Optimization Techniques

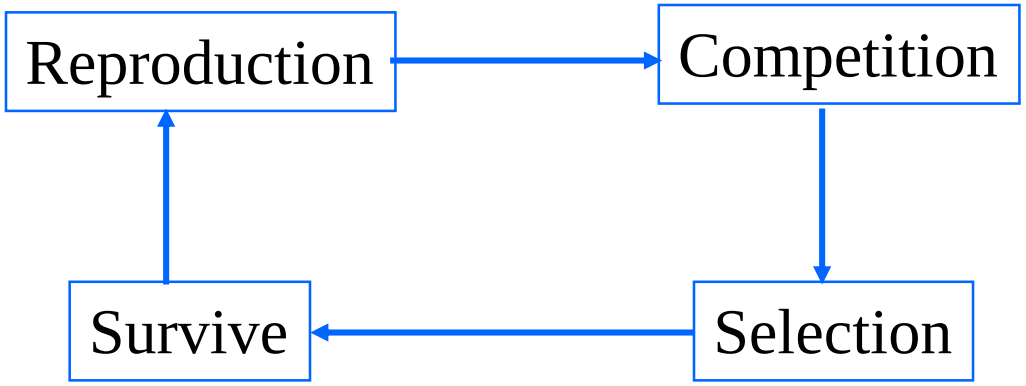
Genetic Algorithms

Optimization Techniques

- Mathematical Programming
- Network Analysis
- Branch & Bound
- Genetic Algorithm
- Simulated Annealing
- Tabu Search

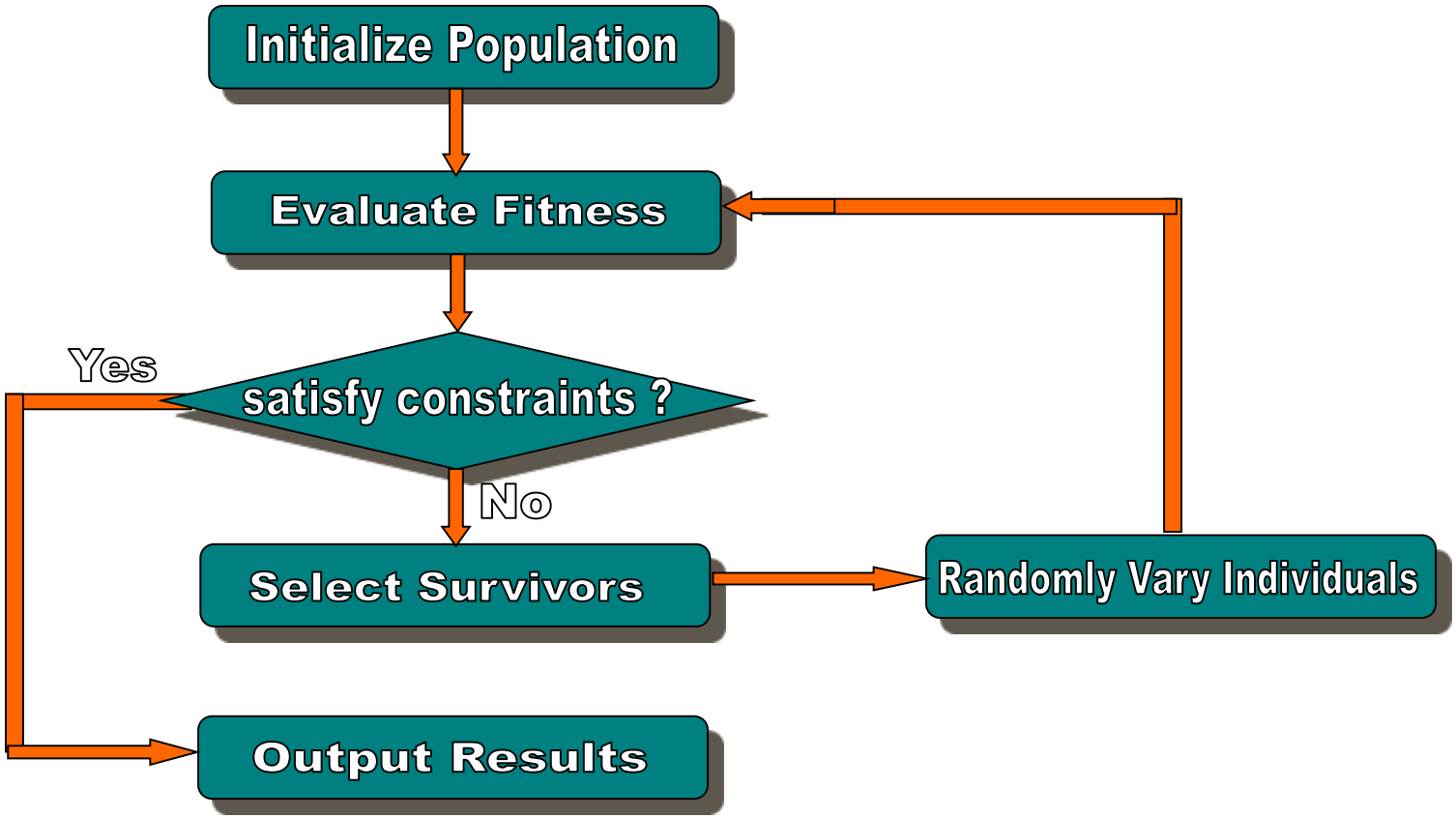
Genetic Algorithm

- Based on Darwinian Paradigm



- Intrinsically a robust search and optimization mechanism

Conceptual Algorithm



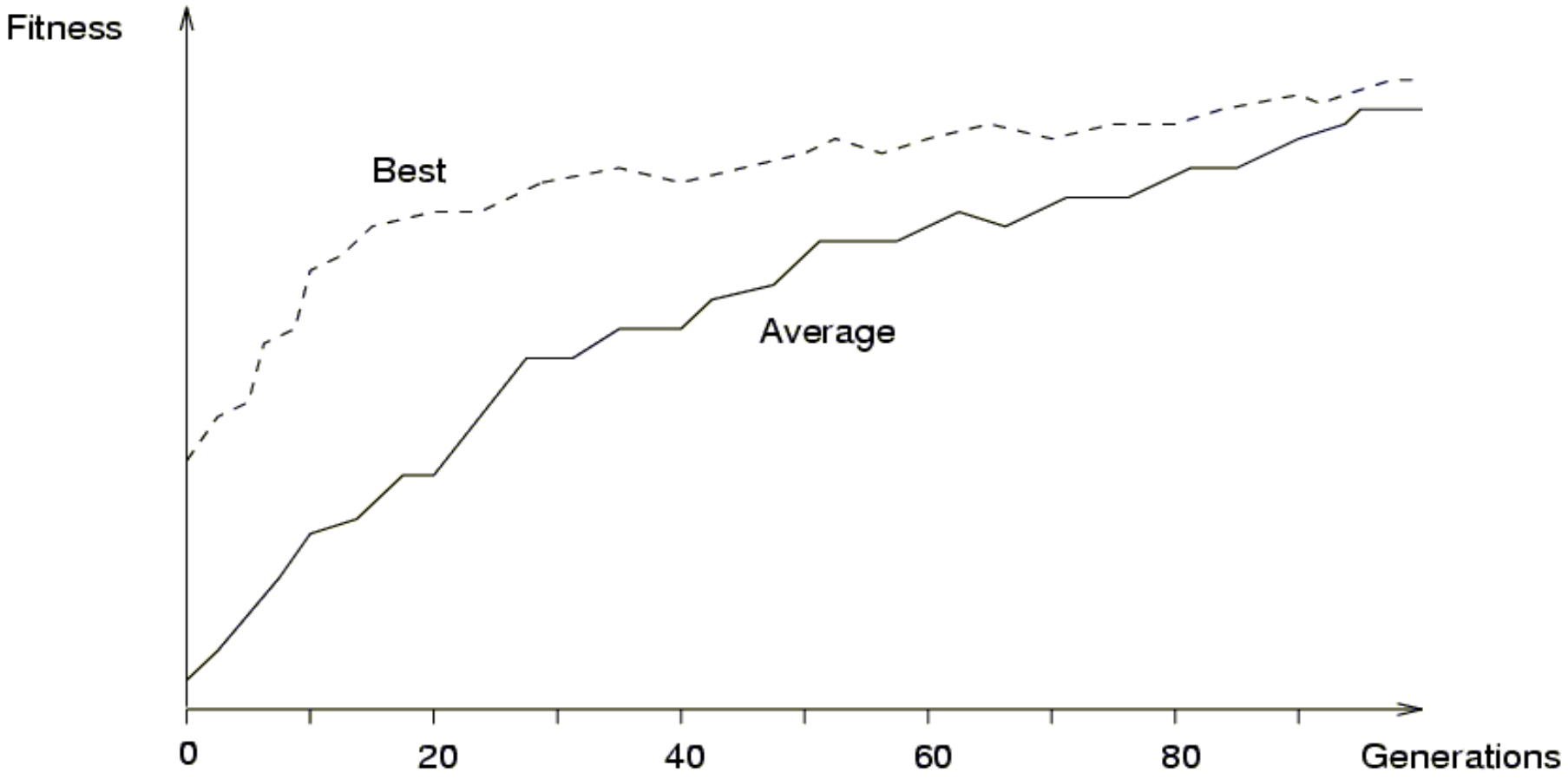
Genetic Algorithm Introduction 1

- Inspired by natural evolution
- Population of individuals
  - Individual is feasible solution to problem
- Each individual is characterized by a Fitness function
  - Higher fitness is better solution
- Based on their fitness, parents are selected to reproduce offspring for a new generation
  - Fitter individuals have more chance to reproduce
  - New generation has same size as old generation; old generation dies
- Offspring has combination of properties of two parents
- If well designed, population will converge to optimal solution

Algorithm

```
BEGIN
  Generate initial population;
  Compute fitness of each individual;
  REPEAT /* New generation /*
    FOR population_size / 2 DO
      Select two parents from old generation;
       /* biased to the fitter ones */
      Recombine parents for two offspring;
      Compute fitness of offspring;
      Insert offspring in new generation
    END FOR
  UNTIL population has converged
END
```

Example of convergence



Introduction 2

- Reproduction mechanisms have no knowledge of the problem to be solved

- Link between genetic algorithm and problem:
  - Coding
  - Fitness function

Basic principles 1

- Coding or Representation
  - String with all parameters
- Fitness function
  - Parent selection
- Reproduction
  - Crossover
  - Mutation
- Convergence
  - When to stop

Basic principles 2

- An individual is characterized by a set of parameters: Genes
- The genes are joined into a string: Chromosome

- The chromosome forms the genotype
- The genotype contains all information to construct an organism: the phenotype

---

- Reproduction is a "dumb" process on the chromosome of the genotype
- Fitness is measured in the real world ('struggle for life') of the phenotype

Coding

- Parameters of the solution (genes) are concatenated to form a string (chromosome)
- All kind of alphabets can be used for a chromosome (numbers, characters), but generally a binary alphabet is used
- Order of genes on chromosome can be important
- Generally many different codings for the parameters of a solution are possible
- Good coding is probably the most important factor for the performance of a GA
- In many cases many possible chromosomes do not code for feasible solutions

Genetic Algorithm

- Encoding
- Fitness Evaluation
- Reproduction
- Survivor Selection

Encoding

- Design alternative → individual (chromosome)
- Single design choice → gene
- Design objectives → fitness

Example

- Problem
  - Schedule *n* jobs on *m* processors such that the maximum span is minimized.

Design alternative: job i (i=1,2,…n) is assigned to processor j (j=1,2,…,m)

Individual: A n-vector **x** such that $x_i = 1, \ldots, or\ m$

Design objective: minimize the maximal span

Fitness: the maximal span for each processor

Reproduction

- Reproduction operators
  - Crossover
  - Mutation

Reproduction

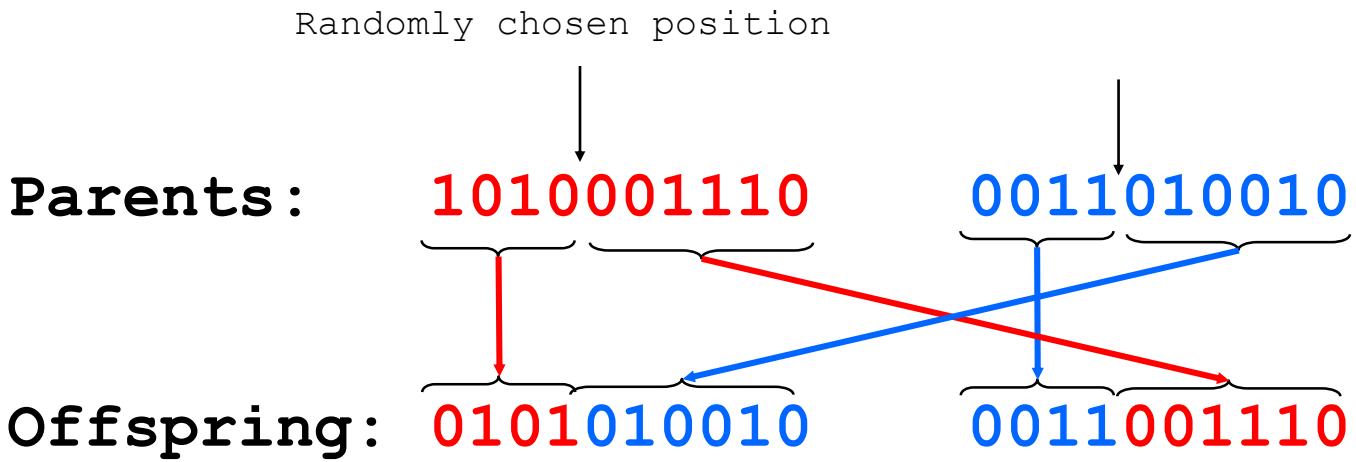
- Crossover
  - Two parents produce two offspring
  - There is a chance that the chromosomes of the two parents are copied unmodified as offspring
  - There is a chance that the chromosomes of the two parents are randomly recombined (crossover) to form offspring
  - Generally the chance of crossover is between 0.6 and 1.0
- Mutation
  - There is a chance that a gene of a child is changed randomly
  - Generally the chance of mutation is low (e.g. 0.001)

Reproduction Operators

- Crossover
  - Generating offspring from two selected parents
    - ☒Single point crossover
    - ☒Two point crossover (Multi point crossover)
    - ☒Uniform crossover

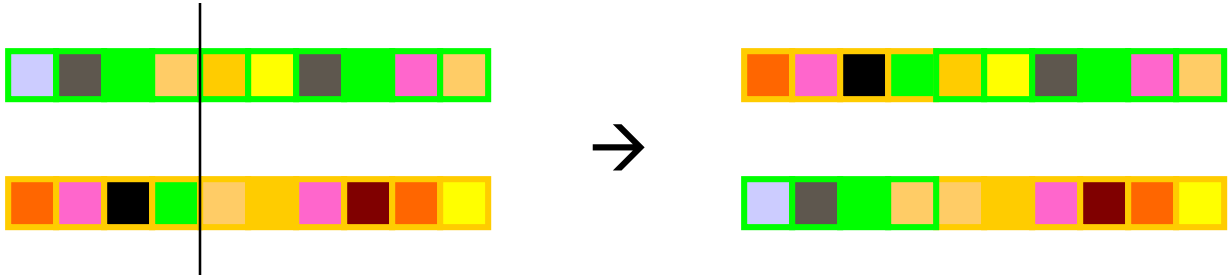
One-point crossover 1

- Randomly one position in the chromosomes is chosen
- Child 1 is head of chromosome of parent 1 with tail of chromosome of parent 2
- Child 2 is head of 2 with tail of 1



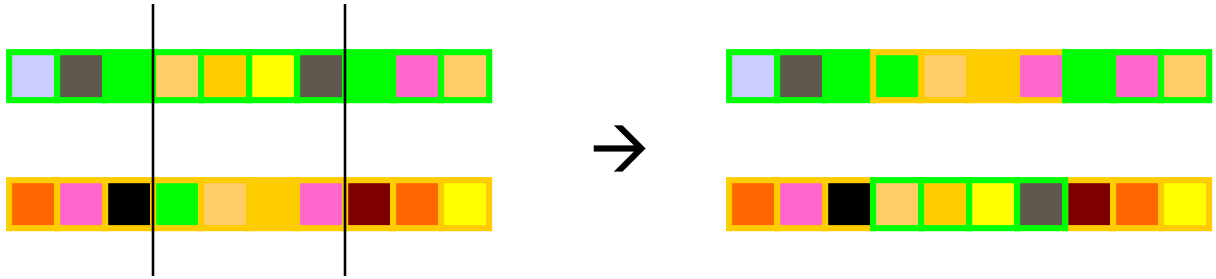
Reproduction Operators comparison

- Single point crossover

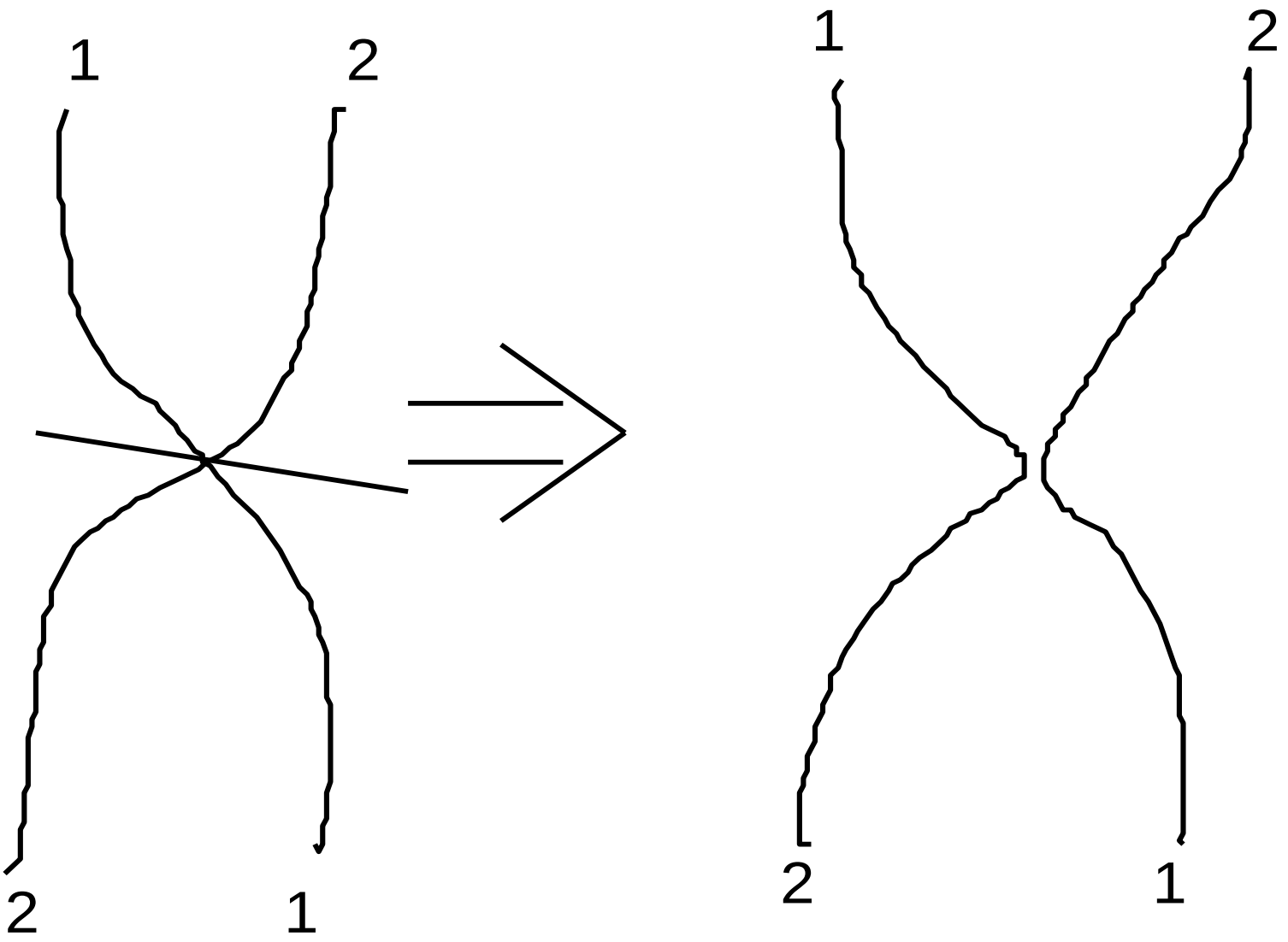


Cross point

- Two point crossover (Multi point crossover)

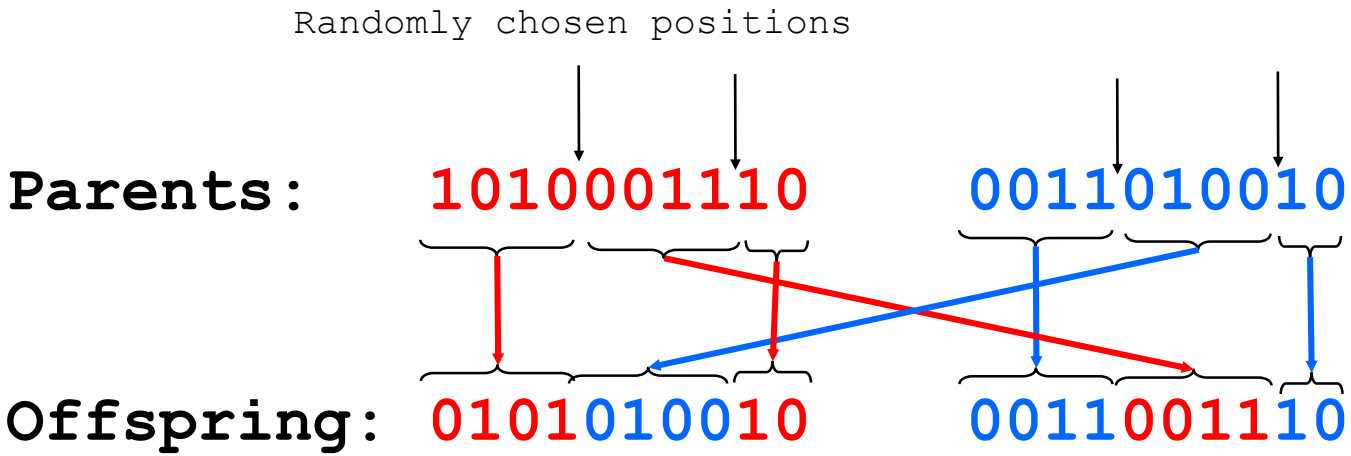


One-point crossover - Nature



Two-point crossover

- Randomly two positions in the chromosomes are chosen
- Avoids that genes at the head and genes at the tail of a chromosome are always split when recombined



Uniform crossover

- A random mask is generated
- The mask determines which bits are copied from one parent and which from the other parent
- Bit density in mask determines how much material is taken from the other parent (takeover parameter)

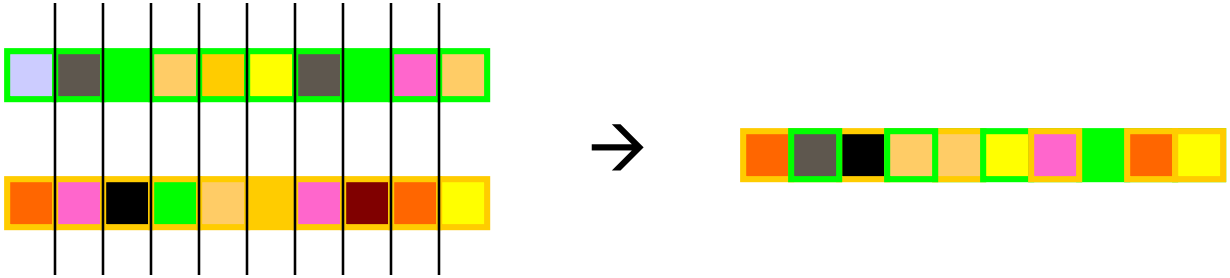
```
Mask:        0110011000      (Randomly generated)

Parents:     1010001110      0011010010

Offspring:   0011001010      1010010110
```

Reproduction Operators

- Uniform crossover



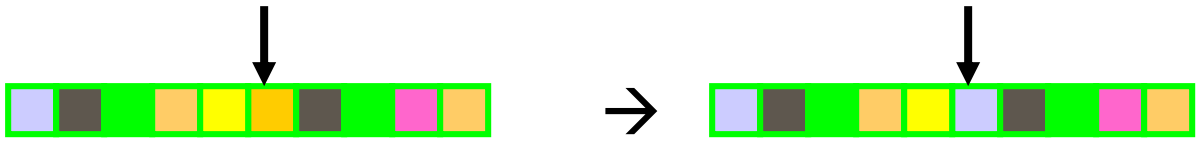
  - Is uniform crossover better than single crossover point?
    - Trade off between
      - Exploration: introduction of new combination of features
      - Exploitation: keep the good features in the existing solution

Problems with crossover

- Depending on coding, simple crossovers can have high chance to produce illegal offspring
  - E.g. in TSP with simple binary or path coding, most offspring will be illegal because not all cities will be in the offspring and some cities will be there more than once
- Uniform crossover can often be modified to avoid this problem
  - E.g. in TSP with simple path coding:
    - ☒Where mask is 1, copy cities from one parent
    - ☒Where mask is 0, choose the remaining cities in the order of the other parent

Reproduction Operators

- Mutation
  - Generating new offspring from single parent



  - Maintaining the diversity of the individuals
    - ☒Crossover can only explore the combinations of the current gene pool
    - ☒Mutation can "generate" new genes

Reproduction Operators

- Control parameters: **population size, crossover/mutation probability**
  - Problem specific
  - Increase population size
    - ☒Increase diversity and computation time for each generation
  - Increase crossover probability
    - ☒Increase the opportunity for recombination but also disruption of good combination
  - Increase mutation probability
    - ☒Closer to randomly search
    - ☒Help to introduce new gene or reintroduce the lost gene
- Varies the population
  - Usually using crossover operators to recombine the genes to generate the new population, then using mutation operators on the new population

Parent/Survivor Selection

- Strategies
  - Survivor selection
    - ☒Always keep the best one
    - ☒Elitist: deletion of the K worst
    - ☒Probability selection : inverse to their fitness
    - ☒Etc.

Parent/Survivor Selection

- Too strong fitness selection bias can lead to sub-optimal solution
- Too little fitness bias selection results in unfocused and meandering search

Parent selection

Chance to be selected as parent proportional to fitness

- Roulette wheel

To avoid problems with fitness function

- Tournament

Not a very important parameter

Parent/Survivor Selection

- Strategies
  - Parent selection
    - ☒Uniform randomly selection
    - ☒Probability selection : proportional to their fitness
    - ☒Tournament selection (Multiple Objectives)
      - Build a small comparison set
      - Randomly select a pair with the higher rank one beats the lower one
        - Non-dominated one beat the dominated one
        - **Niche count**: the number of points in the population within certain distance, higher the niche count, lower the rank.
    - ☒Etc.

Tournament

- Binary tournament
  - Two individuals are randomly chosen; the fitter of the two is selected as a parent
- Probabilistic binary tournament
  - Two individuals are randomly chosen; with a chance p, $0.5<p<1$, the fitter of the two is selected as a parent
- Larger tournaments
  - n individuals are randomly chosen; the fittest one is selected as a parent

- By changing n and/or p, the GA can be adjusted dynamically

Problems with fitness range

- Premature convergence
  - ∆Fitness too large
  - Relatively superfit individuals dominate population
  - Population converges to a local maximum
  - Too much exploitation; too few exploration
- Slow finishing
  - ∆Fitness too small
  - No selection pressure
  - After many generations, average fitness has converged, but no global maximum is found; not sufficient difference between best and average fitness
  - Too few exploitation; too much exploration

Solutions for these problems

- Use tournament selection
  - Implicit fitness remapping
- Adjust fitness function for roulette wheel
  - Explicit fitness remapping
    - ☒Fitness scaling
    - ☒Fitness windowing
    - ☒Fitness ranking

Fitness Function

Purpose

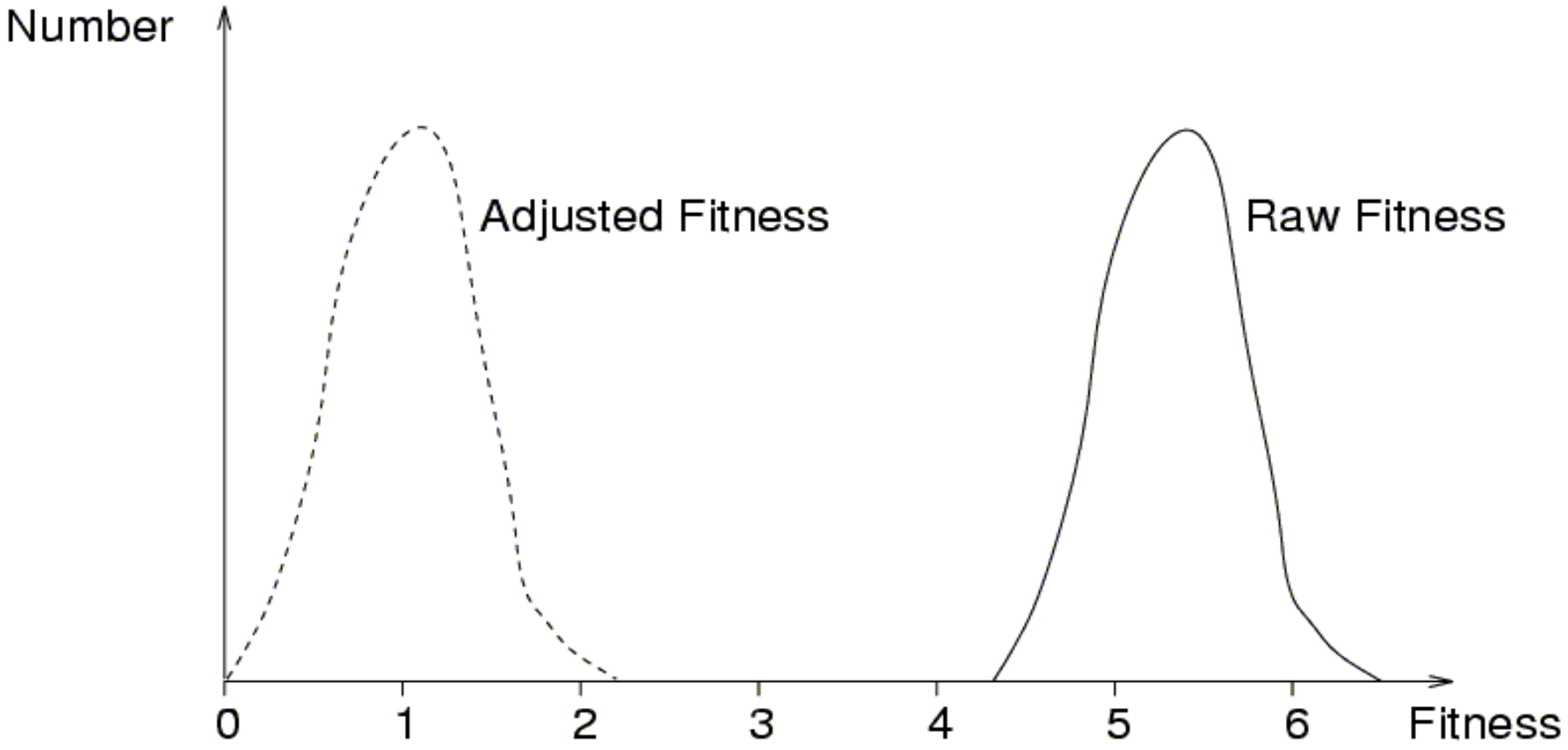
- Parent selection
- Measure for convergence
- For Steady state: Selection of individuals to die

- Should reflect the value of the chromosome in some "real" way
- Next to coding the most critical part of a GA

Fitness scaling

- Fitness values are scaled by subtraction and division so that worst value is close to 0 and the best value is close to a certain value, typically 2
  - Chance for the most fit individual is 2 times the average
  - Chance for the least fit individual is close to 0
- Problems when the original maximum is very extreme (super-fit) or when the original minimum is very extreme (super-unfit)
  - Can be solved by defining a minimum and/or a maximum value for the fitness

Example of Fitness Scaling



Fitness windowing

- Same as window scaling, except the amount subtracted is the minimum observed in the *n* previous generations, with *n* e.g. 10
- Same problems as with scaling

Fitness ranking

- Individuals are numbered in order of increasing fitness
- The rank in this order is the adjusted fitness
- Starting number and increment can be chosen in several ways and influence the results

- No problems with super-fit or super-unfit
- Often superior to scaling and windowing

Fitness Evaluation

- A key component in GA
- Time/quality trade off
- Multi-criterion fitness

Multi-Criterion Fitness

- Weighted sum
  - $F(\boldsymbol{x}) = w_1 f_1(x_1) + w_2 f_2(x_2) + \ldots + w_n f_n(x_n)$
  - *Problems?*
    - ☒ *Convex and convex Pareto optimal front*

      Sensitive to the shape of the Pareto-optimal front
    - ☒ *Selection of weights?*

      Need some pre-knowledge

      Not reliable for problem involving uncertainties

Other parameters of GA 1

- Initialization:
  - Population size
  - Random
  - Dedicated greedy algorithm
- Reproduction:
  - Generational: as described before (insects)
  - Generational with elitism: fixed number of most fit individuals are copied unmodified into new generation
  - Steady state: two parents are selected to reproduce and two parents are selected to die; two offspring are immediately inserted in the pool (mammals)

Other parameters of GA 2

- Stop criterion:
  - Number of new chromosomes
  - Number of new and unique chromosomes
  - Number of generations
- Measure:
  - Best of population
  - Average of population
- Duplicates
  - Accept all duplicates
  - Avoid too many duplicates, because that degenerates the population (inteelt)
  - No duplicates at all

Example run

Maxima and Averages of steady state and generational replacement