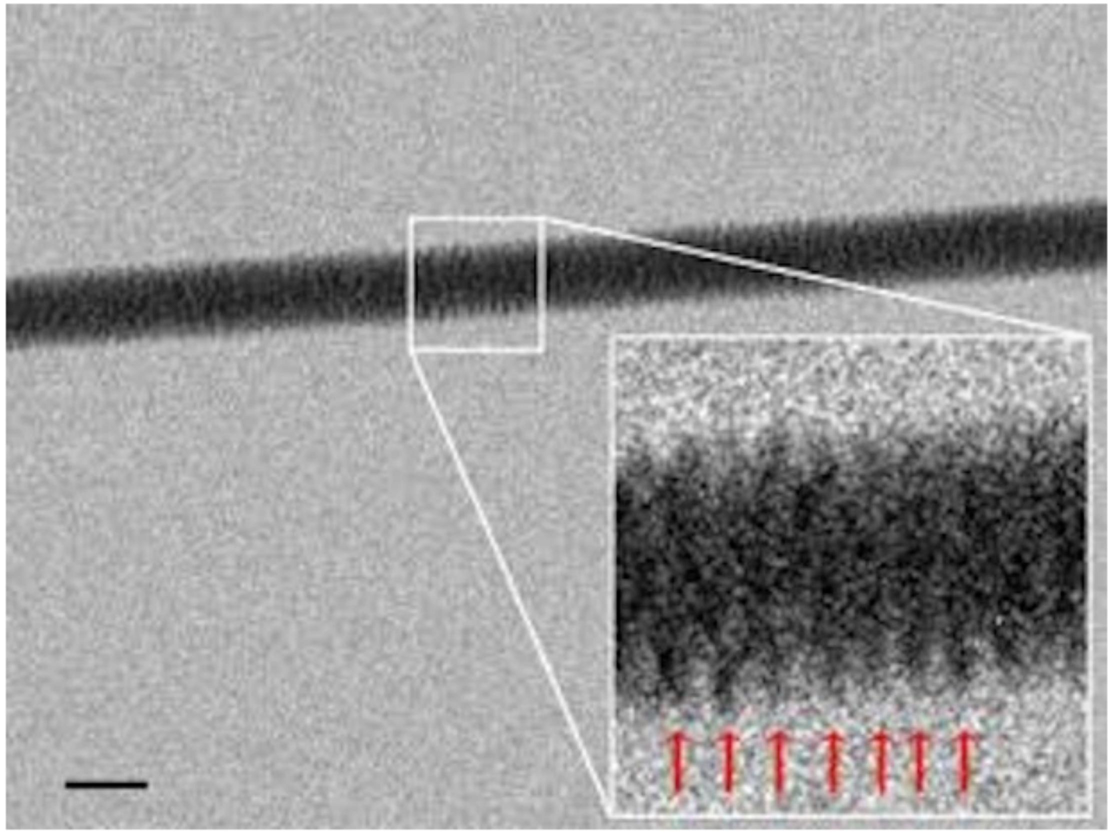
James Watson & Francis Crick
Interpreted the results

Spiral
Structure

DNA – Information Storage



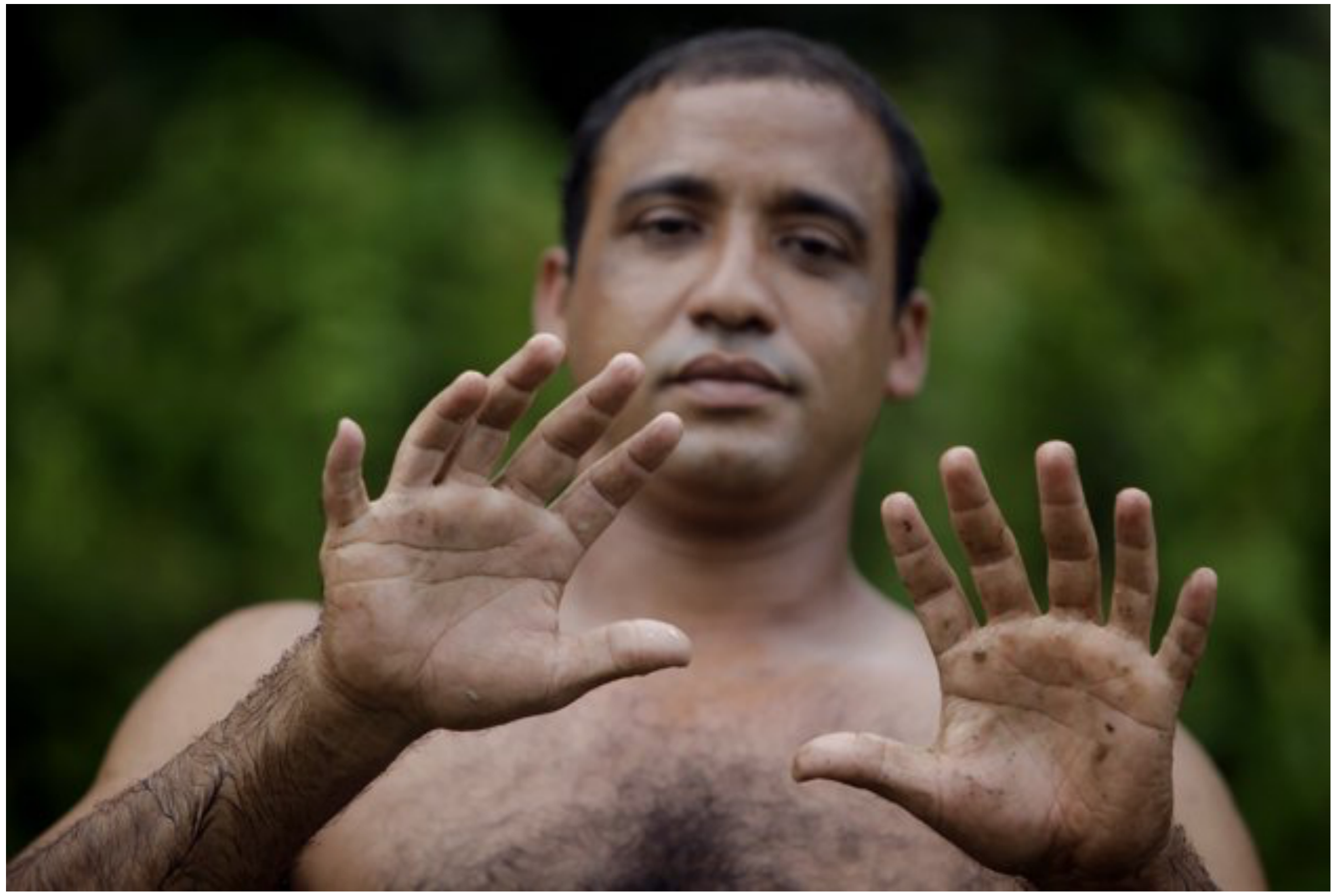
Human: 2.9 billion base pairs = 725 MB data



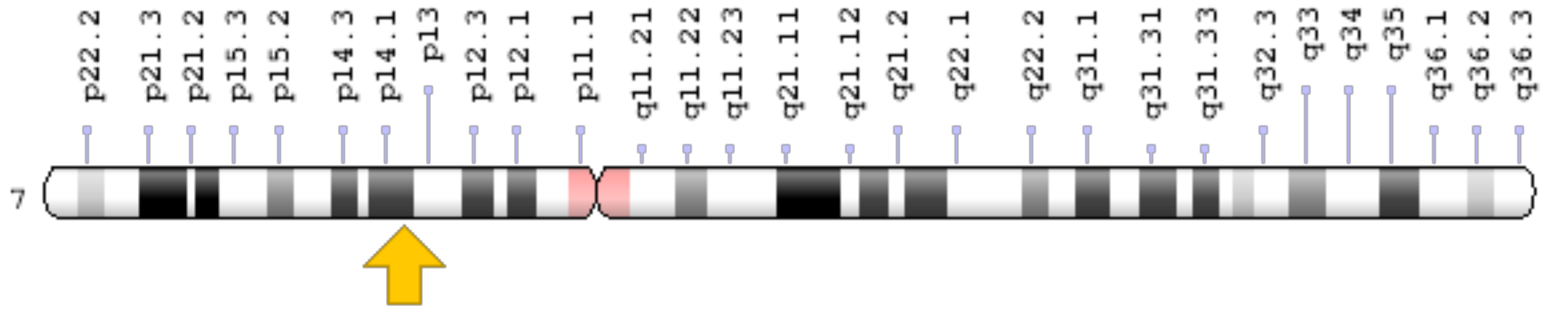
Direct Imaging of DNA Fibers: The Visage of Double Helix, Nanoletters, 2012
Francesco Gentile†, Manola Moretti†, Tania Limongi†§, Andrea Falqui, Giovanni Bertoni, Alice Scarpellini⊥, Stefania Santoriello†, Luca Maragliano§, Remo Proietti Zaccaria†, and Enzo di Fabrizio

Mutation

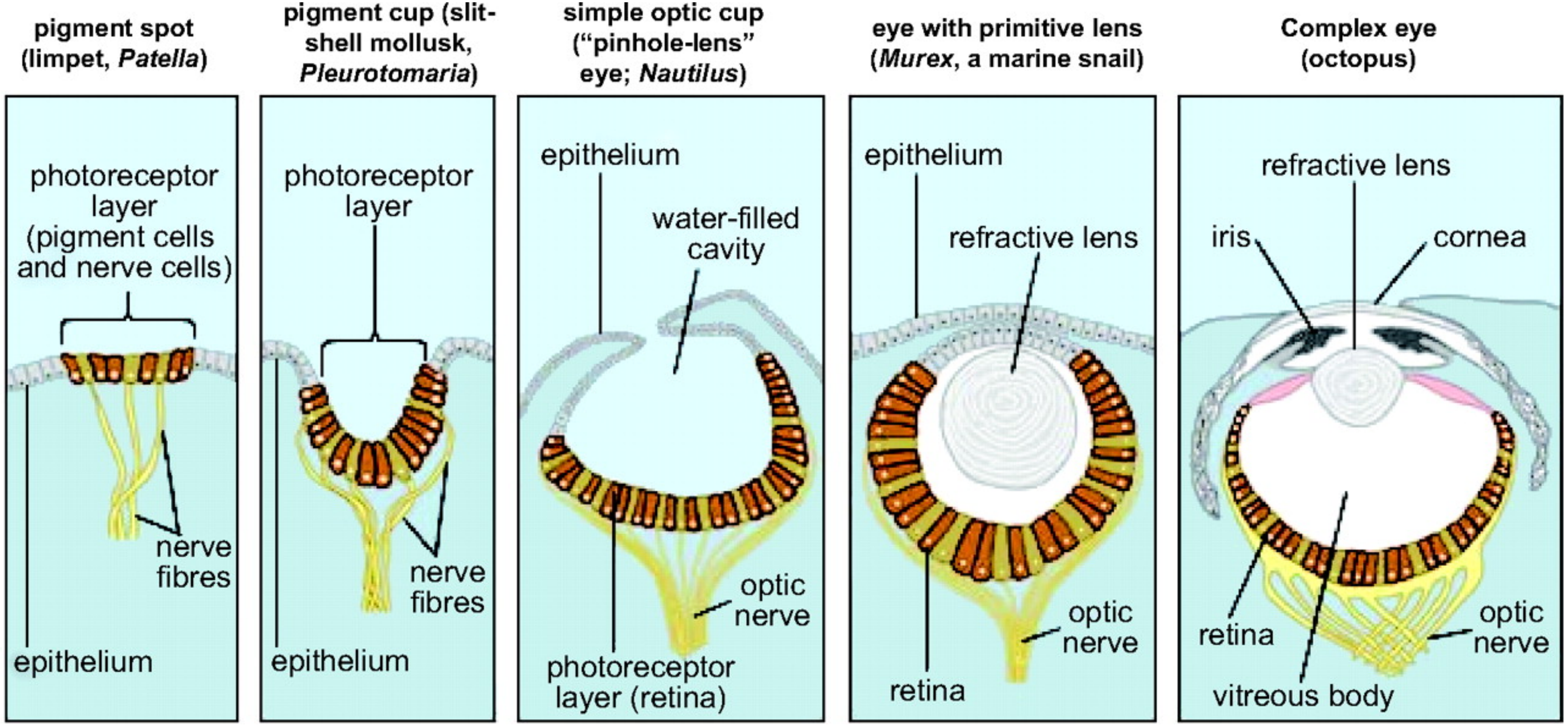
Polydactyly, 24 digits,
Gene GLI3, “Sonic Hedgehog” pathway



Yoandri Hernandez Garrido, 37



Mutation


pigment spot (limpet, *Patella*)
photoreceptor layer (pigment cells and nerve cells)
nerve fibres
epithelium
pigment cup (slit-shell mollusk, *Pleurotomaria*)
photoreceptor layer
nerve fibres
epithelium
simple optic cup ("pinhole-lens" eye; *Nautilus*)
epithelium
water-filled cavity
optic nerve
photoreceptor layer (retina)
eye with primitive lens (*Murex*, a marine snail)
epithelium
refractive lens
optic nerve
retina
Complex eye (octopus)
refractive lens
iris
cornea
retina
optic nerve
vitreous body

DNA – Information Storage

We need to define:

Population size: N

Genome: $G \in \vec{\mathbb{R}}$

Population: $P \in$ Matrix of size $N \times |G|$ with entries $\in \mathbb{R}$

Mutation operator M: $\vec{\mathbb{R}} \rightarrow \vec{\mathbb{R}}$

Selection Operator S: $P_{N\times|G|}(\mathbb{R}) \rightarrow P_{N\times|G|}(\mathbb{R})$

Fitness function: F: $\vec{\mathbb{R}} \rightarrow \mathbb{R}$

- Great! Now how do we use these
- principles to solve problems?

- Code it up…

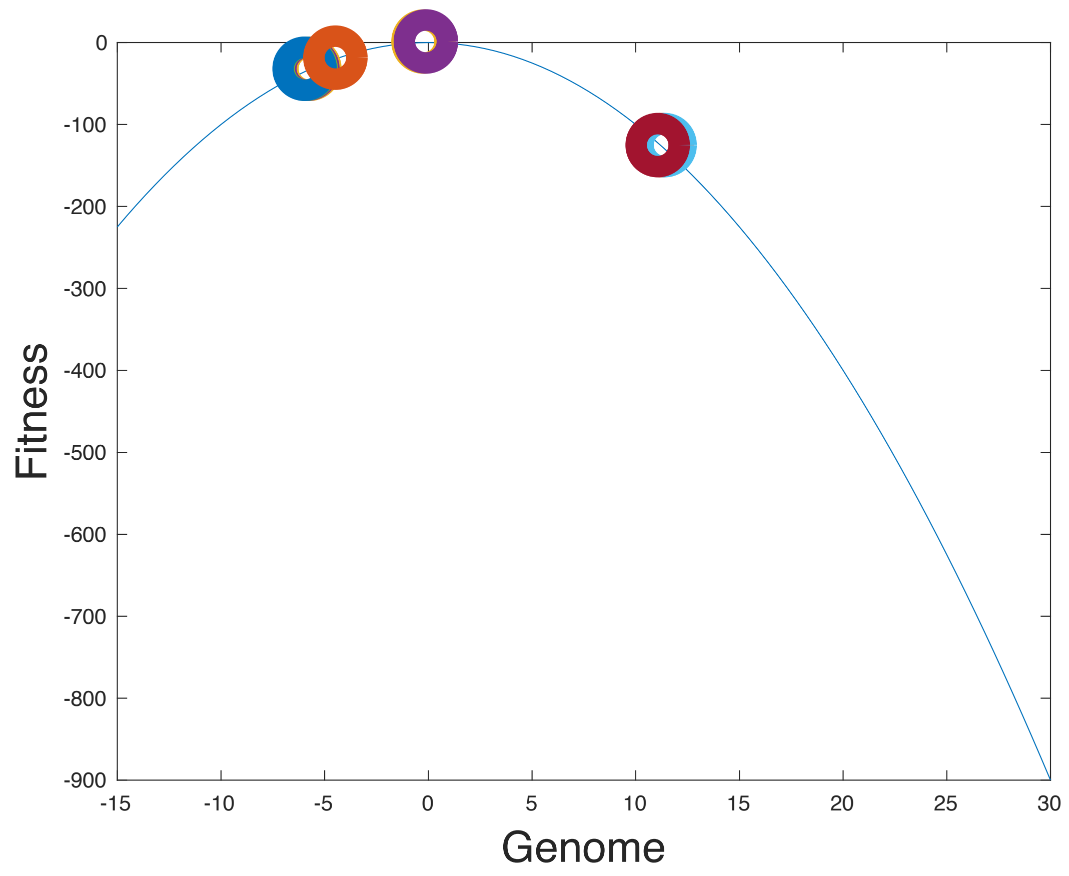
```
classdef GAIndividual
    properties
        genome;
        fitness;
    end
```

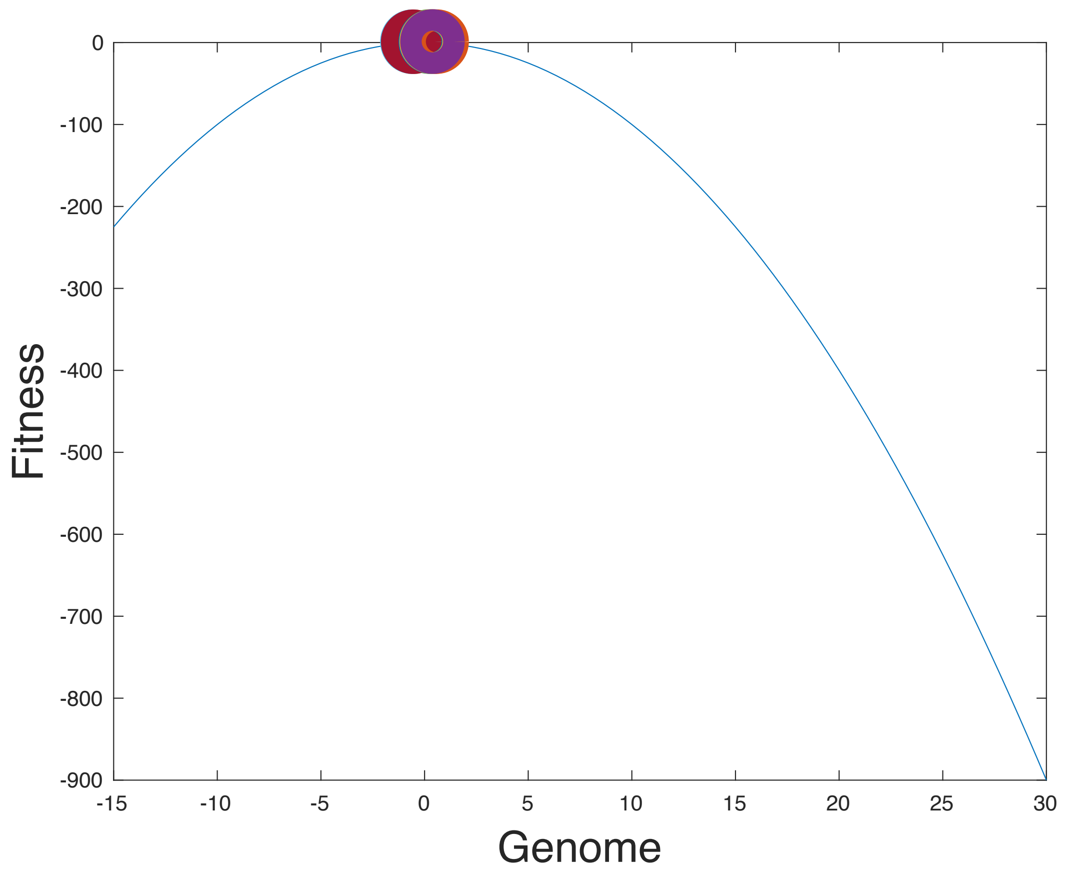
```
methods
    % Make a deep copy of this object
    function new = copy(this)
            % Instantiate new object of the same class.
            new = feval(class(this));

            % Copy all non-hidden properties.
            p = properties(this);
            for i = 1:length(p)
                new.(p{i}) = this.(p{i});
            end
        end
    end
end
```

```
% Simple GA that tries to find the maximum value
% of a 1D Function F.
% Example: GA(10, 0.5, 10, @(X) sin(X).*X.^2, 2.2);
function GA(pop_size, mutation_rate, num_generations, F,
animation_delay)

genome_min = -15; genome_max = 30;
genome_space = linspace(genome_min,genome_max);

% Keep some statistics about the evolution
mean_fitness_over_time = [];
max_fitness_over_time = [];
min_fitness_over_time = [];

population = repmat(GAIndividual, 1, pop_size);

plot(genome_space, F(genome_space));
xlabel('Genome', 'Fontsize', 20);
ylabel('Fitness', 'Fontsize', 20);


hold on
for i = 1:pop_size
    population(i).fitness = 0;
    population(i).genome = rand()*(genome_max-
genome_min)+genome_min;


p(i) = plot(population(i).genome,population(i).fitness,
'o','LineWidth', 10, 'MarkerSize', 20);
end
hold off
```

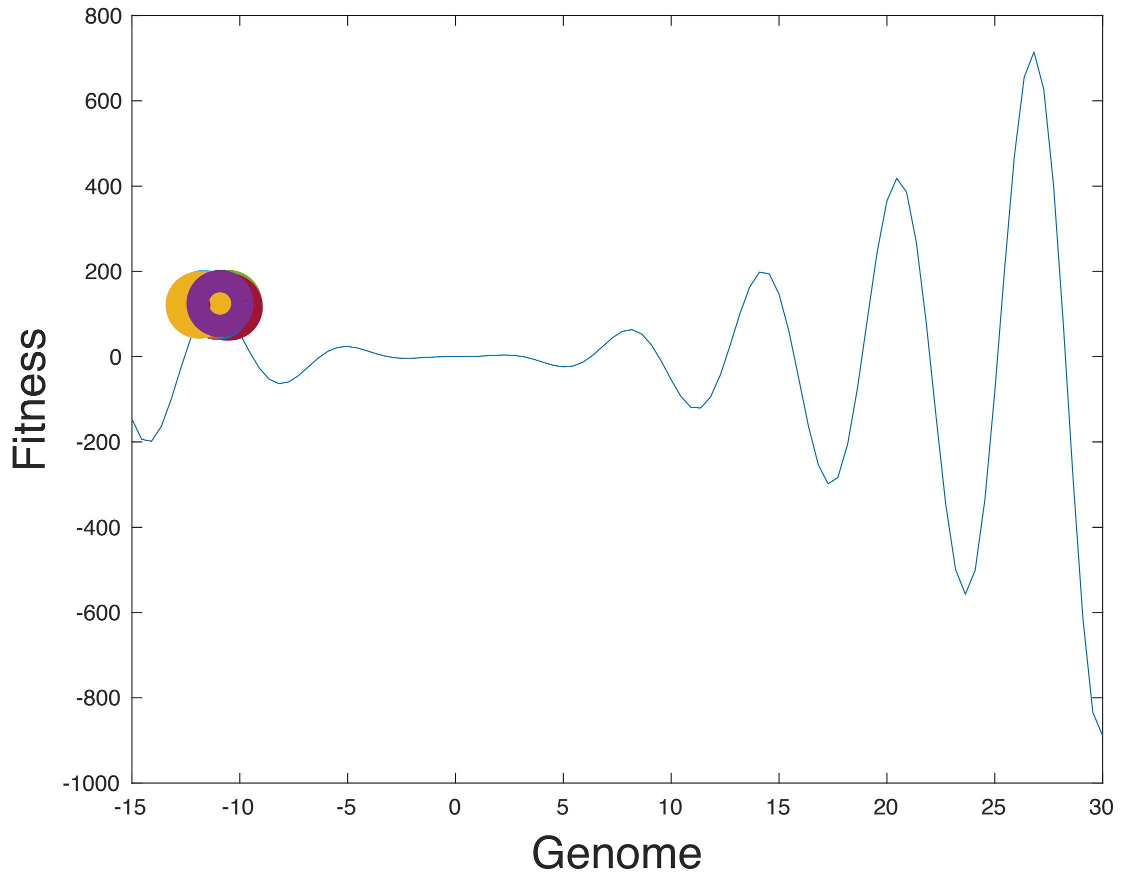
```
for generation = 1:num_generations

    % Mutate population
    for i = 1:pop_size
        candidate = population(i).genome + normrnd(0,mutation_rate);
        if candidate < genome_max && candidate > genome_min
            population(i).genome = candidate;
        end
    end

    % Animation
    for i = 1:pop_size
        p(i).XData = population(i).genome;
        p(i).YData = population(i).fitness;
    end
    pause(animation_delay);
    drawnow

    % Evaluate fitness
    for i = 1:pop_size
        population(i).fitness = F(population(i).genome);
    end

    % update evolution stats
    mean_fitness_over_time = [mean_fitness_over_time,mean([population.fitness])];
    max_fitness_over_time = [max_fitness_over_time, max([population.fitness])];
    min_fitness_over_time = [min_fitness_over_time, min([population.fitness])];

    % Sort population by fitness (lowest to highest)
    [~,sorted_indicies]=sort([population.fitness]);
    population=population(sorted_indicies);

    % Replace bottom half of population with random individuals from the top
    % half
    cutoff = ceil(pop_size/2); % ceiling in case the population size is odd
    for i = 1:cutoff
       population(i) = copy(population(cutoff+randi(cutoff)));
    end
end
```

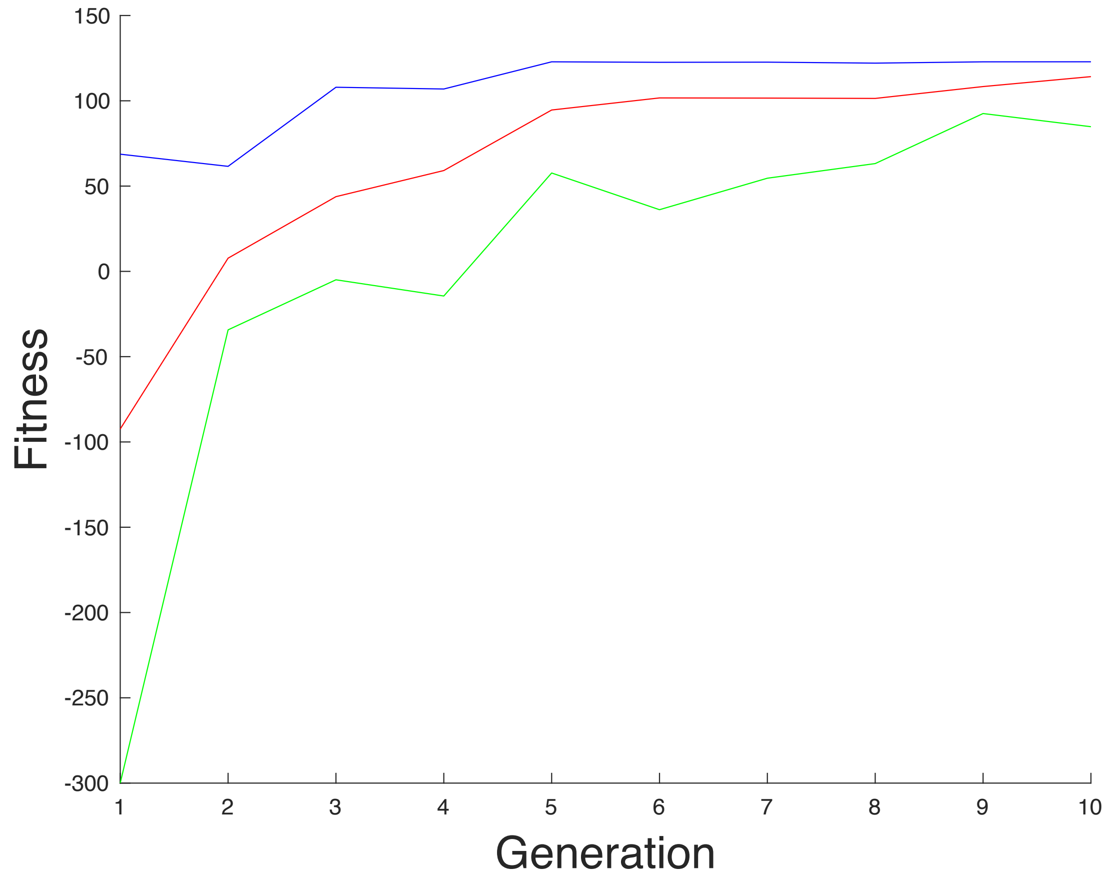
GA(10, 0.05, 10, @(X) -X.^2, 0.2);





GA(10, 0.5, 10, @(X) sin(X).*X.^2, 2.2);

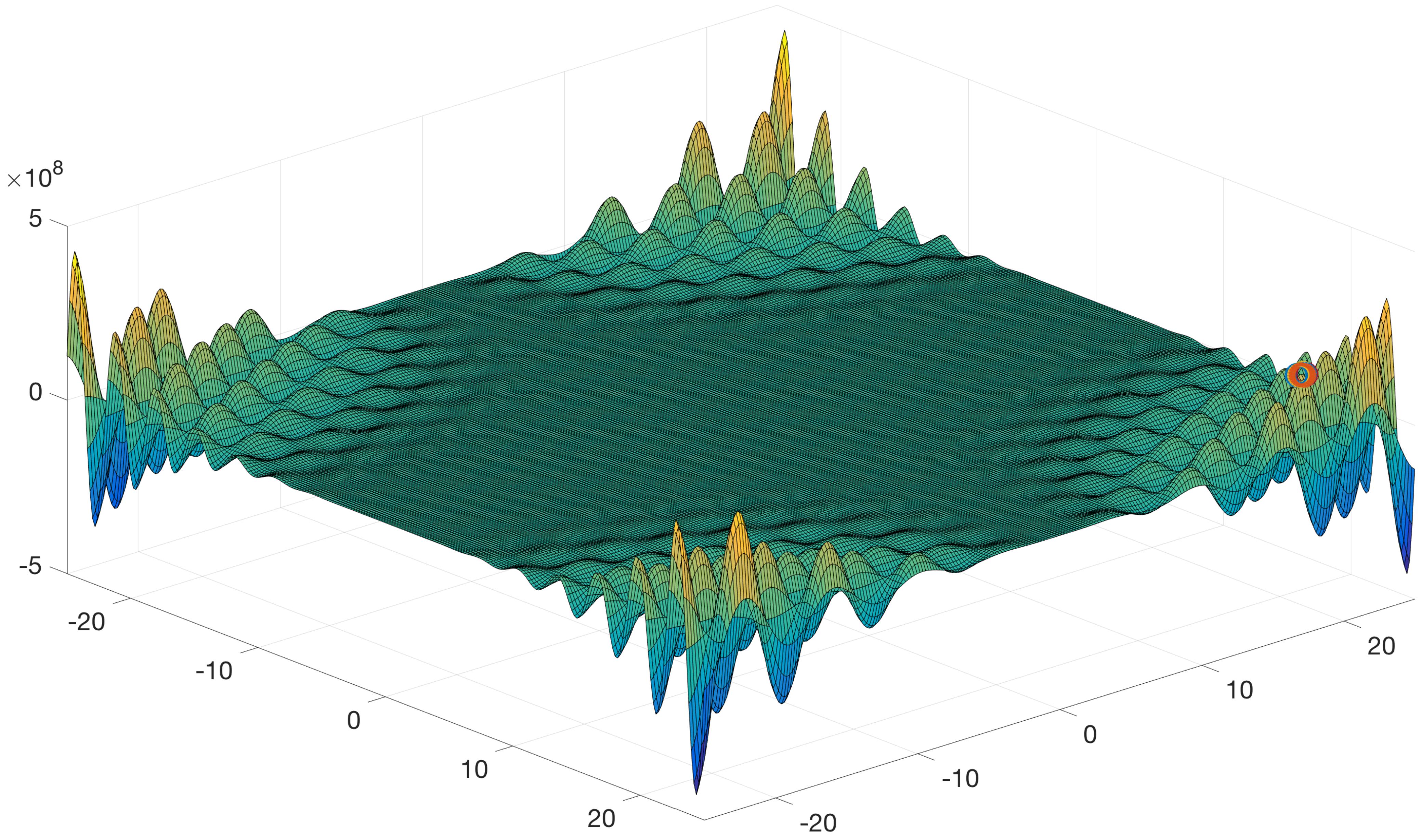




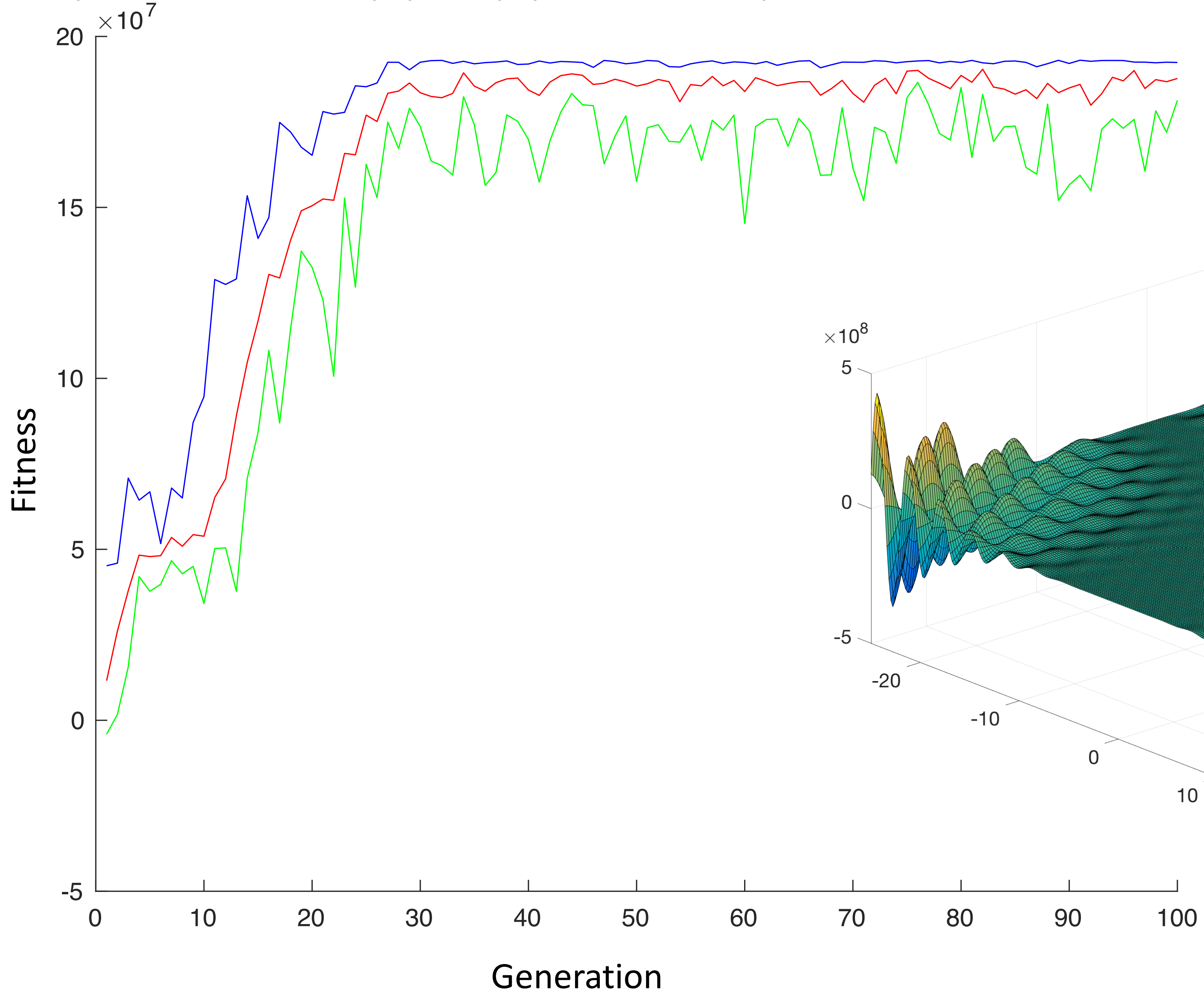
```
function GA(pop_size, mutation_rate, num_generations, F, animation_delay)

genome_min = -25; genome_max = 25;
genome_min = -25; genome_max = 25;
[genome_space_x, genome_space_y] = meshgrid(genome_min:0.2:genome_max);

% Keep some statistics about the evolution
mean_fitness_over_time = [];
max_fitness_over_time = [];
min_fitness_over_time = [];

surf(genome_space_x, genome_space_y, F(genome_space_x, genome_space_y));

population = repmat(GAIndividual, 1, pop_size);

hold on
for i = 1:pop_size

    population(i).genome = rand(1,2)*(genome_max-genome_min)+genome_min;
    population(i).fitness = F(population(i).genome(1),
population(i).genome(2));

    % For animation
    p(i) = plot3(population(i).genome(1), population(i).genome(2),
0,'o','LineWidth', 10, 'MarkerSize', 20);
end
hold off

for generation = 1:num_generations
    % Mutate population
    for i = 1:pop_size
        candidate_x = population(i).genome(1) + normrnd(0,mutation_rate);
        candidate_y = population(i).genome(2) + normrnd(0,mutation_rate);

        if candidate_x < genome_max && candidate_x > genome_min
            population(i).genome(1) = candidate_x;
        end

        if candidate_y < genome_max && candidate_y > genome_min
            population(i).genome(2) = candidate_y;
        end
    end
```

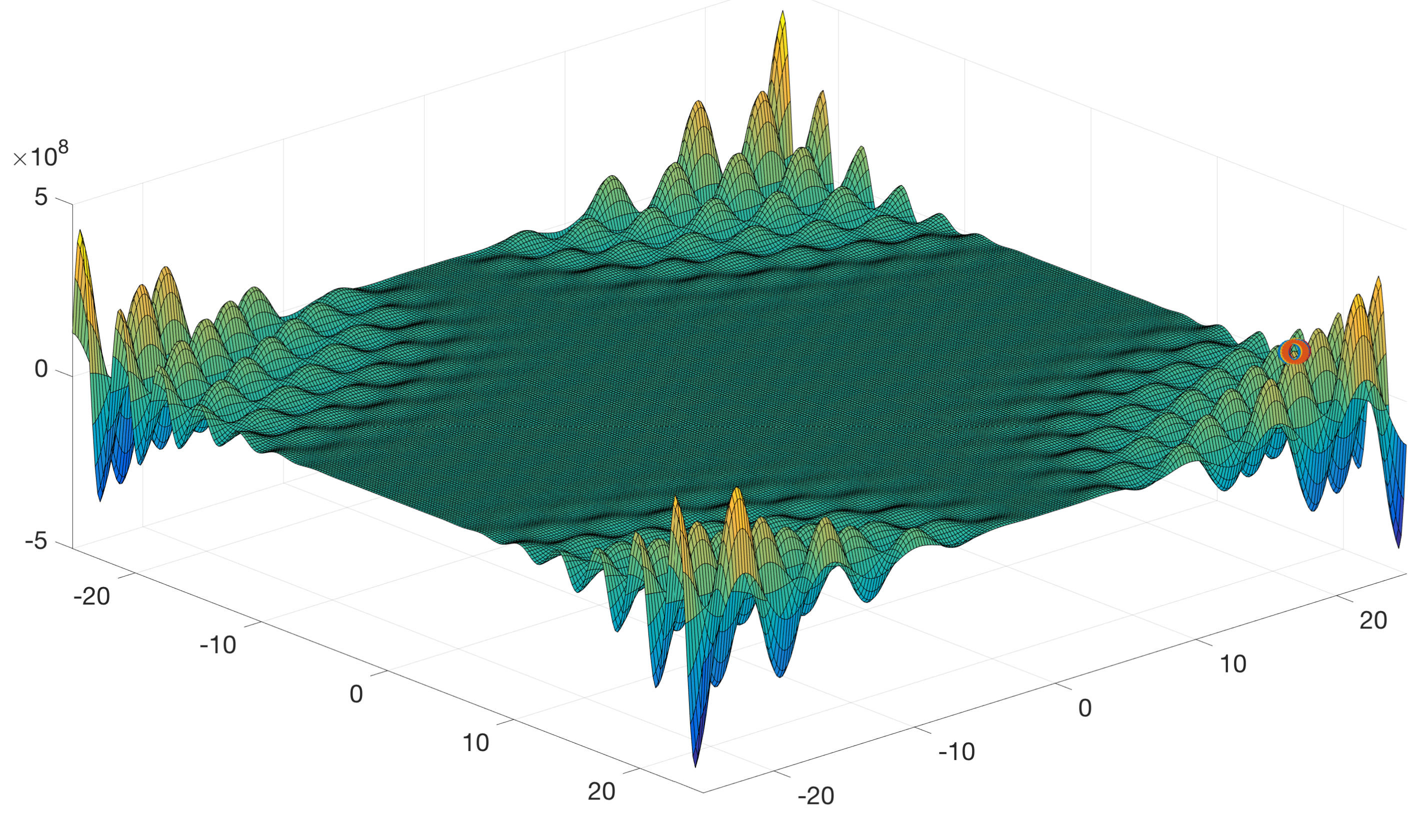
```
% Animation
    for i = 1:pop_size
        p(i).XData = population(i).genome(1);
        p(i).YData = population(i).genome(2);
        p(i).ZData = population(i).fitness;
    end
    pause(animation_delay);
    drawnow


    % Evaluate fitness
    for i = 1:pop_size
        population(i).fitness = F(population(i).genome(1),
population(i).genome(2));
    end

 % update evolution stats
 mean_fitness_over_time = [mean_fitness_over_time, mean([population.fitness])];
 max_fitness_over_time = [max_fitness_over_time, max([population.fitness])];
 min_fitness_over_time = [min_fitness_over_time, min([population.fitness])];

    % Sort population by fitness (lowest to highest)
    [~,sorted_indicies]=sort([population.fitness]);
    population=population(sorted_indicies);

    % Replace bottom half of population random individuals from the top
    % half
    cutoff = ceil(pop_size/2); % floor in case the population size is odd
    for i = 1:cutoff
        population(i) = copy(population(cutoff+randi(cutoff)));
    end
end

figure
hold on
plot(1:generation, mean_fitness_over_time, 'r-');
plot(1:generation, max_fitness_over_time, 'b-');
plot(1:generation, min_fitness_over_time, 'g-');
hold off

End
```

```
GA(10, 0.5, 10, @(X) sin(X).*X.^2, 2.2);
```

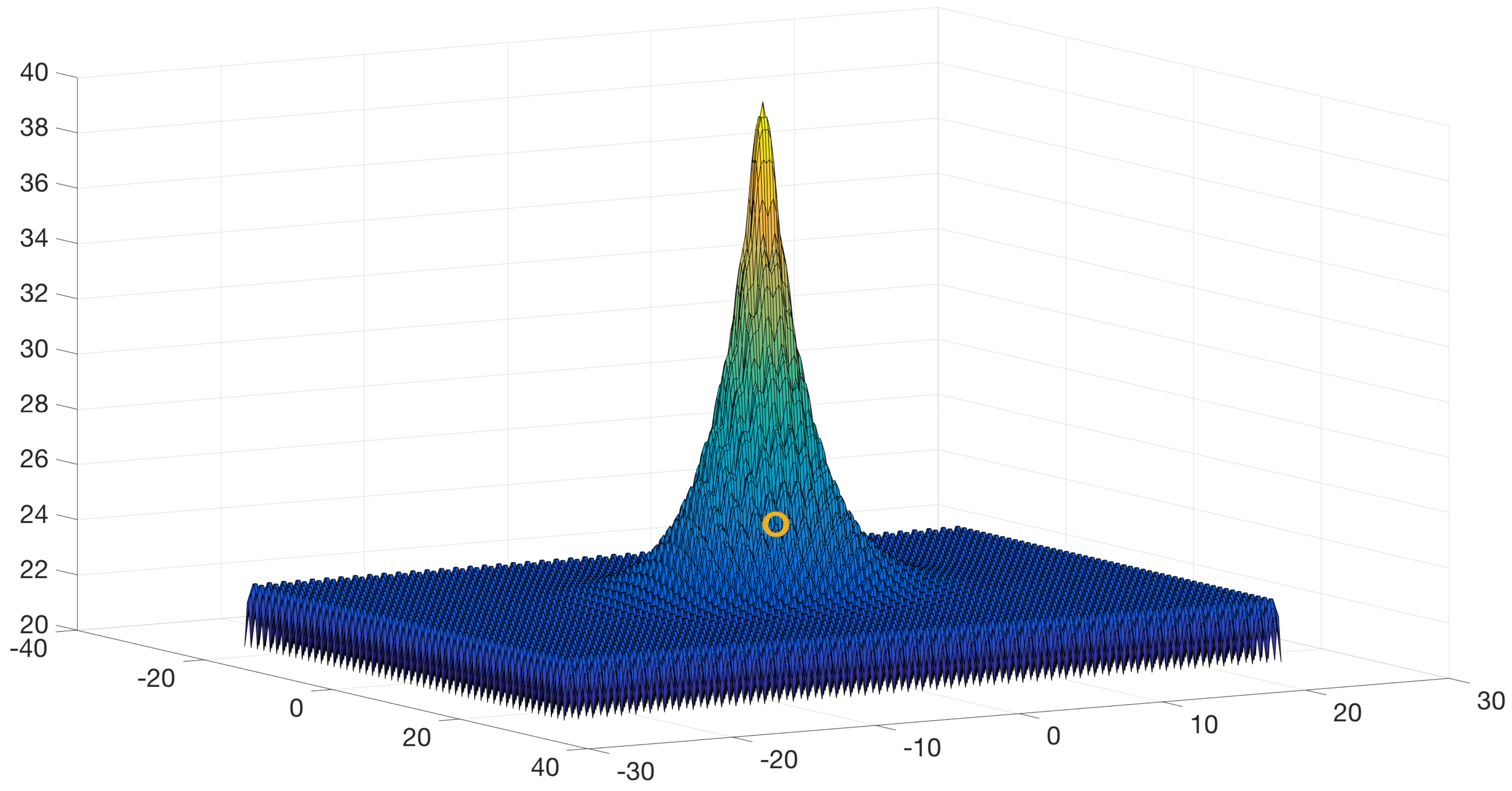


GA(10, 0.5, 10, @(X) sin(X).*X.^2, 2.2);

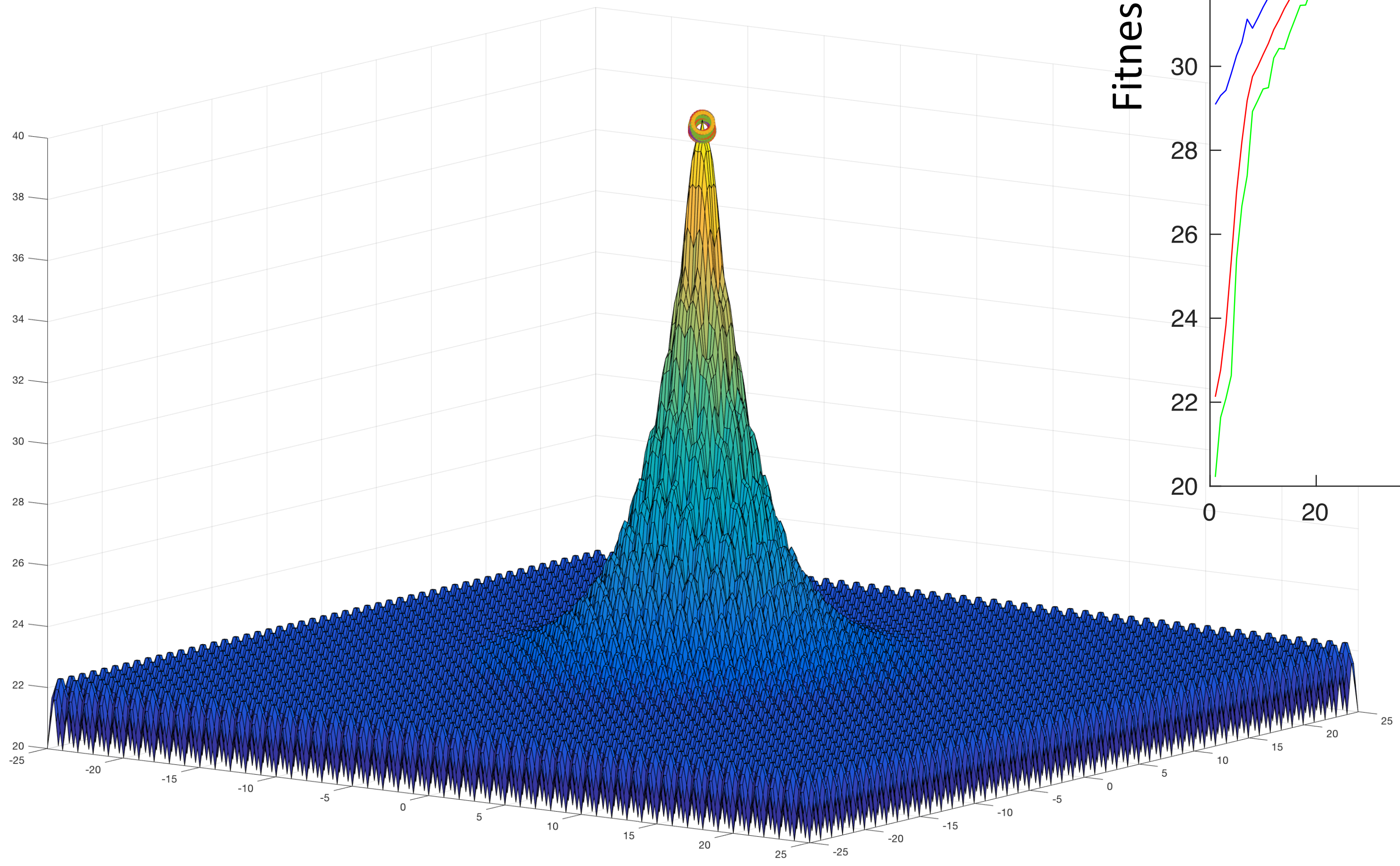


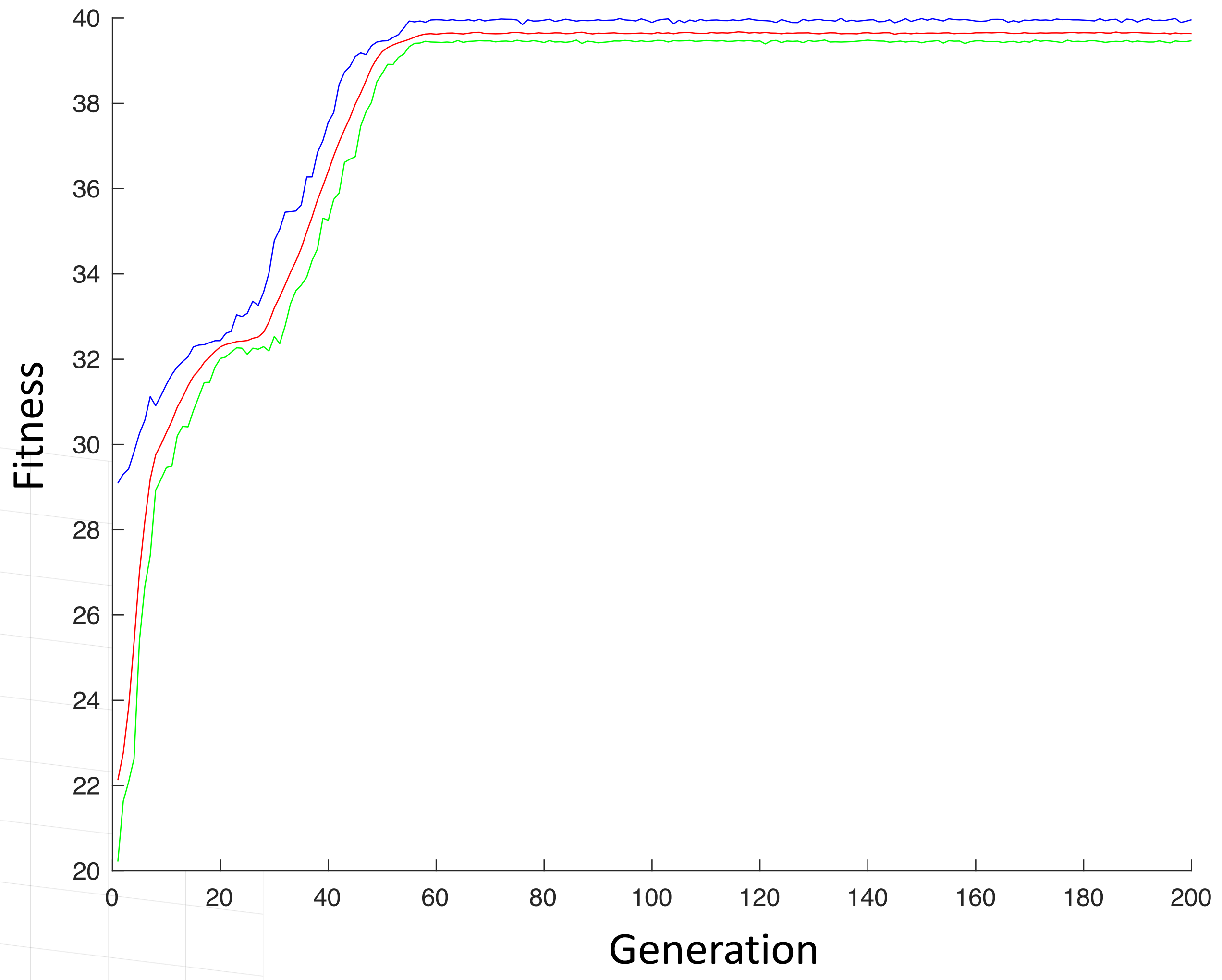


GA(10, 0.5, 10, @ackley, 0.2);



GA(100, 0.5, 100, @ackley, 0.2);





Curse of Dimensionality

We can place individuals in a 1D search space so that the whole
region trivially well covered.

But when more dimensions are added an exponential increase
in population size is required. We have to get better at evolving solutions in
high-dimensional spaces.

Genetic Recombination

1. Sound off along the rows (write down your number).

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.
4. Pass the paper to someone other than the person who gave it to you.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.
4. Pass the paper to someone other than the person who gave it to you.
5. Pass the paper to someone whose number was even.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.
4. Pass the paper to someone other than the person who gave it to you.
5. Pass the paper to someone whose number was even.
6. If you have more than one piece of paper pass the extras to someone else whose number is even.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.
4. Pass the paper to someone other than the person who gave it to you.
5. Pass the paper to someone whose number was even.
6. If you have more than one piece of paper pass the extras to someone else whose number is even.
7. If you have a piece of paper add your name to it.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.
4. Pass the paper to someone other than the person who gave it to you.
5. Pass the paper to someone whose number was even.
6. If you have more than one piece of paper pass the extras to someone else whose number is even.
7. If you have a piece of paper add your name to it.
8. Pass the papers to Bianca.

Genetic Recombination

1. Sound off along the rows.
2. If your number was odd write your name on a piece of paper.
3. Pass that piece of paper to a neighbour.
4. Pass the paper to someone other than the person who gave it to you.
5. Pass the paper to someone whose number was even.
6. If you have more than one piece of paper pass the extras to someone else whose number is even.
7. If you have a piece of paper add your name to it.
8. Pass the papers to Bianca.
9. Welcome to your new project groups.

Logistics

- Reviews are due Monday
- Project 2 will be assigned Monday

Massive Parallelism in Biology

Some bacteria divide every 20 minutes

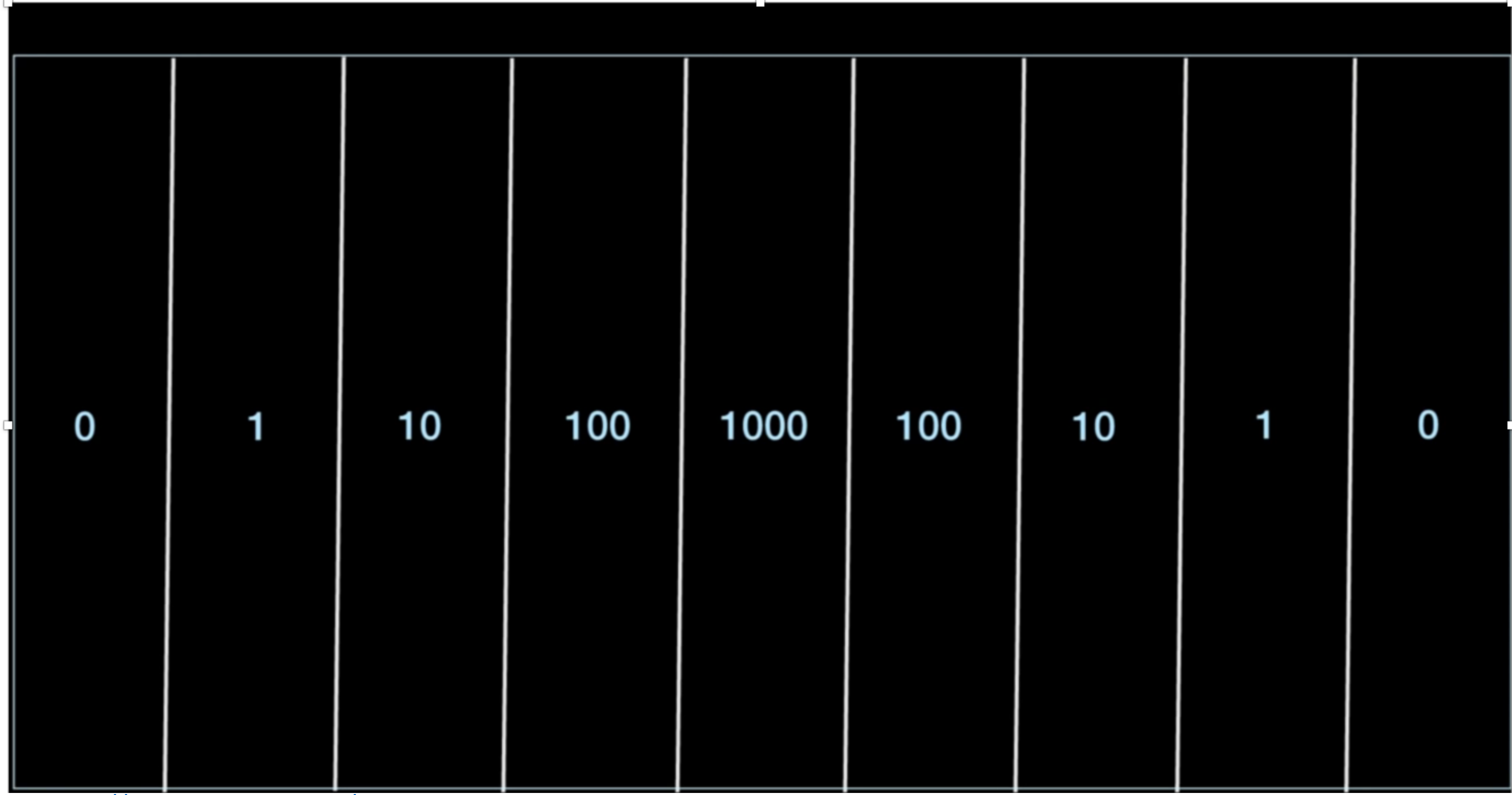
(under laboratory conditions)

2^n bacteria after n 20 min periods.

$n = 72$ in one day.

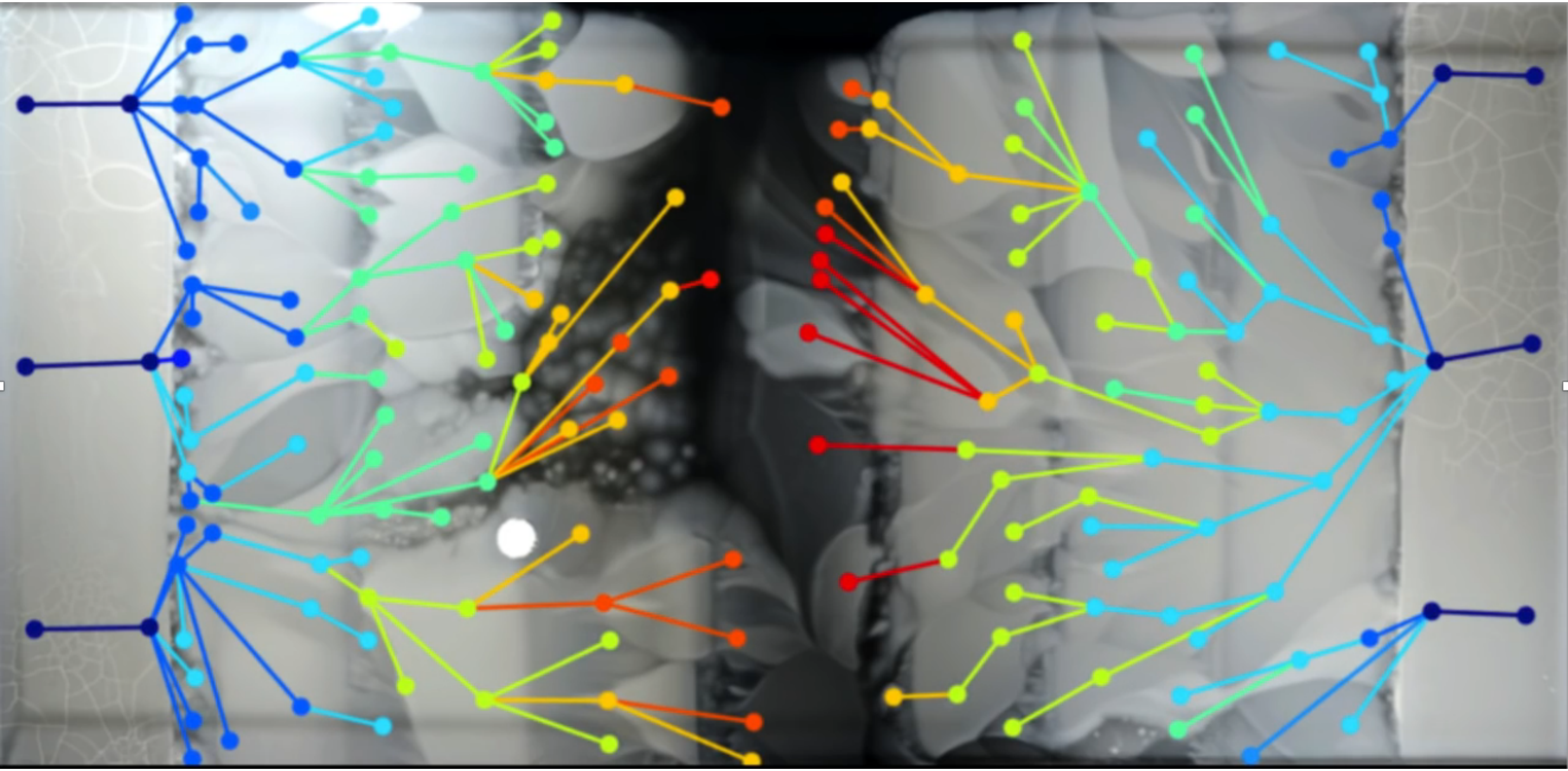
```
>> 2^72
ans =   4.7224e+21
```

https://www.youtube.com/watch?v=plVk4NVIUh8



https://www.youtube.com/watch?v=plVk4NVIUh8

https://www.youtube.com/watch?v=plVk4NVIUh8

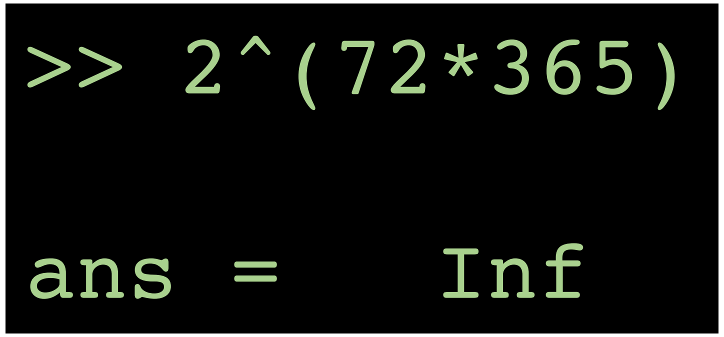


https://www.youtube.com/watch?v=plVk4NVIUh8

Some bacteria divide every 20 mins

2^n bacteria after n 20 min periods.

$n = 72 \times 365$ in one year.



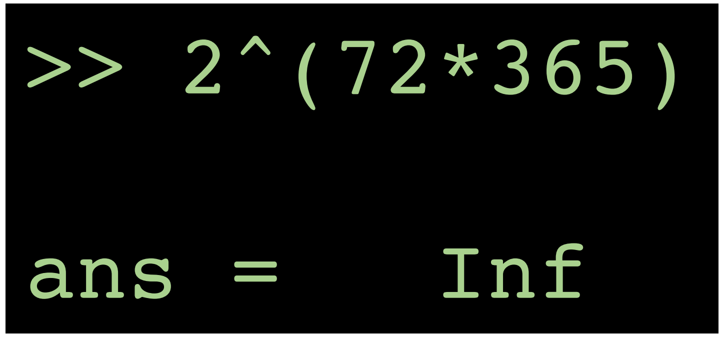
```
>> 2^(72*365)

ans =    Inf
```

Atoms in the observable universe: 10^{79} to 10^{81}.

Some bacteria divide every 20 mins

2^n bacteria after n 20 min periods.

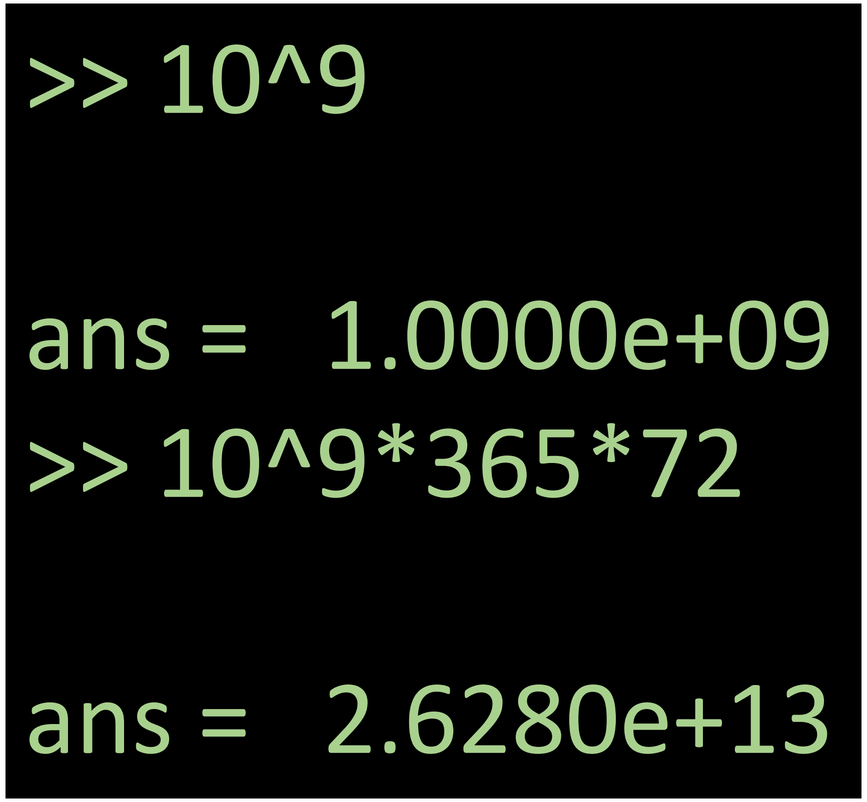
$n = 72 \times 365$ in one year.



```
>> 2^(72*365)

ans =    Inf
```

Life originated some 3.5×10^9 years ago.

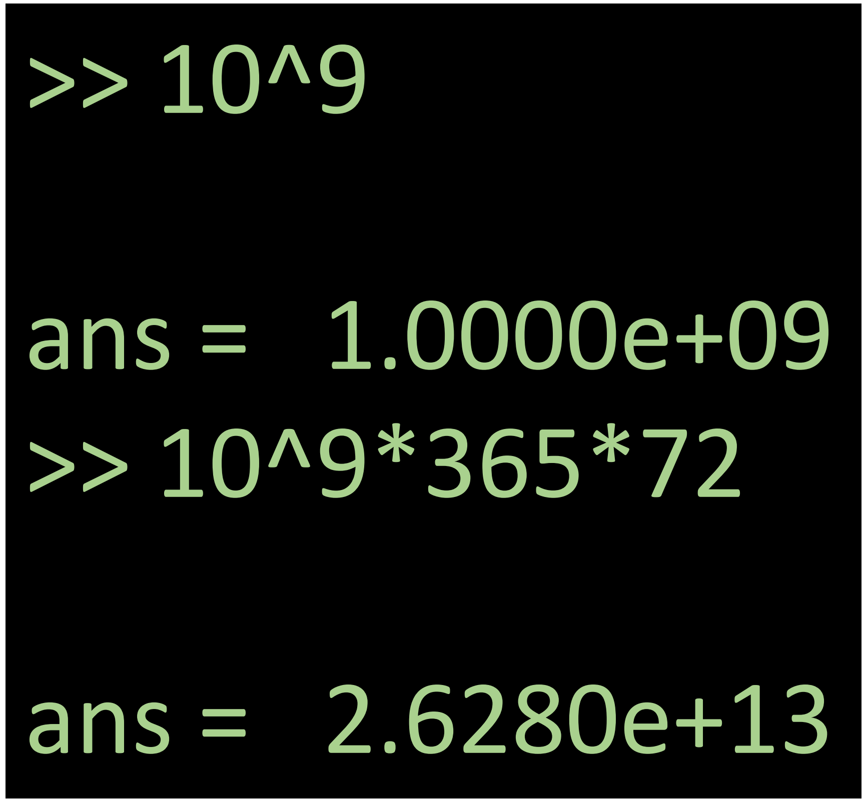
Atoms in the observable universe: 10^{79} to 10^{81}.



```
>> 10^9

ans =   1.0000e+09
>> 10^9*365*72

ans =   2.6280e+13
```

This of course ignores a number of factors such as the rate at which division fails to occur, lack of resources, etc

$$\approx 2^{1000000000000} \text{ evolutionary trials.}$$



```
>> 10^9

ans =   1.0000e+09
>> 10^9*365*72

ans =   2.6280e+13
```

This of course ignores a number of factors such as the rate at which division fails to occur, lack of resources, etc

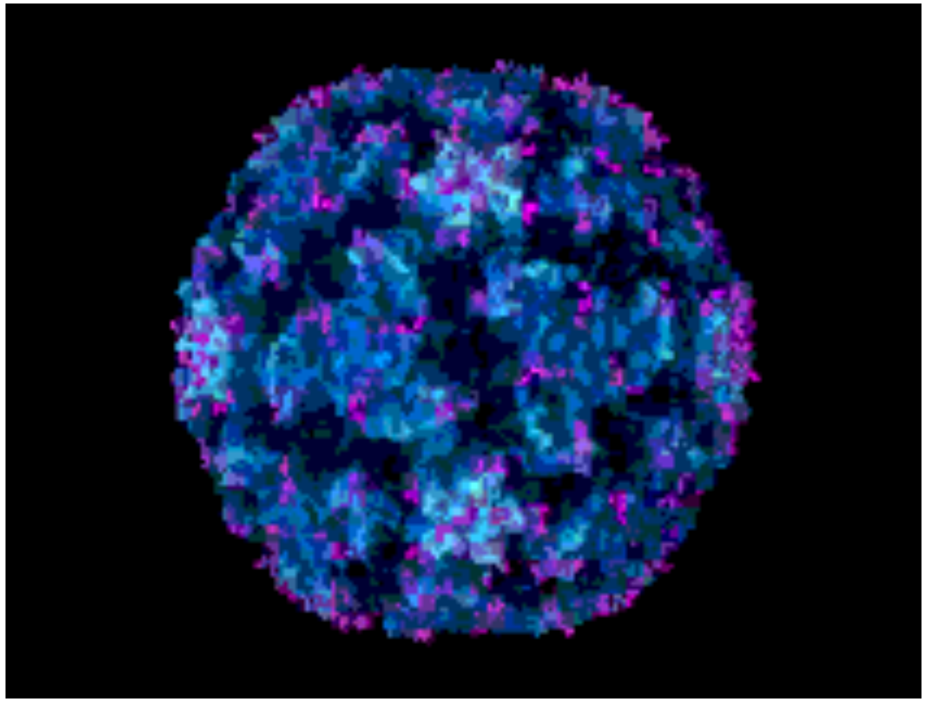
$$\approx 2^{10000000000000} \text{ evolutionary trials.}$$

This of course ignores a number
of factors such as the rate at which
division fails to occur, lack of resources, etc

... let's be more pessimistic.
Instead of doubling let's only allow the population to increase
by one in ten thousand per year.
Now the predicted number of evolutionary samples is 2.94×10^{151995}.

Crossover

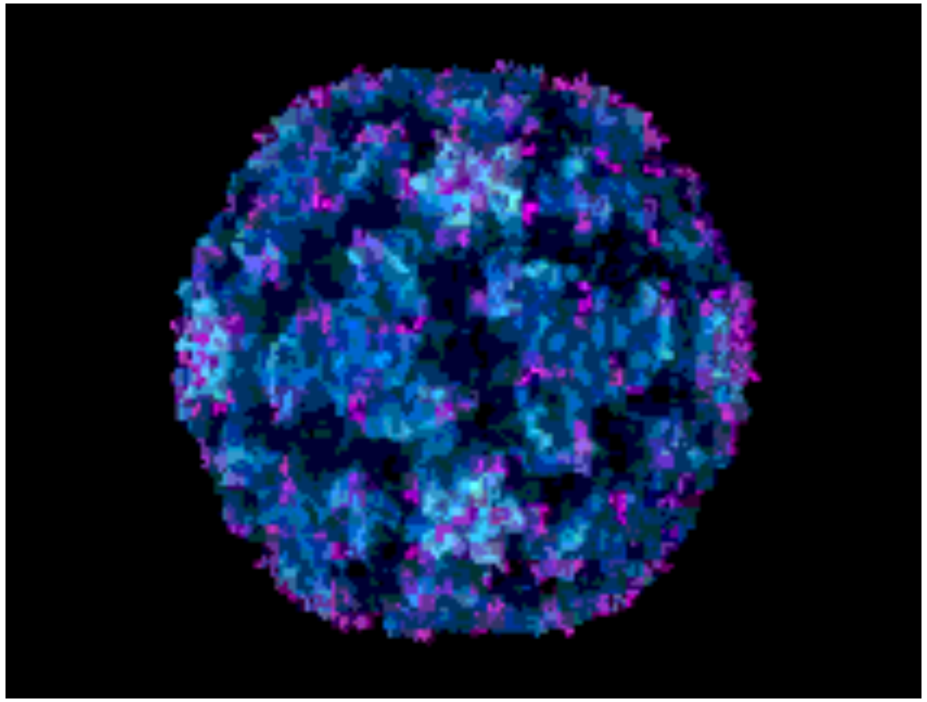
Horizontal gene flow in bacteria



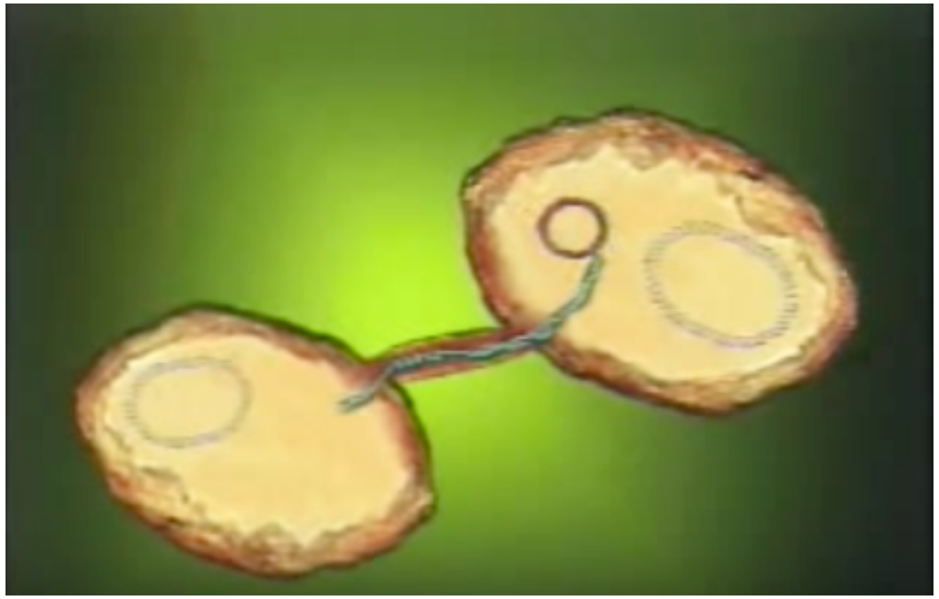
Transduction by viruses

Crossover

Horizontal gene flow in bacteria



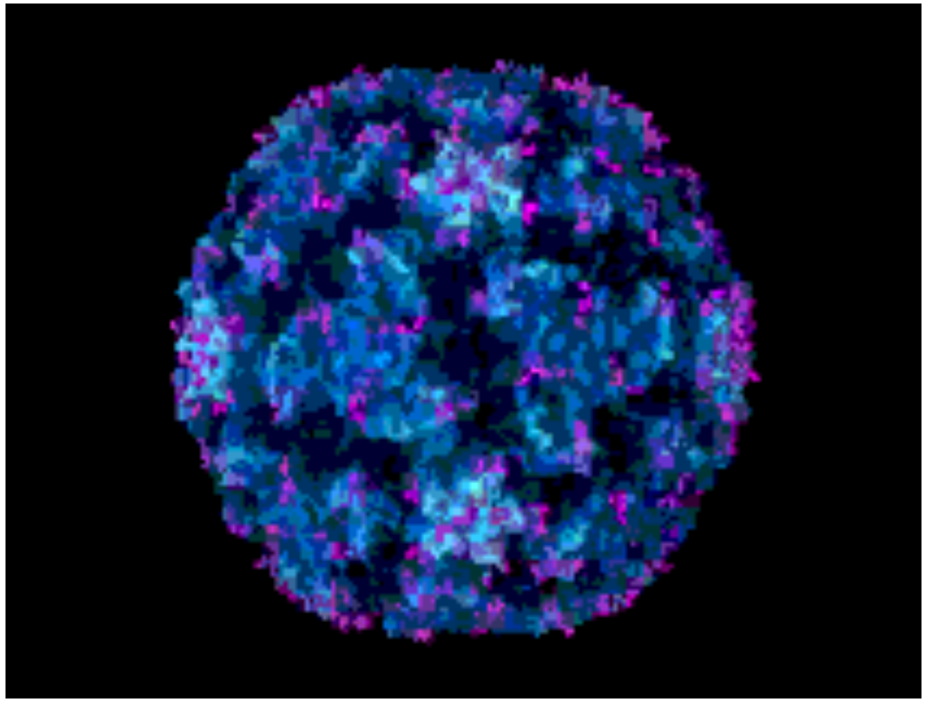
Transduction by viruses



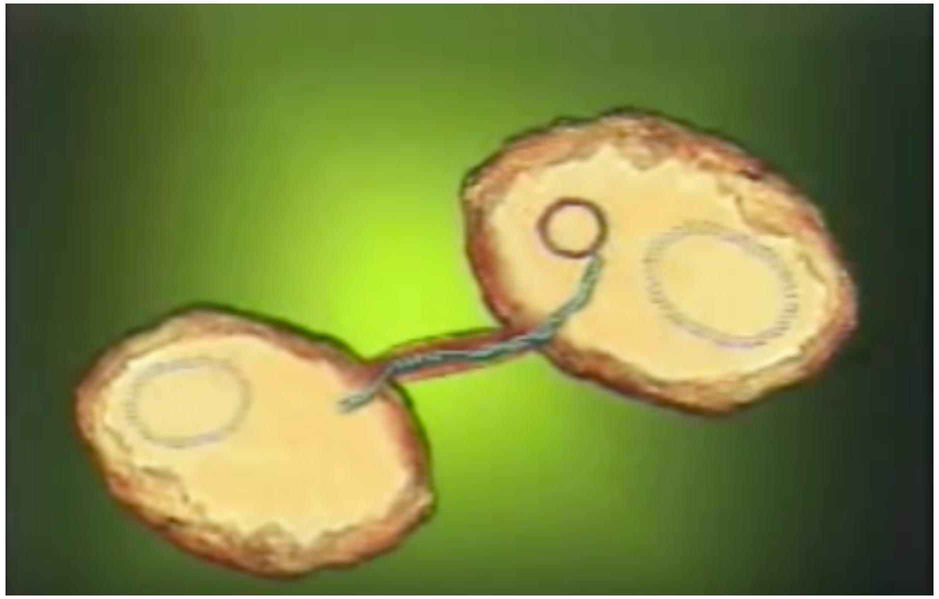
Conjugation

Crossover

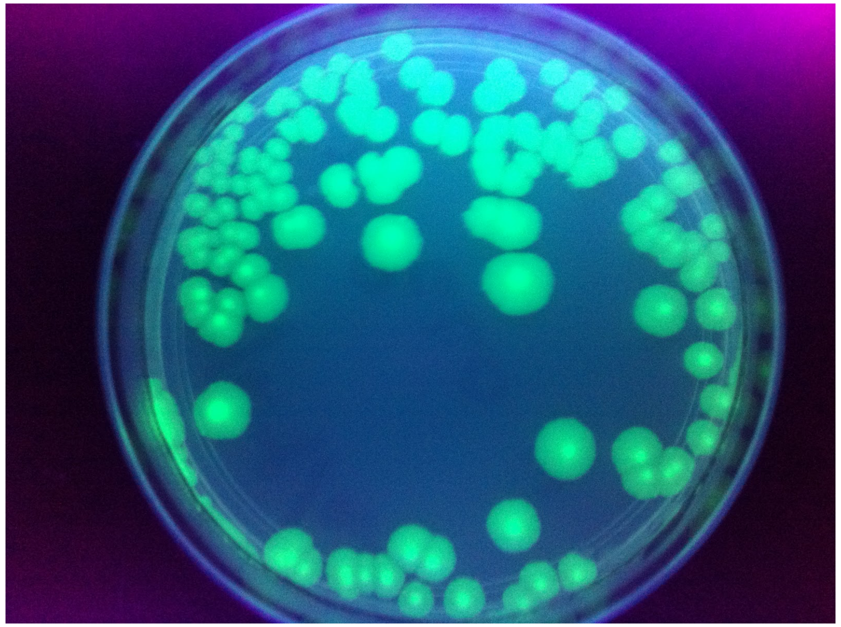
Horizontal gene flow in bacteria



Transduction by viruses



Conjugation



Transformation (DNA uptake)

Crossover (Parental)

In mammals:

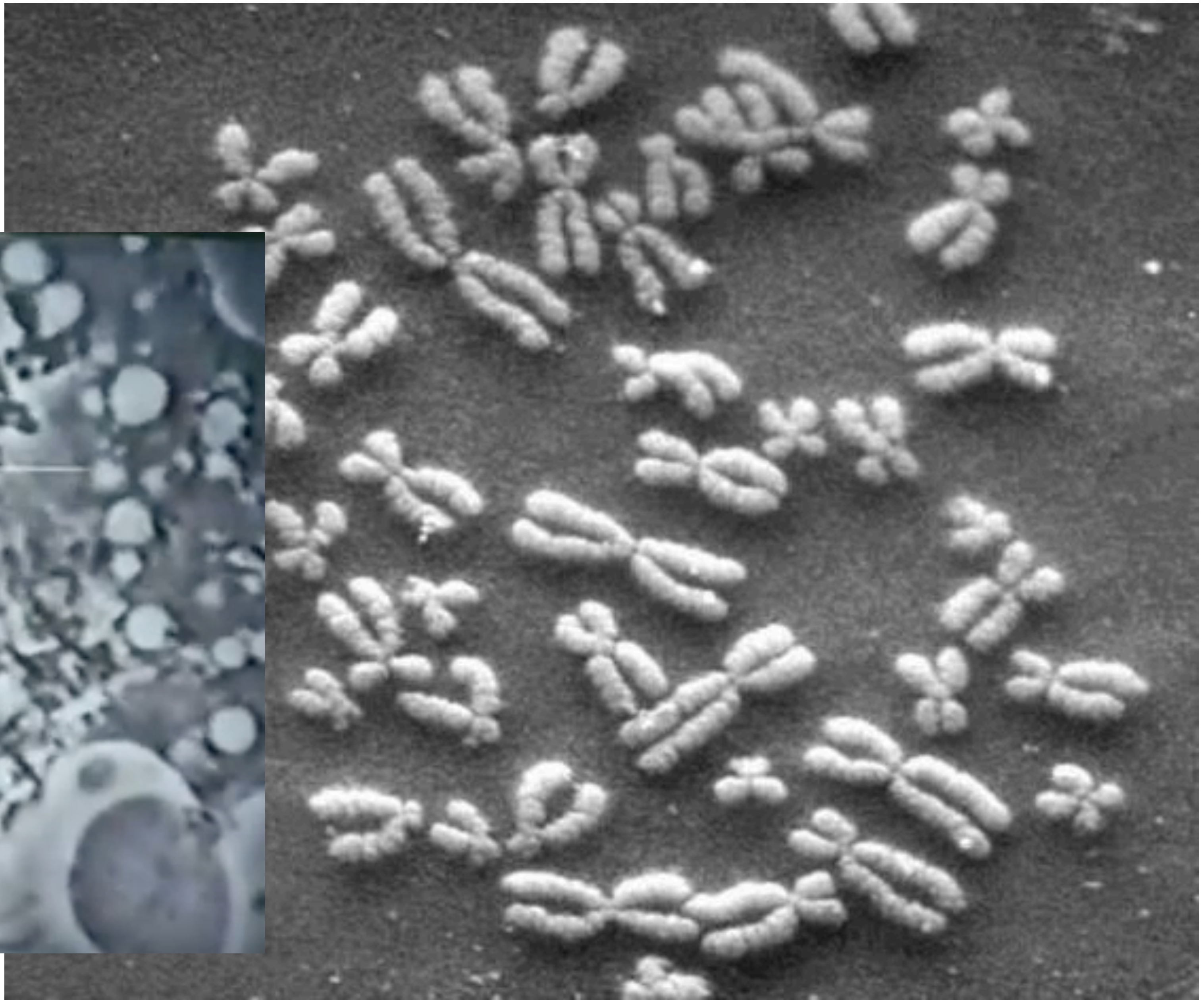
Adults are diploid (Paired chromosomes).
46 in humans

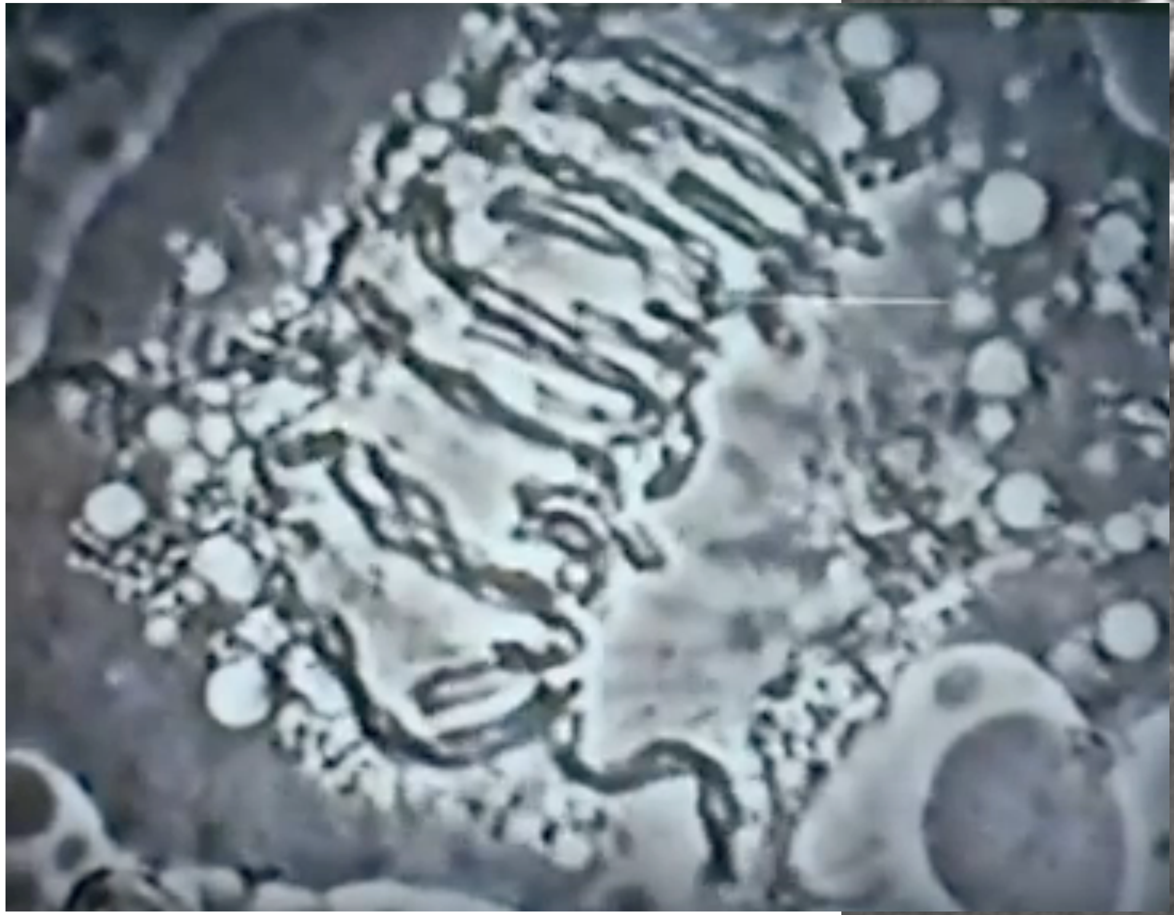
Gametes are haploid (Unpaired).
26 in humans

Meiosis – Generates Gametes

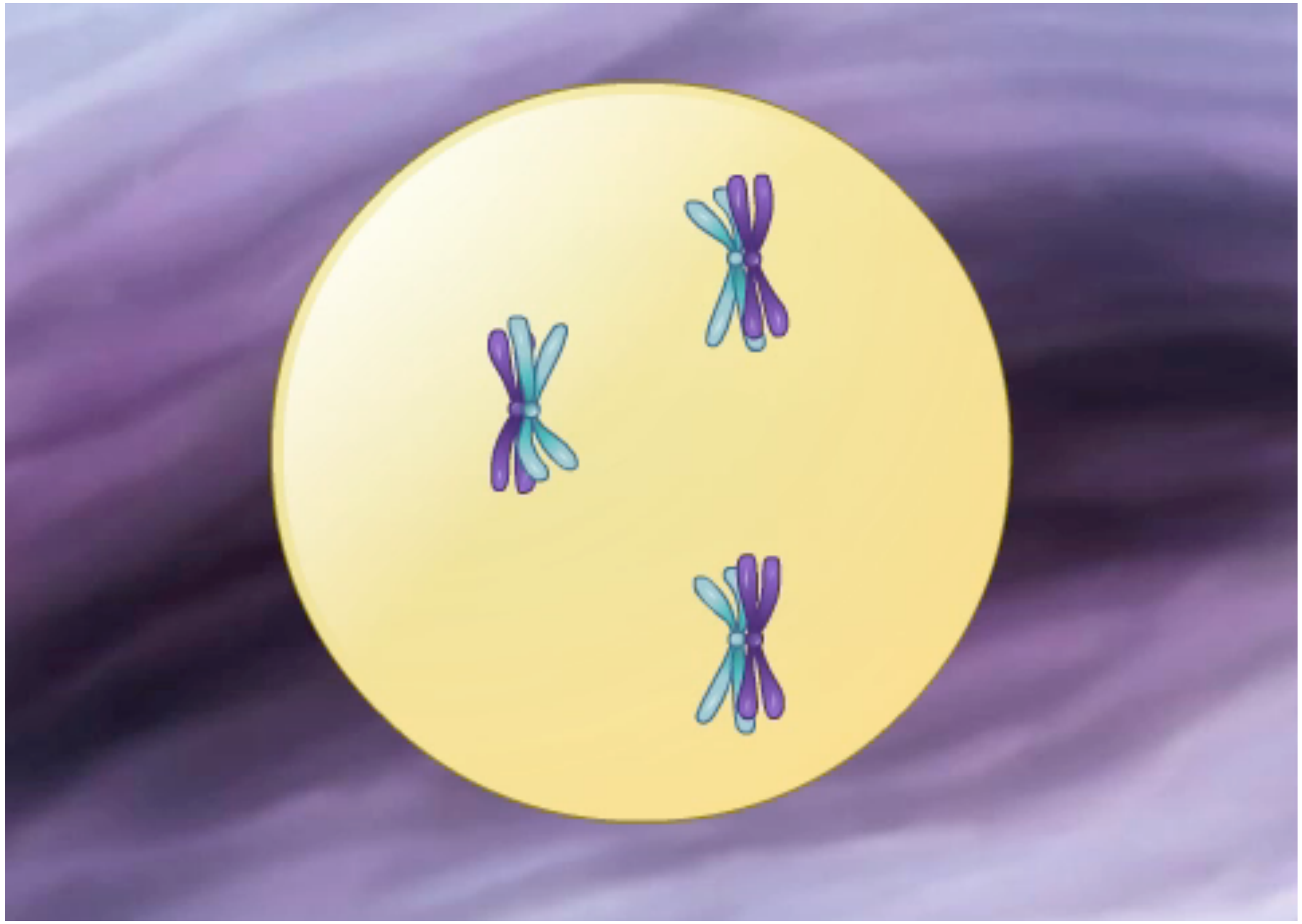
- Gametes are sperm in males and eggs in females
- The gametes are generated by recombination of the chromosomes received from the organisms parents.
- Once a gamete from each parent meets a new organism is formed (using the recombined DNA from their parents – really the 4 grandparents).

Crossover (Meiosis)





Meiosis (Parental)

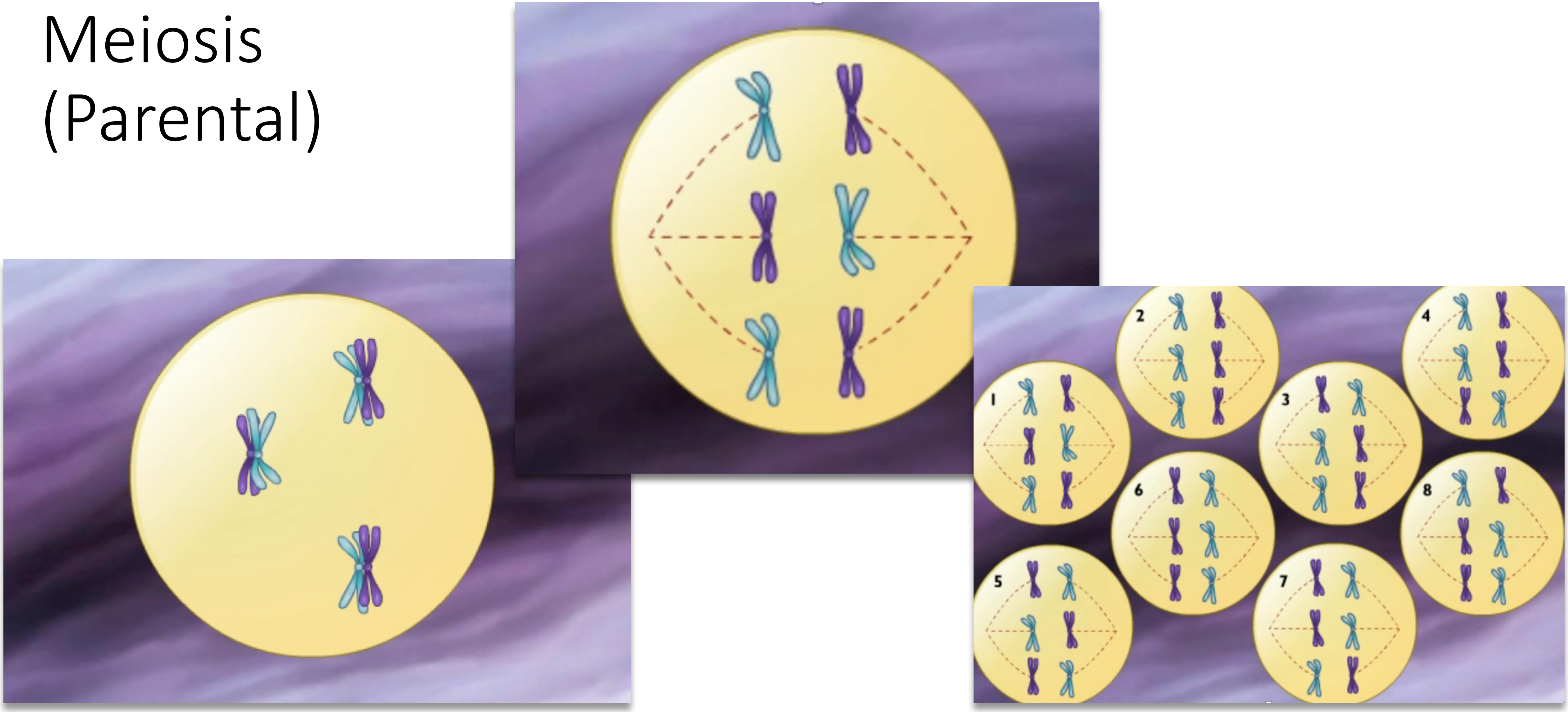


McGraw-Hill

2^n possible chromosome alignments, for n chromosomes.

In humans there are $2^{26} = 67,108,864$ possible combinations.

Meiosis (Parental)

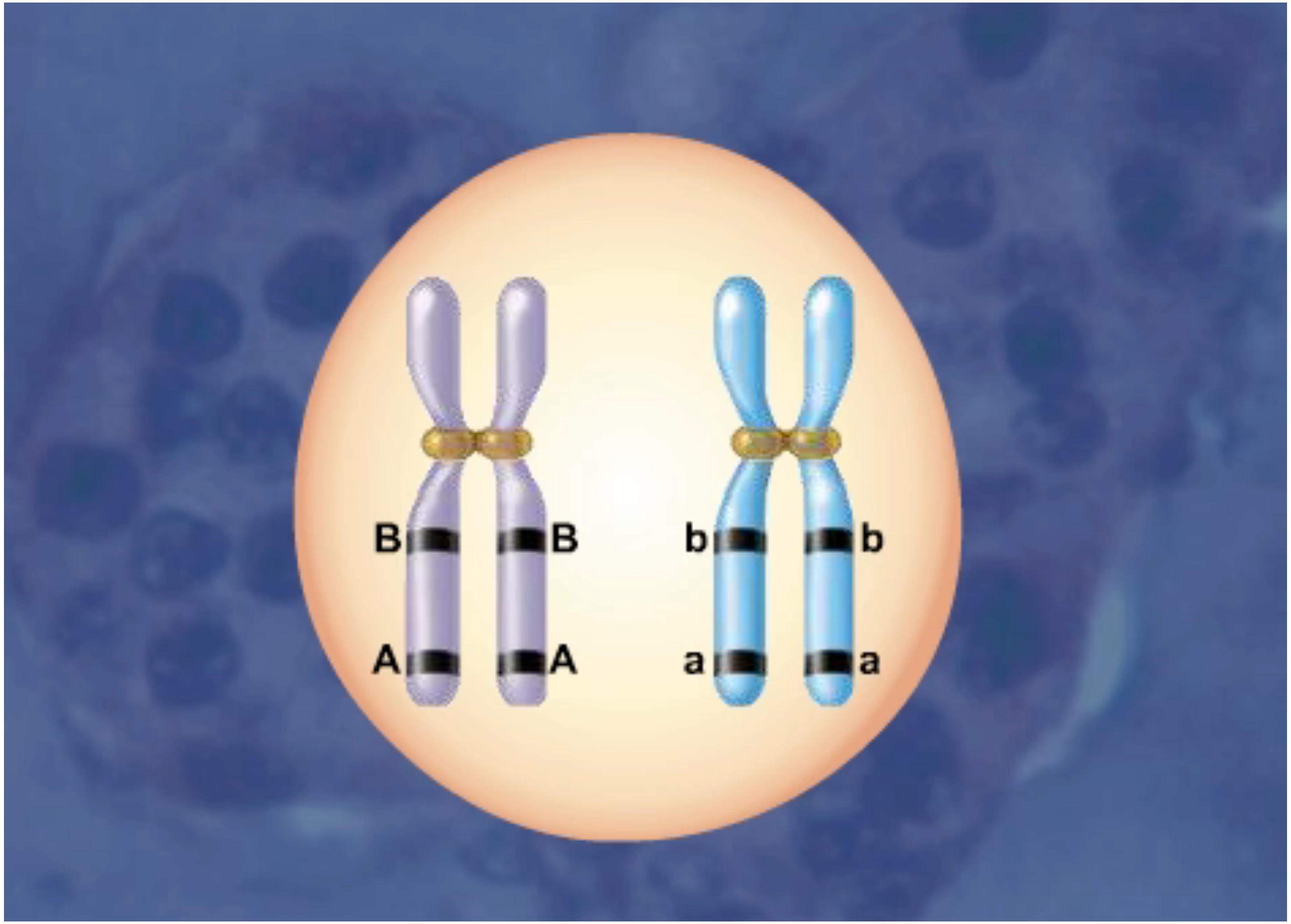


McGraw-Hill

2^n possible chromosome alignments, for n chromosomes.

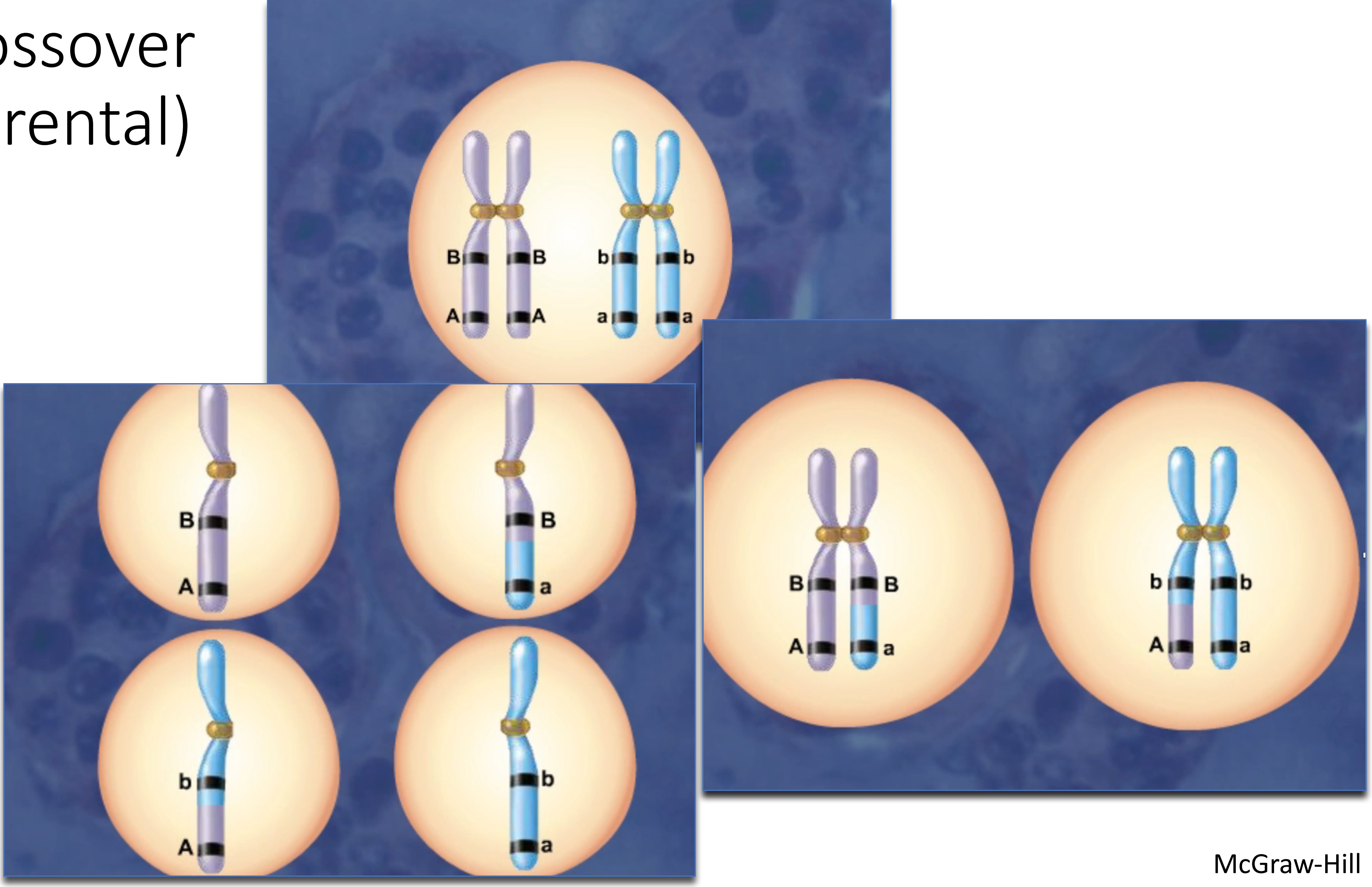
In humans there are $2^{26} = 67,108,864$ possible combinations.

Crossover (Parental)



McGraw-Hill

Crossover (Parental)



McGraw-Hill

Explore vs Exploit Trade-off

- Very common in optimisation problems

- K-armed bandit
- Resource collection (when should ants collect from a known seed pile vs look for new ones)
- When should you look for a new job vs stick with the one you have.
- When should you try a new restaurant vs going to a known one.
- GAs have to make this trade-off in exploring local maxima vs looking for potentially higher fitness scores.

Crossover

- GAs have to make this trade-off in exploring local maxima vs looking for potentially higher fitness scores.
- Increasing the mutation rate increases exploration.
- Increasing crossover rates increases exploitation of known information.
- A harsher selection operator also increases exploitation.

Crossover

- Adding a crossover operator drives the population to a uniform distribution of the current high fitness genomes. (Maximises the entropy of the population given the current collected data).
- Crossover drives the current population to thoroughly sample near known maxima at the expense of exploring new areas.
- Preserves useful information across the population so it is less likely to be lost through mutation (increases the entropy over the population of discovered genomes).

Crossover and Ergodicity

- Ideally a GA will be able to reach every point in the search space independent of the starting population given enough time.

- i.e. We want the GA to be ergodic (recall the definition from dynamical systems).

- Crossover alone is not ergodic (cannot explore the whole space).

Complete GA:

1. Reproduction (selection)
2. Crossover
3. Mutation

Implementing “Digital Sex”

1. Select two genomes, A, B in the population
2. Choose a position in the genome, p
3. Swap the genomes in each individual at point, p

$A = A_1, A_2, A_3, A_4, A_5$

$B = B_1, B_2, B_3, B_4, B_5$

after crossover with $p = 2$

$A = A_1, A_2, B_3, B_4, B_5$

$B = B_1, B_2, A_3, A_4, A_5$

Implementing “Digital Sex”

1. Select two genomes, A, B in the population
2. Choose a position in the genome, p
3. Swap the genomes in each individual at point, p

$A = A_1, A_2, A_3, A_4, A_5$

$B = B_1, B_2, B_3, B_4, B_5$

after crossover with $p = 2$

$A = A_1, A_2, B_3, B_4, B_5$

$B = B_1, B_2, A_3, A_4, A_5$

This is 1-point crossover
Imagine what n-point crossover would look like.

Locality is important… exploits current knowledge.

Implementing "Digital Sex"

1. Select two genomes, A, B in the population
2. Choose a probability, p
3. Swap the genomes in each individual at each point with probability, p

$A = A_1, A_2, A_3, A_4, A_5$

$B = B_1, B_2, B_3, B_4, B_5$

possible crossover with $p = 0.5$

$A = B_1, A_2, A_3, B_4, A_5$

$B = A_1, B_2, B_3, A_4, B_5$

This is uniform crossover.
Locality does not matter.
Less information preserving
so more exploratory.