



UiT The Arctic University of Norway

# Genetic algorithms

*An optimization algorithm modelled after Darwin's evolutionary theory*

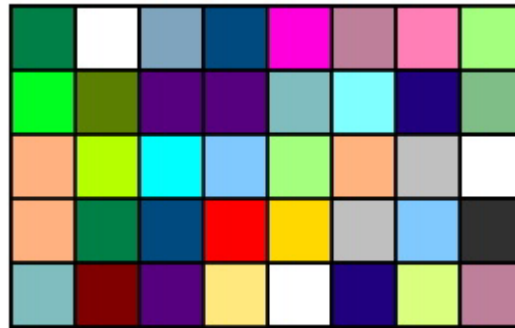
*Lecture 3/3 – A practical example*

Andreas Dyrøy Jansson

# Practical example: RGB colors

- Problem: find the target color
- Three features/genes: R, G and B
  - Values may range from 0 – 255

- Initial population:



- Target color:



# Representation

- Consider the following color (123, 63, 139):
- High level representation:
  - R=123
  - G=63
  - B=139
- Low level representation
  - Binary string (01111011 00111111 10001011)



# How to determine fitness

- Again, this depends on the task we are trying to solve
  - Target color
  - Compare each feature with the target
  - Fitness  $f$  is based on the difference from the target
    - Smaller difference means a fitter individual
    - Values are assigned as follows: 0 – no similarity, 100 – perfect match
- Other problems may optimize the speed, cost or similar of some system

# Initial population fitness

- Fitness is assigned as values from 0-100 for every individual of the initial population:
- 0 = low fitness
- 100 = high fitness

0	30	20	25	70	60	70	15
25	0	80	80	20	40	45	15
50	25	40	35	15	50	20	30
50	0	25	65	10	20	40	10
35	50	80	10	30	45	25	70

# Selection

- Based on fitness
- In our example, how different is each color from the target
- Elitist selection
  - We select from the top 50% of the population:
  - Only the fittest individuals are used to create the next generation
  - Can potentially find the solution faster, as the worst individuals are removed from the gene pool



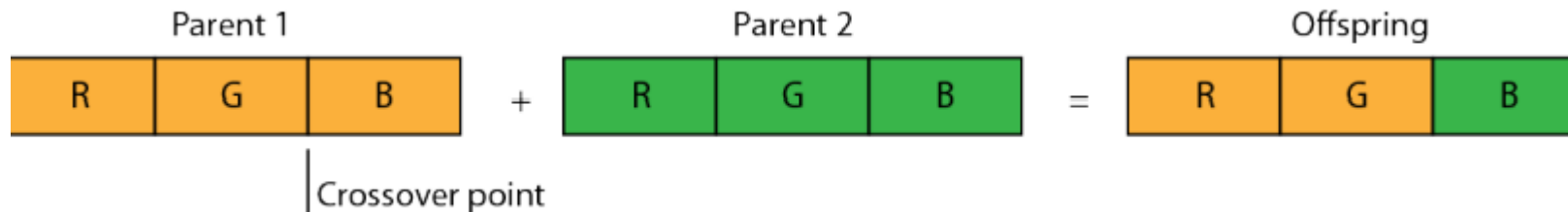
# Selection – roulette wheel

- As before, all individuals are assigned a fitness score from 0-100. Only this time, every individual may be selected for crossover.
- Fit individuals have a higher chance of being selected
- Less fit individuals may still be selected, and their features carried over to the next generation, albeit with a lower chance
  - This may actually be beneficial in the long run to ensure genetic diversity



# Crossover – single point

- How features from each “parent” combine
- Two parents are selected using either elitist or roulette wheel selection
- Consider an RGB color as example:

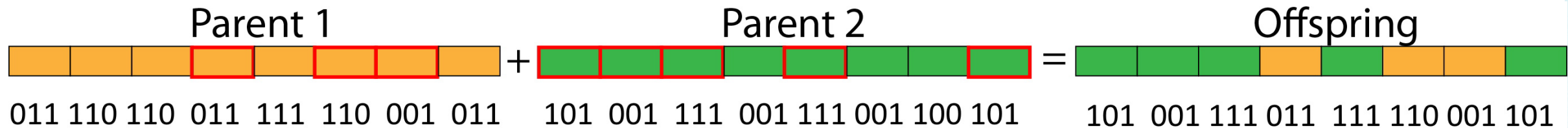


- A crossover point is selected, and determines what genes are copied over from each parent



# Crossover – multi-point




- As before, two parents are selected and their features combined. However, this time, multiple genes are copied randomly from each parent:



- In this example, RGB values are encoded as binary strings and split into 8 chunks.
- Note: Binary representation is not necessary for using multi-point crossover, but makes it easier to show the method in this context.

# Crossover - arithmetic

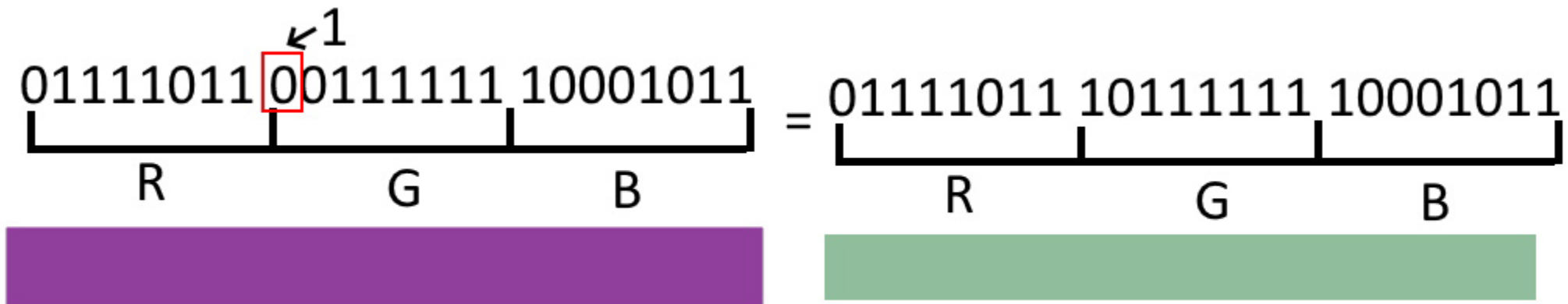
- Instead of just copying features directly from each parent, some arithmetic function may be used. Consider again the RGB color example:

Parent 1				Parent 2				Offspring		
43	148	255	x	252	35	62	=	$\frac{43+252}{2}$	$\frac{148+35}{2}$	$\frac{255+62}{2}$
										

- In this case, we take the average of each feature value to produce the offspring. Using the average is just an example, there is no limit on what function may be used
  - On binary strings, one may for example use bitwise operations

# Mutations

- Mutations are a way to ensure genetic diversity
  - If the population becomes too uniform, it may have a hard time adapting to unexpected changes in its fitness function
- Additionally, the algorithm may converge into a sub-optimal solution before it has tried every combination possible
- Mutations may be implemented in different ways, the most popular being “bit-flip” for binary strings:



# Demonstration

