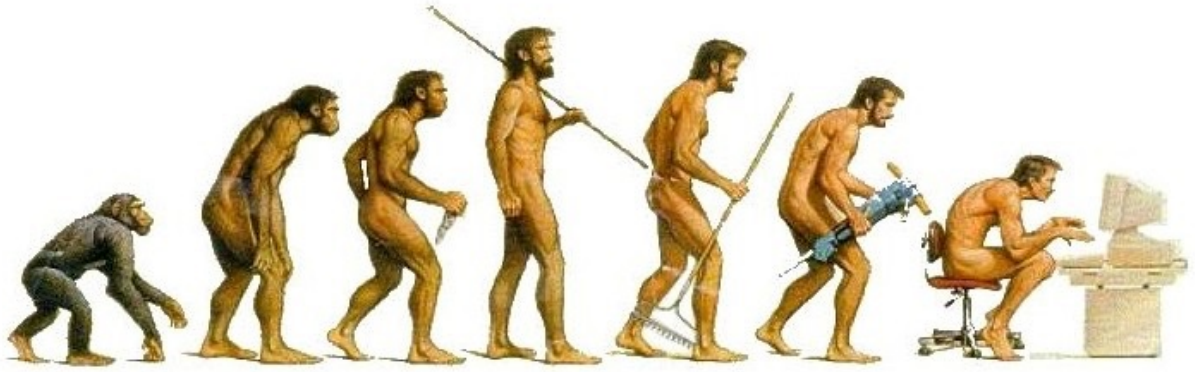
Survival of the Fittest

An Introduction to Genetic Algorithms and Evolutionary Computation



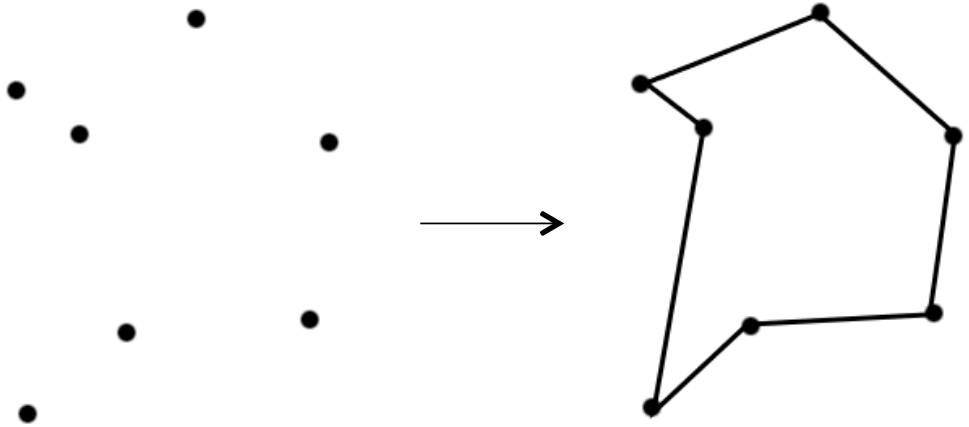
Lecture outline

- An NP-Complete Problem – The TSP
- Darwin's Theory of Evolution
- Genetic Algorithms (GA)
- Applications of GA
- Genetic Operators
- Generic GA
- Why does GAs work?

Traveling Salesperson

- Given a list of cities to visit.
- Goal: find the shortest tour that visits each city exactly once, returning in the end to the starting point.
- Complexity: O(n!)

NP-Hard



Darwin's Theory of Evolution

- All life is related and has descended from a common ancestor.
- Natural selection
  - **“Survival of the fittest”**
- Organisms can produce more offspring than their surroundings can support -> natural struggle to survive.
- Organisms whose variations best adapt them to their environments survive, the others die.
- Variations are heritable -> can be passed on to the next generation -> *i.e.,* ***evolution***

Inheritance

Mutation

Selection

Crossover

Genetic Algorithms

What is Genetic Algorithms?

- **John Holland (70's)**
- *Nature's mechanism for evolution could be modeled in computers to find successful solutions for difficult problems.*

- By taking a population of possible answers and evaluating them against the best possible solution, the *fittest* individuals of the population are determined.
- After evaluation, *combining* and *mutating*, the members of the current generation generate a new population.
- This new generation is then evaluated and the process is repeated, until we have approximated optimal solution.

TSP cont.

What was our TSP problem?

- population of possible answers:

  Possible tours: “1-3-5-6-7-4-2-1”

- evaluate – best possible solution:

  Shortest tour!

- generate a new population by **combining** and **mutating**

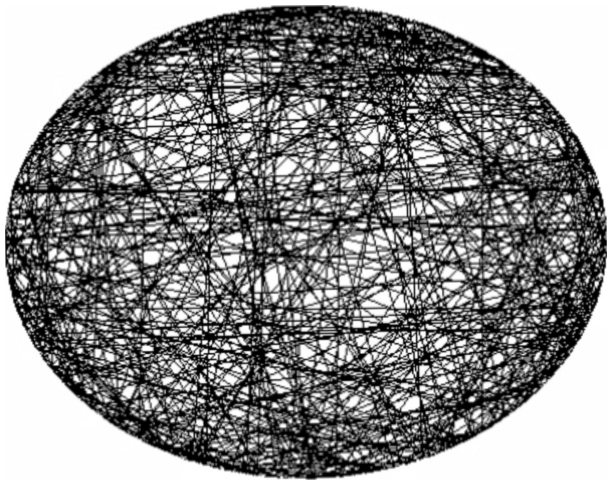
- evaluate new population, “*rinse, repeat*”

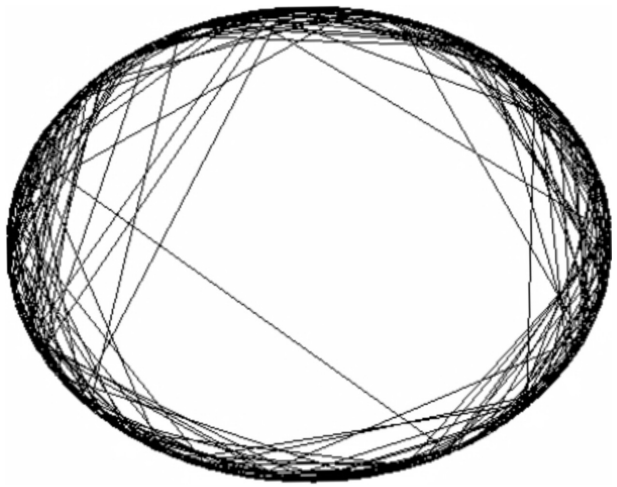
TSP search space

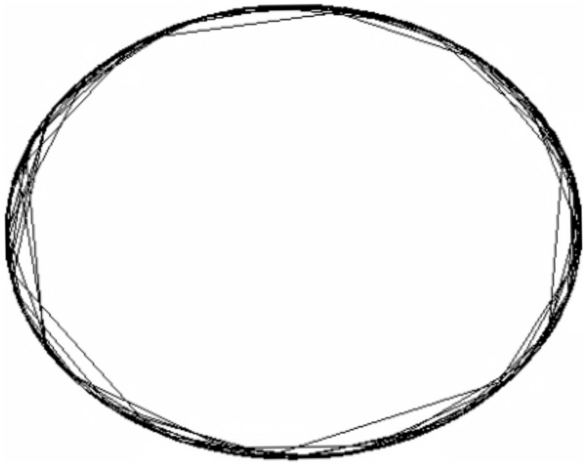
500 cities on a circle – simple?

1220136825991110068701238785423046926253574342803192842192413588385845373
1538819976054964475022032818630136164771482035841633787220781772004807852
0515932928547790757193933060377296085908627042917454788242491272634430567
0173270769461062802310452644218878789465754777149863494367781037644274033
8273653974713864778784954384895955375379904232410612713269843277457155463
0997720278101456108118837370953101635632443298702956389662891165897476957
2087926928871281780070265174507768410719624390394322536422605234945850129
9185715012487069615681416253590566934238130088562492468915641267756544818
8650659384795177536089400574523894033579847636394490531306232374906644504
8824665075946735862074637925184200459369692981022263971952597190945217823
3317569345815085523328207628200234026269078983424517120062077146409794561
1612762914595123722991334016955236385094288559201872743379517301458635757
0828355780158735432768888680120399882384702151467605445407663535984174430
4801289383138968816394874696588175045069263653381750554781286400000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000

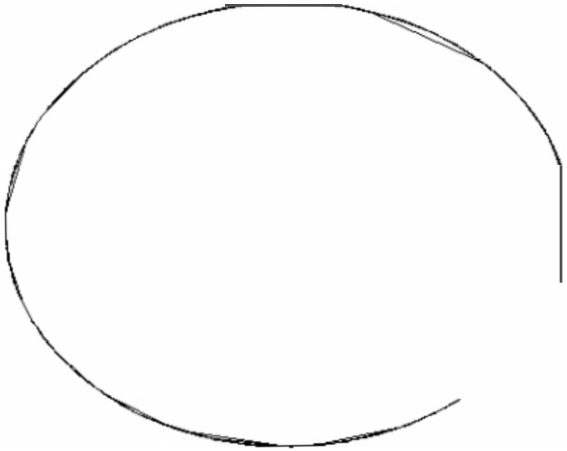
TSP example







after ca. 3000 generations (popSize 500)



Applications of GA

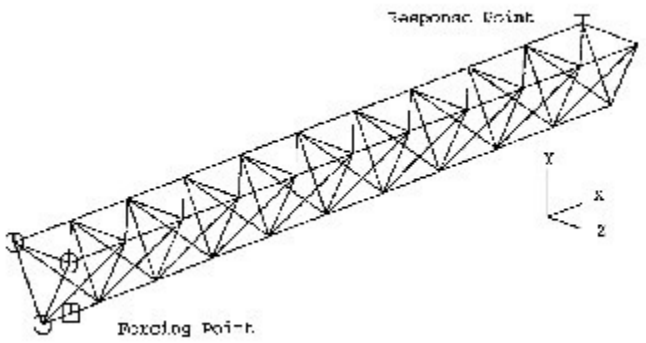
Applications

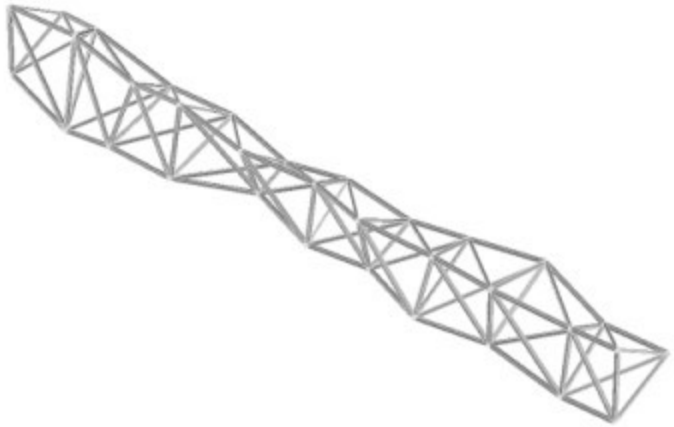
- **What problems can we solve with a GA?**
  - Optimization & Design
    - TSP, function optimizations, time tables..
  - Approximate NP-Hard problems
  - Simulation
  - Modeling, system identification
    - Evolutionary machine learning

"Mom and dad jet engine can get together and have baby jet engines. You find the ones that work better, mate them, and just keep going." - *Goldberg*

Example: Shape optimization

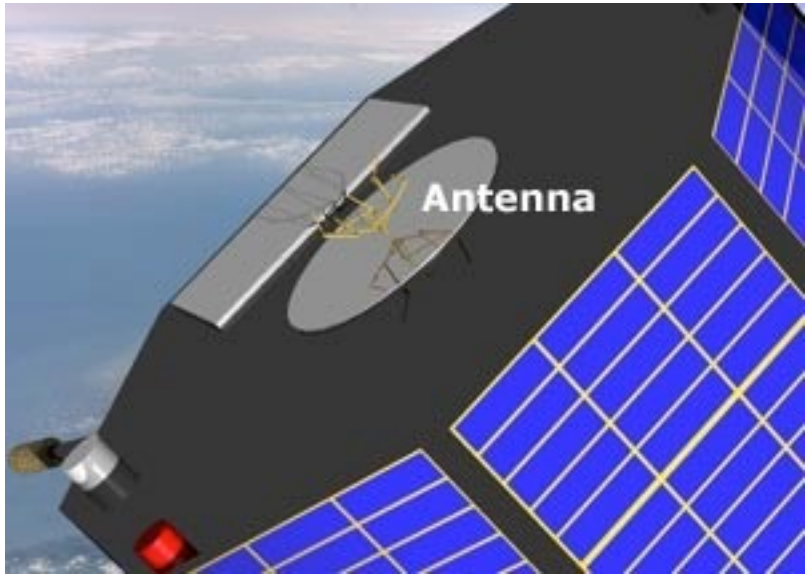
- NASA: Satellite truss or boom design
  - the design of satellite trusses with enhanced vibration isolation characteristics
  - produced using Genetic Algorithm methods and a highly customized vibrational energy flow code
- Evolutionary Design: **20,000% better!!!**

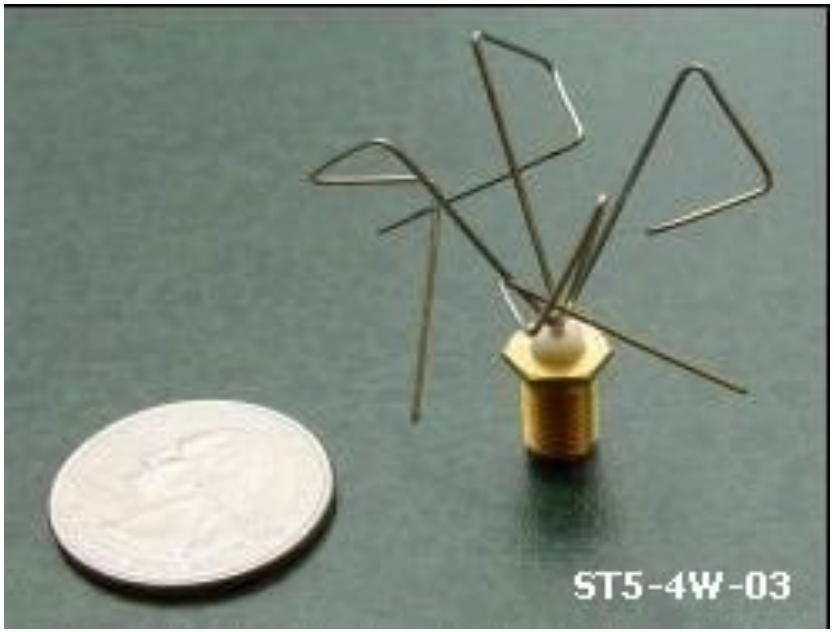




Example: Antenna design (NASA)

- Encode antenna structure into a genome
- Use GA to evolve an antenna
- Evaluation: Convert the genotype into an antenna structure
  - Simulate using antenna simulation software

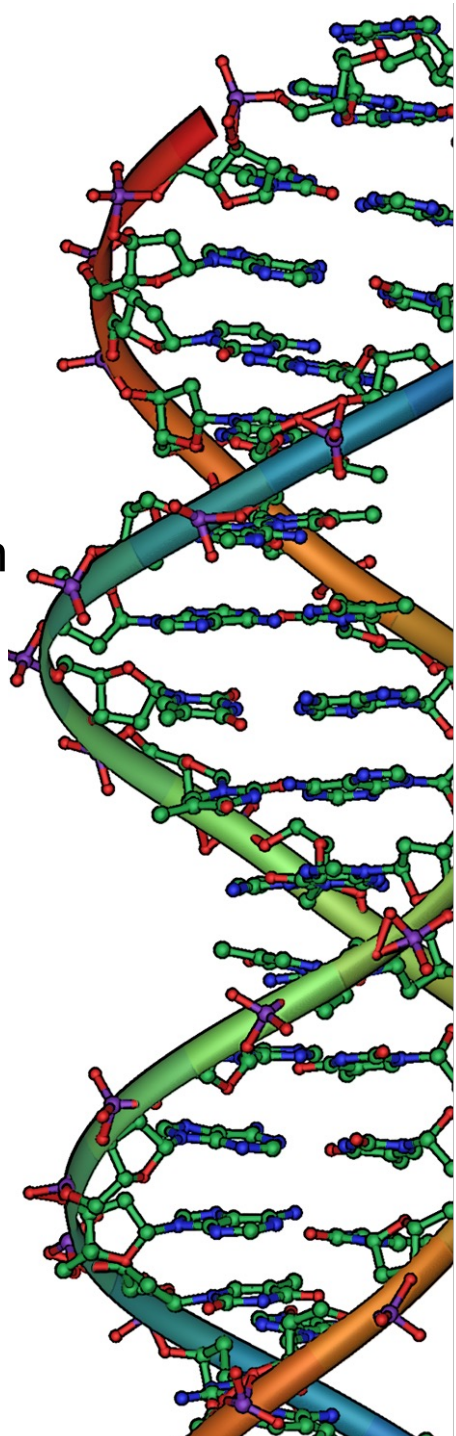




GA terminology

GA terminology:

- *Population*
- *Individuals* – Chromosomes – Representation
- *Generations* – Evolution
- *Fitness* – How “fit” is the individual?
- ***Development – Selection – Reproduction***



Evolution

–from one generation to the next

- *Duplication*? -> No improvement
- *Randomly* produced? -> Past advances are not preserved

- Fitness is not preserved by duplication
- Observed variety is not due to random variation

- *So, how do we retain past successes?*
- *How do we use them to increase the probability of fit (and novel) variants?*

- ***Fitness proportional reproduction & genetic operators***

What should our GA do?

- Recombine 'surface' similarities among the fittest individuals...
- Combine 'good ideas' from different good individuals...
- *... because certain substructures in good individuals cause their high fitness, and recombining such 'good ideas' may lead to better individuals...*

Genetic Operators

Genetic operators

- Fitness-proportional **reproduction**
- Genetic recombination -> **Crossover**
- **Mutation** -> “copy errors” -> additions / deletions of base pairs

GA use for finding suitable framework architecture

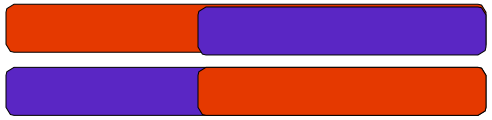
Genetic operators: - Crossover

- **Crossover**
  - Sexual reproduction (pass on 50% of your genes)
- **Benefits?**
  - Stability – occurs between very similar DNA segments
    - Leads to clearly defined species!
  - Stability - lengths of DNA molecules are preserved
  - Variability – combining “good” ideas

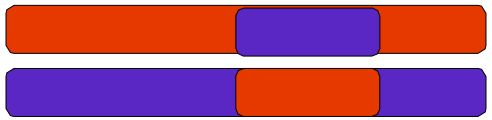
Genetic operators:
- Crossover

- **Crossover**

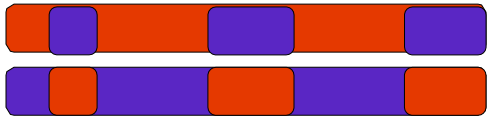
  - Single Point



  - Two point



  - Uniform



Genetic operators: - Mutation

- **Mutation**
  - Insert / Delete / Substitute
- Mass mutation -> harmful! (e.g. genetic disorder)
- Small changes -> beneficial! -> Variations!

- ***Help to better adapt to changes in their environment!***

Genetic operators: - Mutation

- **Mutation**
  - *Inversion*
    0 0 1 1 1 0 1 0 0 1 => 0 0 1 0 1 0 1 0 0 1
  - *Substitution*
    1 2 3 4 8 7 6 9 5 => 1 2 9 4 8 7 6 3 5
  - *Update*
    8.3 1.2 4.3 2.2 2.7 7.1 => 8.3 1.2 4.3 2.3 2.7 7.1
  - *Insertion*
    1 2 3 4 8 7 6 9 5 (select random)
    1 2 3 4 9 8 7 6 5 (insert at random location)

Genetic operators:
- Reproduction

- **Fitness-proportional reproduction / Selection strategies**
- Again; survival of the fittest
- Population fitness $F = \sum_{k=1}^{popSize} f_k$

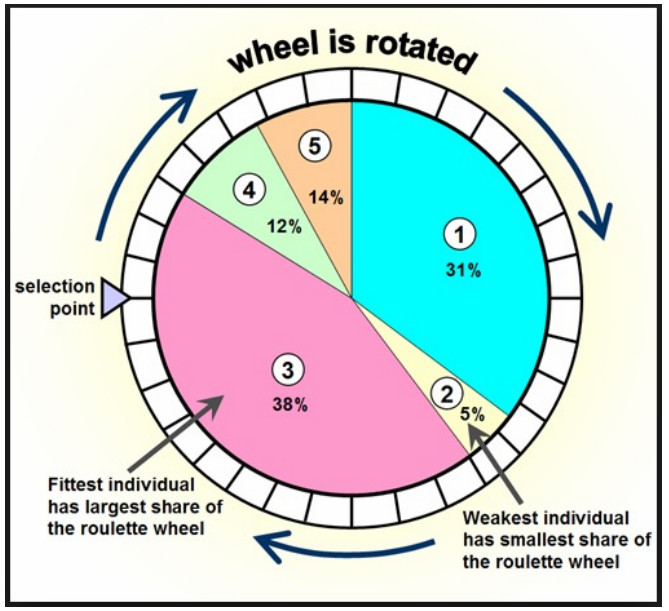
- Roulette Wheel selection
- Rank selection
- Tournament selection

- Elitism
  - First copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way.
  - Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution.

Roulette Wheel selection

- Rank individuals in increasing order of fitness, from 1 to popsize (n)
- Probability of selecting individual $v_i = f_i / F$

```
for (int k = 0; k < population.size(); k++) {
 sum += (population.get(k).getFitness() / populationFitness);
 if (sum >= random)
     return population.get(k);
}
```

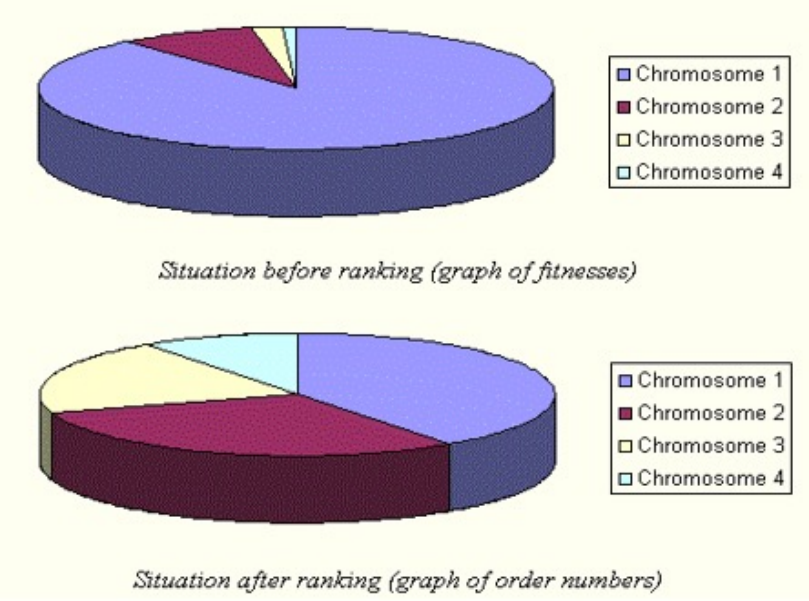


Rank selection

- Rank individuals in increasing order of fitness, from 1 to popsize (n)
- Better when fitness differs a lot

=> No super individuals

```
for (int k = 0; k < population.size(); k++) {
    double pk = Math.pow(selectionPressure, k + 1);
    sum += pk;
    if (sum >= random)
        return population.get(k);
}
```



The components of a GA

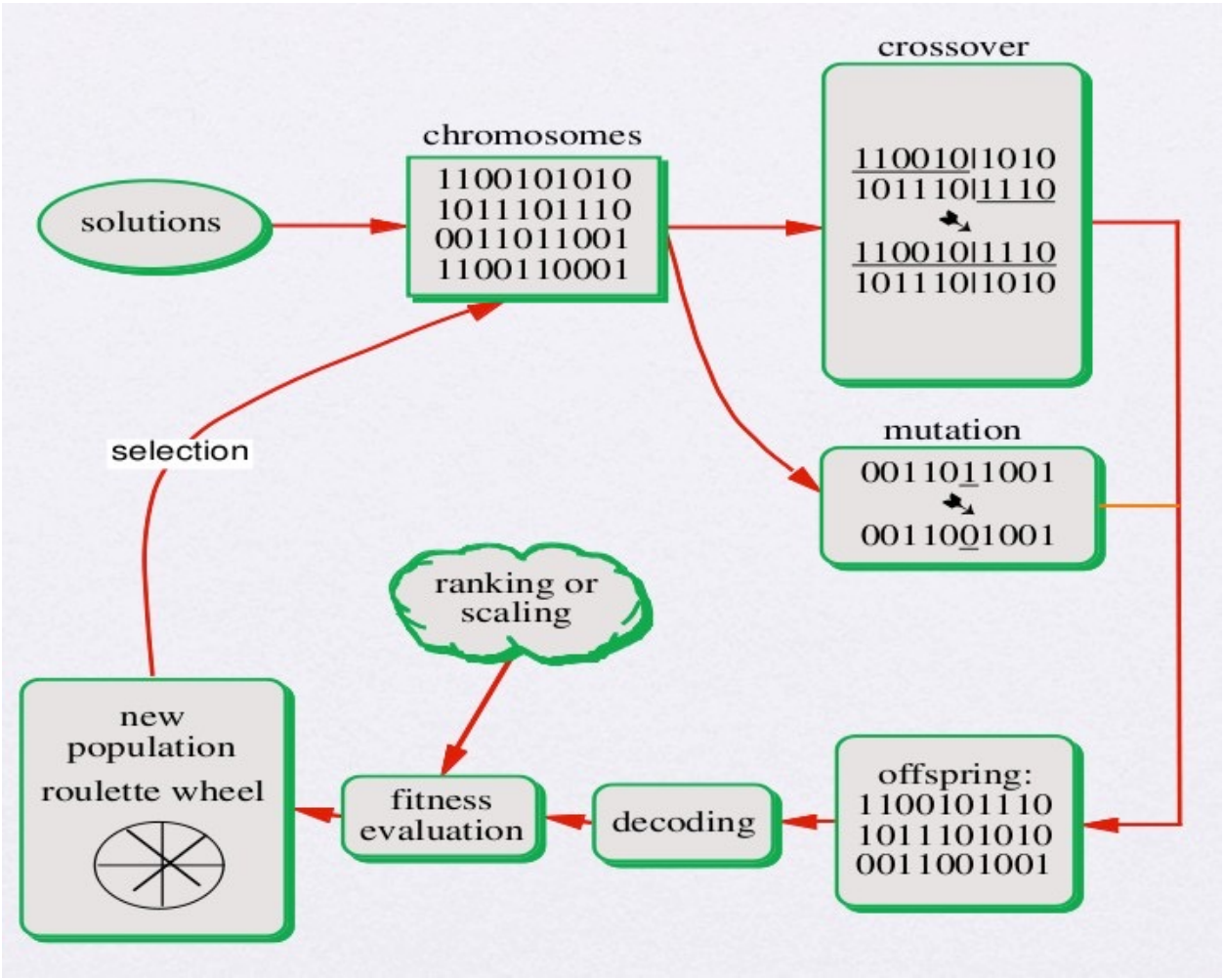
The components of a GA

- **Representation** / Encoding of a Chromosome
  - Binary, Permutation, Value...
- **Initialization**
- **Evaluation** / Fitness function
- Genetic operators / Selection

- **Parameters**
  - Population size
  - Xover probability
  - Mutation probability
  - ...

offspring - solutions

Generic GA


crossover
chromosomes
1100101010
1011101110
0011011001
1100110001
solutions
110010|1010
101110|1110
110010|1110
101110|1010
mutation
0011011001
0011001001
selection
ranking or scaling
new population
roulette wheel
fitness evaluation
decoding
offspring:
1100101110
1011101010
0011001001

Generic GA – Pseudo code

- 1.Choose initial population - random
- 2.Evaluate the fitness of each individual in the population
- 3.Repeat until termination
  - Select best-ranking individuals to reproduce (*parents*)
  - Breed new generation through crossover and mutation (genetic operations) and give birth to offspring
  - Evaluate the individual fitness of the offspring
  - Select individuals for next generation

Some recommendations

- "Generally good parameters"
  - High **crossover** probability! (≈ 0.6)
  - Low **mutation** probability! (≈ 0.1 to 0.001)
  - **Population size**? - usually bigger is better!

- Chromosome / String size -> determines search space!
  - e.g. 30 bits? -> search space = 2^{30} = 1.07 billion points

Problems with GAs

- No convergence guarantee

Sometimes GA gets stuck on a solution that is "okay" but not the best

- Premature convergence

Premature = too early

- Disadvantages:
  - May be difficult to choose encoding
  - May be difficult to define the fitness function
  - May be slow (not really a problem with todays computers)

Why does it work?

Why does GAs work?

- Directed and stochastic search!
  - Population of potential solutions (randomly spread out)
  - "Re-use" relatively good (surviving) solutions
  - Exchange information among these relatively good solutions
  - Search in multiple directions – in parallel!
- Exploration & Exploitation
- Start with an "*open mind*" - decisions based on randomness
  - All possible search pathways are theoretically open to a GA
  - ***"Uncover solutions of startling and unexpected creativity that might never have occurred to human designers"***
- *Once you have your GA; simple to solve new problems!!*

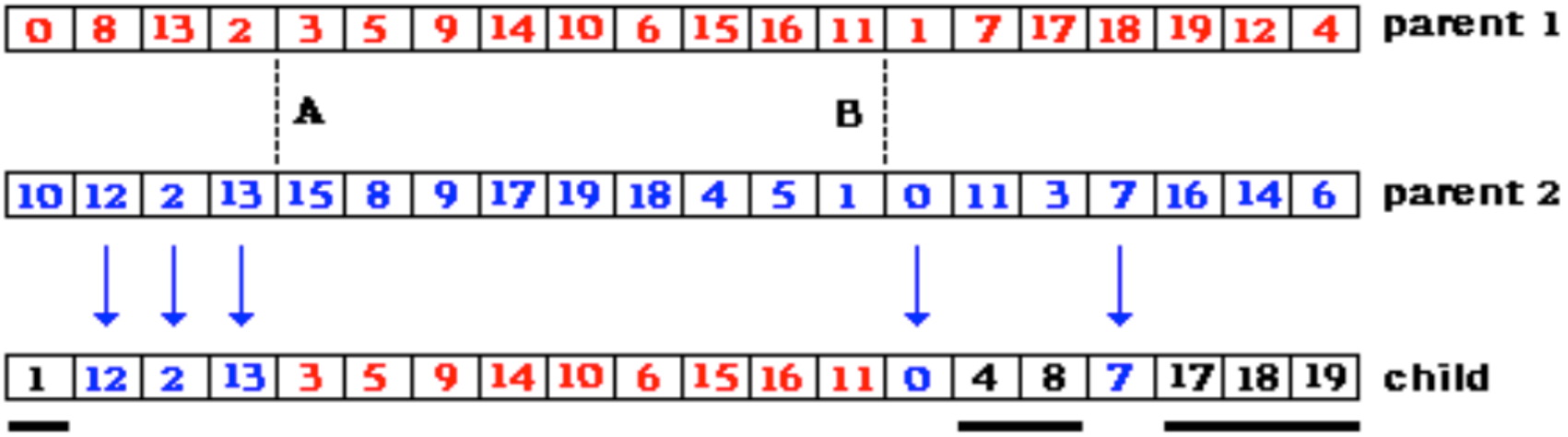
Exploration = Look at new places (randomness and mutation).

Exploitation = Improve good areas (crossover and selection)

Workshop

Why GA Works?	Simple Meaning
Directed + Random search	Randomly try ideas, but keep good ones
Exploration + Exploitation	Balance between trying new things and improving old
Open mind	Start without bias, so discover crazy good solutions
Flexible framework	Once built, can solve many different problems easily

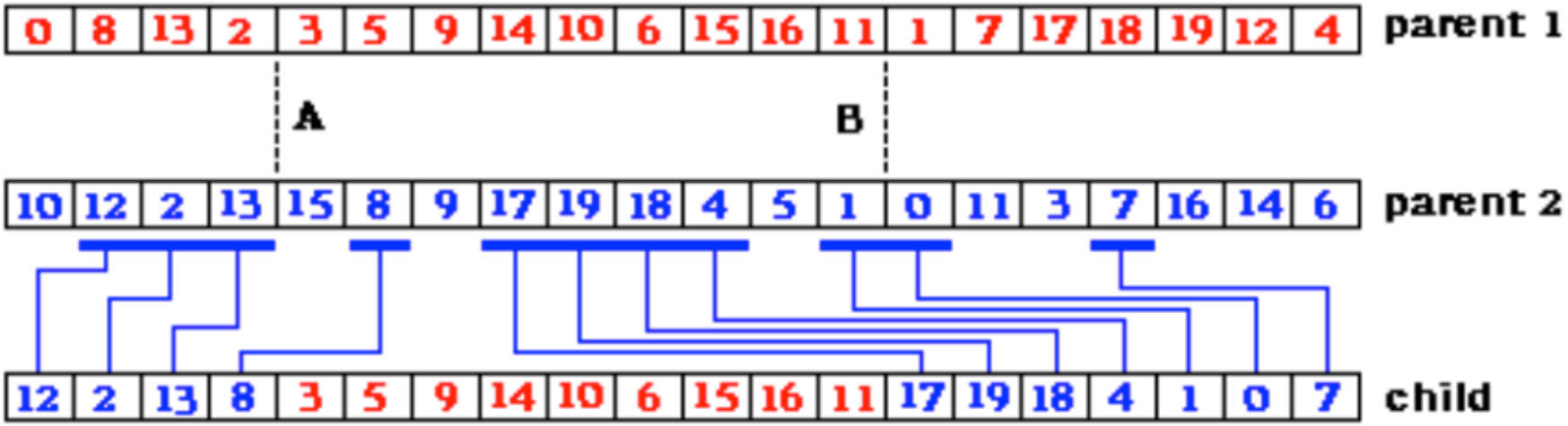
- PMX (Partially Mapped Xover)



Workshop

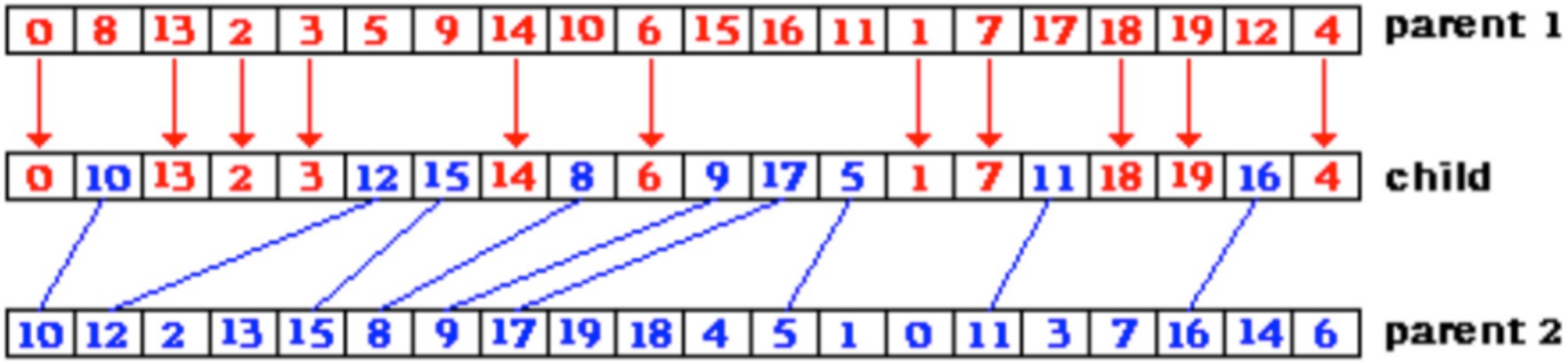
: Fill the rest from Parent 2 (blue numbers) IN ORDER, skipping numbers already in the child.

- Order Xover



Workshop

- Position-Based Xover



Workshop

- Mutation:
  - Inversion
    1 2 3 4 5 6 7 8 9 ==> 1 2 6 5 4 3 7 8 9
  - Insertion
    1 2 3 4 5 6 7 8 9 (select random city) ==>
    1 2 6 3 4 5 7 8 9 (insert in random spot)
  - Reciprocal Exchange
    1 2 3 4 5 6 7 8 9 ==> 1 2 6 4 5 3 7 8 9

Bibliography

- “Adaption in Natural and Artificial Systems”, Holland, 1975
- Evolutionary Computation, Lecture notes, F. Oppacher, Carleton U.
- http://www2.econ.iastate.edu/tesfatsi/holland.gaintro.htm
- https://www.youtube.com/watch?v=Z3668A0zLCM
- http://ti.arc.nasa.gov/projects/esg/research/antenna.htm
- http://www.talkorigins.org/faqs/genalg/genalg.html
- http://www.obitko.com/tutorials/genetic-algorithms/