TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

PROBLEM SOLVING AND SEARCH IN ARTIFICIAL INTELLIGENCE

Lecture 9 Evolutionary Algorithms

Sarah Gaggl

Dresden, 17th December 2019

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Agenda

Outline

Motivation

- Search algorithms so far modified (resp. constructed) one single solution.
- Process a complete solution or construct the final solution from smaller building blocks.
- There is a single best solution to be improved.

Motivation

- Search algorithms so far modified (resp. constructed) one single solution.
- Process a complete solution or construct the final solution from smaller building blocks.
- There is a single best solution to be improved.

New Idea

- Work on a population of solutions
- Let the solutions compete against each other
- Use random variation to search for new solutions

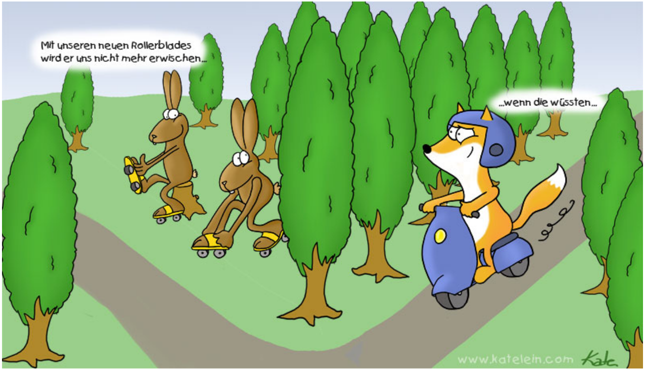
Rabbits and Foxes

- Some rabbits are faster and smarter - they don't get eaten by foxes
- They do what rabbits do best: make more rabbits
- Breeding mixes the rabbits' genetic material
- Every once in a while: mutation
- Over generations, rabbits become faster and smarter

Rabbits and Foxes ctd.

- The same happens with foxes
- They are forced to get better at finding a meal

Rabbits and Foxes ctd.



Outline

Evolutionary Algorithms (EAs)

- A population of individuals exists in an environment with limited resources
- Competition for resources causes selection of fitter individuals that are better adapted to environment
- These individuals act as seeds for generation of new individuals through recombination and mutation
- New individuals have their fitness evaluated and compete (possibly also with parents) for survival
- Over time natural selection causes a rise in the fitness of the population

Evolutionary Algorithms (EAs)

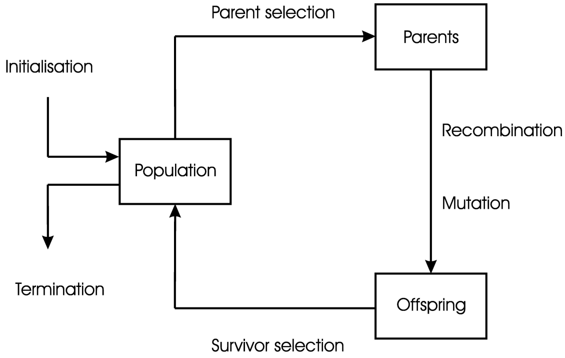
- A population of individuals exists in an environment with limited resources
- Competition for resources causes selection of fitter individuals that are better adapted to environment
- These individuals act as seeds for generation of new individuals through recombination and mutation
- New individuals have their fitness evaluated and compete (possibly also with parents) for survival
- Over time natural selection causes a rise in the fitness of the population

Facts on EAs

- EAs are generate and test algorithms
- They are stochastic and population-based
- Variation operators (recombination and mutation) create necessary diversity
- Selection reduces diversity and acts as a force pushing quality

Outline

General Schema of EAs



Outline

Structure of an EA

Algorithm 1: evolutionary algorithm

```
INITIALISE population with random candidate solutions
EVALUATE each candidate
while not TERMINATION-CONDITION is satisfied do
    SELECT parents
    RECOMBINE pairs of parents
    MUTATE the resulting offspring
    EVALUATE new candidates
    SELECT individuals for the next generation
end while
```

Types of EAs

Historically different types of EAs have been associated with different representations.

- Binary strings: Genetic Algorithms
- Real-valued vectors: Evolution Strategies
- Finite state machines: Evolutionary Programming
- Trees: Genetic Programming

Types of EAs

Historically different types of EAs have been associated with different representations.

- Binary strings: Genetic Algorithms
- Real-valued vectors: Evolution Strategies
- Finite state machines: Evolutionary Programming
- Trees: Genetic Programming

Technically

- choose representation to suit problem
- choose variation operators to suit representation
- selection operators only use fitness – so they are independent of representation

Outline

Components of EAs

- Representation (definition of individuals)
- Evaluation function (or fitness function)
- Population
- Parent selection mechanism
- Variation operators, recombination and mutation
- Survivor selection mechanism (replacement)

Necessary

- Initialisation procedure
- Termination condition

Outline

Representation

- Candidate solutions (individuals) exist in phenotype space
- They are encoded in chromosomes, which exist in genotype space
  - Encoding: phenotype $\rightarrow$ genotype (not necessarily one-to-one)
  - Decoding: genotype $\rightarrow$ phenotype (mutst be one-to-one)
- Chromosomes contain genes, which are in (usually fixed) positions called loci (sing. locus) and have a value (allele)

Representation

- Candidate solutions (individuals) exist in phenotype space
- They are encoded in chromosomes, which exist in genotype space
  - Encoding: phenotype $\rightarrow$ genotype (not necessarily one-to-one)
  - Decoding: genotype $\rightarrow$ phenotype (mutst be one-to-one)
- Chromosomes contain genes, which are in (usually fixed) positions called loci (sing. locus) and have a value (allele)

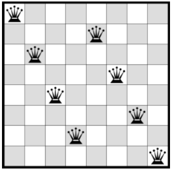
To find global optimum, every feasible solution must be represented in genotype space.

Representation

- Candidate solutions (individuals) exist in phenotype space
- They are encoded in chromosomes, which exist in genotype space
  - Encoding: phenotype $\rightarrow$ genotype (not necessarily one-to-one)
  - Decoding: genotype $\rightarrow$ phenotype (mutst be one-to-one)
- Chromosomes contain genes, which are in (usually fixed) positions called loci (sing. locus) and have a value (allele)

To find global optimum, every feasible solution must be represented in genotype space.

Example (8-Queens)



1	3	5	7	2	4	6	8

- Phenotype: a board configuration
- Genotype: a permutation of the numbers $1 - 8$

Outline

Evaluation (Fitness) Function

- Represents the requirements the population should adopt to
- aka quality function or objective function
- Assigns a single real-valued fitness to each phenotype which forms the basis for selection
- Typically we talk about fitness being maximized

Evaluation (Fitness) Function

- Represents the requirements the population should adopt to
- aka quality function or objective function
- Assigns a single real-valued fitness to each phenotype which forms the basis for selection
- Typically we talk about fitness being maximized

Example (8-Queens ctd.)

- Penalty of one queen: number of queens she can check
- Penalty of a configuration: sum of penalties of all queens

$\Rightarrow$ Penalty needs to be minimized

$\Rightarrow$ Fitness of a configuration: inverse penalty to be maximized

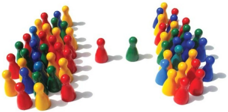
Outline

Population

- Holds (representations of) possible solutions
- Usually has a fixed size and is a multiset of genotypes
- Some sophisticated EAs also assert a spatial structure on the population e.g. a grid
- Selection operators usually take whole population into account i.e. reproductive probabilities are relative to current generation
- Diversity of a population refers to the number of different solutions
- No single measure for diversity exists
- Typically one refers to number of different fitness values/phenotypes/genotypes present

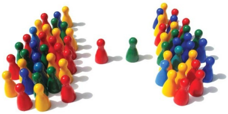
Outline

Parent Selection Mechanism



- Distinguish among individuals based on their quality – allow better individuals to become parents of next generation
- Individual is a parent if it has been selected to create offspring
- Responsible for pushing quality improvements
- Usually probabilistic
  - high quality solutions more likely to become parents than low quality
  - BUT: not guaranteed
  - even worst in current population has non-zero probability of becoming a parent
- Stochastic nature can aid escape from local optima

Parent Selection Mechanism



- Distinguish among individuals based on their quality – allow better individuals to become parents of next generation
- Individual is a parent if it has been selected to create offspring
- Responsible for pushing quality improvements
- Usually probabilistic
  - high quality solutions more likely to become parents than low quality
  - BUT: not guaranteed
  - even worst in current population has non-zero probability of becoming a parent
- Stochastic nature can aid escape from local optima

Example (8-Queens ctd.)

Pick 5 parents randomly and take the two best to generate offspring.

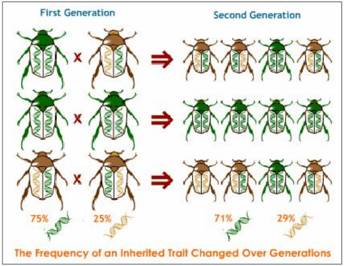
Outline

Variation Operators

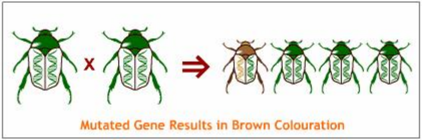
- Role is to generate new candidate solutions
- Usually divided into two types according to their arity

Arity 1: mutation operator
Arity >1: recombination operator; arity=2 typically called crossover

- Debate about relative importance of recombination and mutation
  - Nowadays most EAs use both
  - Choice of particular variation operator is representation dependent



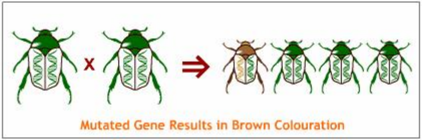
Mutation



- Acts on one genotype and delivers another
- Element of randomness is essential and differentiates if from other unary heuristic operators
- Generating a child amounts to stepping to a new point in search space

$\Rightarrow$ Mutation may guarantee connectedness of search space

Mutation



- Acts on one genotype and delivers another
- Element of randomness is essential and differentiates if from other unary heuristic operators
- Generating a child amounts to stepping to a new point in search space

⇒ Mutation may guarantee connectedness of search space

Example (8-Queens ctd.)

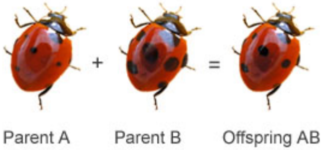
Swap values of two randomly chosen positions

1	3	5	7	2	4	6	8

⇒

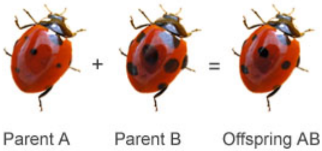
1	3	6	7	2	4	5	8

Recombination



- Merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as parents
- Hope that some are better by combining elements of genotypes that lead to good traits

Recombination



- Merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as parents
- Hope that some are better by combining elements of genotypes that lead to good traits

Note

- Variation operators are representation dependent
- Different representations require different variation operators

Example (8-Queens ctd.)

Combine two permutations into two new permutations

- Choose random crossover point
- Copy fist parts into children
- Create second part by inserting values from other parent
  - in order they appear there
  - beginning after crossover point
  - skipping values already in child

Example (8-Queens ctd.)

Combine two permutations into two new permutations

- Choose random crossover point
- Copy fist parts into children
- Create second part by inserting values from other parent
  - in order they appear there
  - beginning after crossover point
  - skipping values already in child

Note

Offspring needs to be a permutation (as genotype is permutation)!

Example (8-Queens ctd.)

Combine two permutations into two new permutations

- Choose random crossover point
- Copy fist parts into children
- Create second part by inserting values from other parent
  - in order they appear there
  - beginning after crossover point
  - skipping values already in child

Note

Offspring needs to be a permutation (as genotype is permutation)!

1	3	5	7	2	4	6	8
8	7	6	5	4	3	2	1

$\Rightarrow$

1	3	5	4	2	8	7	6
8	7	6	2	4	1	3	5

Example (8-Queens ctd.)

Combine two permutations into two new permutations

- Choose random crossover point
- Copy fist parts into children
- Create second part by inserting values from other parent
  - in order they appear there
  - beginning after crossover point
  - skipping values already in child

Note

Offspring needs to be a permutation (as genotype is permutation)!

1	3	5	7	2	4	6	8
8	7	6	5	4	3	2	1

$\Rightarrow$

1	3	5	4	2	8	7	6
8	7	6	2	4	1	3	5

Children inherit genetic material from both parents!

Outline

Survivor Selection

- aka replacement
- Most EAs use fixed population size
- Often deterministic
  - Fitness-based: e.g. rank parents and offspring and take the best
  - Age-based: make as many offspring as parents and delete all parents
  - Combinations of the former two (elitism)

Survivor Selection

- aka replacement
- Most EAs use fixed population size
- Often deterministic
  - Fitness-based: e.g. rank parents and offspring and take the best
  - Age-based: make as many offspring as parents and delete all parents
  - Combinations of the former two (elitism)

Example (8-Queens ctd.)

Merge population and offspring – rank them according to fitness – delete the worst two.

Outline

Initialization/Termination

Initialization

- Usually done at random
- Needs to ensure even spread and mixture of possible allele values
- Can include existing solutions, or use problem-specific heuristics to seed the population

Initialization/Termination

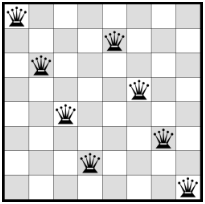
Initialization

- Usually done at random
- Needs to ensure even spread and mixture of possible allele values
- Can include existing solutions, or use problem-specific heuristics to seed the population

Termination Condition

- Checked every iteration
- Reaching some (known/hoped for) fitness
- Reaching some maximum allowed number of generations
- Reaching some minimum level of diversity
- Reaching some specified number of generations without fitness improvement

Example (8-Queens ctd.)



- Initial population: randomly generated permutations
- Termination condition: solution or 10000 fitness evaluations
- Population size: 100
- Recombination probability: 100%
- Mutation probability: 80%

Outline

Working of EAs

Typical progress of an EA illustrated in terms of population distribution

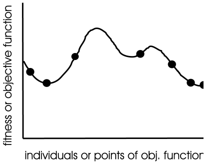


Figure: begin

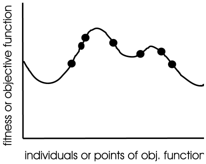


Figure: halfway

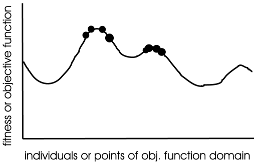
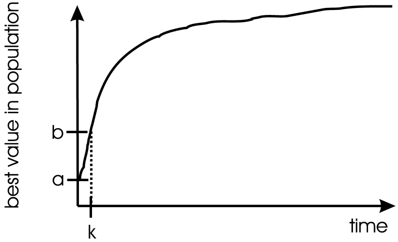


Figure: end

Working of EAs ctd.

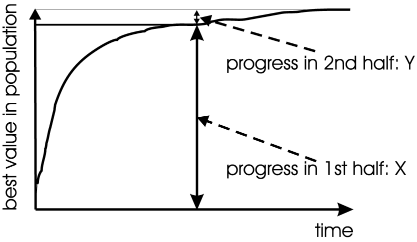
Typical progress in terms of development of best fitness value within population in time.

No Need for Heuristic Initialisation



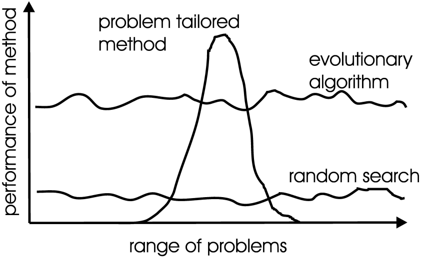
- Level a: best fitness in a randomly initialised population
- Level b: heuristic initialisation
- After k generations same level reached

Termination Conditions



- Divide the run into two equally long sections
- Fitness increases in the first half X
- Progress in second half Y is much smaller
- Due to anytime behaviour, efforts spent after a certain time may not result in better solution quality

Performance from Global Perspective



- Performance on a wide range of problems
- EAs are robust problem solving tools
- For most problems a problem-specific tool may
  - perform better than a generic search algorithm on most instances
  - have limited utility
  - not do well on all instances
- EAs provide an evenly good performance over a range of problems and instances

Outline

Summary

- Idea for EAs come from evolution theory
- Components
  - Representation (definition of individuals)
  - Evaluation function (or fitness function)
  - Population
  - Parent selection mechanism
  - Variation operators, recombination and mutation
  - Survivor selection mechanism (replacement)
  - Initialisation and termination condition
- Performance

References

Zbigniew Michalewicz and David B. Fogel.
How to Solve It: Modern Heuristics, volume 2. Springer, 2004.

A.E. Eiben and J.E. Smith.
Introduction to Evolutionary Computing, Springer, 2003.