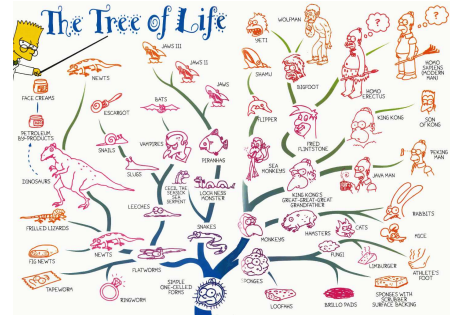


Evolutionary Computation

Christian Blum

Artificial Intelligence Research Institute (IIIA-CSIC)
Bellaterra, Spain

Evolution



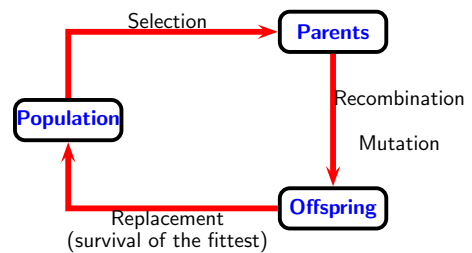
Evolution

Optimization process based on ...

- **Selection**, for example:
 1. Females prefer males with a very different immune system
 2. Plants with a nicer flower (or a better smell) attract more flies
- **Survival of the fittest**, for example:
 1. Slow rabbits are more easily caught by foxes
 2. In a cold winter the one with more fat reserves survive
- **Recombination and mutation** introduce innovation and diversity into a population

Evolution

The evolutionary cycle



Technical algorithm

Pseudo code

```
P ← GenerateInitialPopulation()
Evaluate(P)
while termination conditions not met do
  P' ← Recombine(P)
  P'' ← Mutate(P')
  Evaluate(P'')
  P ← Select(P'', P)    (Replacement)
end while
```

Technical algorithm

Notation

- **Selection:**
 1. The choice of the individuals (i.e., solutions) that act as parents to produce offspring for the next generation
 2. The choice of the individuals that replace old individuals. (Also called **replacement**).

Technical algorithm

Notation (continued)

- ▶ **Recombination:** The recombination (or crossover) of two or more individuals for producing offspring.
- ▶ **Mutation:** The change that an individual undergoes (either randomly or in a heuristically guided way)
- ▶ **Evaluation:** The evaluation of the fitness of an individual (usually inversely proportional to the objective function)

Technical algorithm

Historical note

EC methods are **among the oldest** metaheuristic methods

Families of evolutionary algorithms

- ▶ **Evolutionary Programming (EPs)** introduced by [Fogel et al., 1966]
- ▶ **Evolutionary Strategies (ESs)** introduced by [Rechenberg, 1973]
- ▶ **Genetic Algorithms (GAs)** introduced by [Holland, 1975], [Goldberg, 1989]
- ▶ **Genetic programming (GP)** introduced by [Koza, 1992]

Technical algorithm

Families of evolutionary algorithms (continued)

- ▶ **Scatter search (SS)** introduced by [Glover, 1977]
- ▶ **Estimation of distribution algorithms (EDAs)** introduced in the [90ties]
- ▶ **Differential evolution (DE)** introduced by [Storn, 1995]

Technical algorithm

Differences between the different EC families

- ▶ **Purpose:**
 1. **Continuous optimization:** ESs, SS, EDAs, DE
 2. **Combinatorial optimization:** GAs, GP, SS
 3. **Programs:** GP
- ▶ **Solution representation.** Examples:
 1. **Bit strings:** GAs
 2. **Tree structures:** GP
- ▶ **Evolutionary operators:**
 1. **Exclusively mutation:** ESs
 2. **2-parent crossover:** GAs

Literature

Seminal books

- ▶ **[Fogel et al., 1966]** L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966
- ▶ **[Rechenberg, 1973]** I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973
- ▶ **[Holland, 1975]** J. H. Holland. *Adaption in natural and artificial systems*. The University of Michigan Press, Ann Harbor, MI, 1975
- ▶ **[Goldberg, 1989]** D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, Reading, MA, 1989

Good introduction to EC algorithms for CO

- ▶ **[Hertz and Kobler, 2000]** A. Hertz and D. Kobler. A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126:1–12, 2000

Our view on evolutionary computation

Note

Before dealing with some specific families of EC algorithms, we will look at EC from a higher level (the problem solving level).

Seven algorithm features

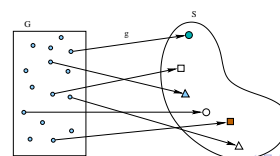
- ▶ Representation of the individuals (solution representation)
- ▶ Evolution process (replacement strategy)
- ▶ Neighborhood structure (which individuals can be recombined)
- ▶ Information sources (how to recombine?)
- ▶ Infeasibility
- ▶ Intensification strategy
- ▶ Diversification strategy

Geno-/Phenotype space

Two different spaces

1. **Genotype space \mathcal{G} :** The space of all individuals
2. **Phenotype space \mathcal{S} :** The space of all solutions to the problem

Mapping



Geno-/Phenotype space

Requirements on the mapping

- ▶ **Surjectivity:** For each solution in the phenotype space there should be an individual from the genotype space that maps to it
- ▶ **Non-disruptiveness:** Phenotypes that are similar should map to similar solutions

Note

The representation of individuals is **crucial** for the success of an EC algorithm.

Examples for direct representations

- ▶ Bit-strings
- ▶ Arrays of real numbers (for continuous optimization)
- ▶ Tree structures

On the board!!

Examples for indirect representations

- ▶ Permutations of n numbers
- ▶ Arrays of real numbers (gray encoding)
- ▶ String of greedy function identifiers for a solution generator

On the board!!

A bad example for an indirect encoding: Prüfer numbers encoding spanning trees

The evolutionary process

What is the evolutionary process?

The type of the evolution process determines the way of assembling the new generation.

Well-known types of evolution processes

- ▶ **Generational replacement:** The offspring entirely replaces the old generation.
- ▶ **Steady state:** The best offspring individuals replace a percentage (≤ 100) of the worst individuals of the old generation.

Note: The population size can be constant or varying

Introduction In general: evolutionary computation Some other issues Some specific EC variants	Representation of the individuals The evolutionary process Neighborhood functions Information sources for recombination Infeasibility Intensification strategies Diversification strategies
---	--

Which solutions can be used for recombination?

Neighborhood functions in EC algorithms

A neighborhood function $\mathcal{N}_{EC} : \mathcal{I} \rightarrow 2^{\mathcal{I}}$ defines, for every individual $i \in \mathcal{I}$, the set of individuals $\mathcal{N}_{EC}(i) \subseteq \mathcal{I}$ which can be recombined with it.

Different types of neighborhood functions

- ▶ **Unstructured population:** every individual can be recombined with any other one (e.g., simple GA)
- ▶ **Structured population:**
 - ▶ **Enforced:** for example in the k -cardinality tree problem
 - ▶ **Intentional:** Parallel evolutionary algorithms (PEAs)

© Christian Blum 19

Introduction In general: evolutionary computation Some other issues Some specific EC variants	Representation of the individuals The evolutionary process Neighborhood functions Information sources for recombination Infeasibility Intensification strategies Diversification strategies
---	--

Parallel evolutionary algorithms (PEAs)

Motivation

The intrinsic parallelism of EC algorithms has given rise to PEAs, whose main feature is the geographical decentralization of the population. One of the aims is to save computation time

Different PEAs

- ▶ **Coarse-grained or distributed EAs (dEAs):** Population is partitioned into different subpopulations (islands). They operate independently, but exchange information.
- ▶ **Fine-grained or cellular EAs (cEAs):** Individuals are placed on a toroidal grid; each individual on a different grid position. Individuals can only recombine with neighboring individuals.

© Christian Blum 20

Introduction In general: evolutionary computation Some other issues Some specific EC variants	Representation of the individuals The evolutionary process Neighborhood functions Information sources for recombination Infeasibility Intensification strategies Diversification strategies
---	--

Different selection schemes

- ▶ **Roulette-wheel-selection:** Select parents for recombination randomly with respect to their objective function values
- ▶ **Tournament selection:** In order to choose a parent choose uniformly at random a number of m individuals. The best one wins.

© Christian Blum 21

Introduction In general: evolutionary computation Some other issues Some specific EC variants	Representation of the individuals The evolutionary process Neighborhood functions Information sources for recombination Infeasibility Intensification strategies Diversification strategies
---	--

Different kinds of recombinations

- ▶ Two-parent crossover (oldest):
 1. Bit-strings: 1-point crossover versus n -point crossover
 2. Permutation-based crossover operators
- ▶ Multi-parent crossover
- ▶ Population statistics-based recombination operators
- ▶ Path constructions between solutions (Path relinking)
- ▶ ...

On the board!!

© Christian Blum 22

Introduction In general: evolutionary computation Some other issues Some specific EC variants	Representation of the individuals The evolutionary process Neighborhood functions Information sources for recombination Infeasibility Intensification strategies Diversification strategies
---	--

How to deal with infeasibility?

Note: The recombination of individuals might result in unfeasible individuals.

Three possible ways of dealing with infeasibility

- ▶ **Reject:** discard infeasible solutions
- ▶ **Punish:** decrease the fitness of individuals violating constraints
- ▶ **Repair:** apply some operators to change the solution trying to obtain a feasible one

© Christian Blum 23

Introduction In general: evolutionary computation Some other issues Some specific EC variants	Representation of the individuals The evolutionary process Neighborhood functions Information sources for recombination Infeasibility Intensification strategies Diversification strategies
---	--

Intensification of the search process

Problem

Therefore, EC algorithms **might be slow** in comparison to "Local Search"-based metaheuristics

Solution

Intensification strategies that apply operators or algorithms to improve the fitness of single individuals

Examples

- ▶ Before replacement, apply iterative improvement local search to each individual of the population (**Memetic Algorithms**)
- ▶ Instead of a random mutation, apply mutation operators based on local search (some steps).

© Christian Blum 24

Diversification of the search process

Problem

Early convergence, which means that all individuals are becoming quickly very similar

Solution

Apply mechanisms to diversify the search. For example:

- ▶ Random mutation
- ▶ Introduce into the population new individuals 'coming' from not yet explored areas of the search space
- ▶ Niching, Crowding

How to generate the initial population

- ▶ Population of **random solutions**
- ▶ Construct a population with a **randomized greedy heuristic** (e.g., GRASP)

TSP example: A simple algorithm

- ▶ **Solution representation**: Each permutation of the n cities is a permutation
- ▶ **Evolution process**: Generational replacement
- ▶ **Neighborhood function**: Each individual can reproduce with each other individual
- ▶ **Selection and information sources**: Tournament selection plus **MPX** crossover (two-parent crossover)
- ▶ **Infeasibility**: Does not apply, because all offspring are feasible
- ▶ **Intensification strategy**: Apply to each offspring produced by crossover an iterative improvement 3-opt local search.
- ▶ **Diversification strategy**: With a probability of 0.1 choose a city randomly and insert it at a random point in the permutation.

When are EC algorithms good/bad?

General advantage

Good in discovering high-quality areas of the search space

General disadvantage

Deficiencies in exploiting good solutions (danger of a disrupting crossover)

Motivation and principle

Motivation

Overcome the drawbacks of usual recombination operators likely to break good building blocks

Principle

Repeat the following two steps:

1. Produce a **probability distribution** over the search space from
2. **Sample the probability distribution** and update it with the sampled solutions

Pseudo-code: EDAs working on populations

```
P ← GenerateInitialPopulation()
while termination criteria not met do
  Psel ← ChooseFrom(P)    {Psel ⊆ P}
  p(· | Psel) ← EstimateProbabilityDistribution(Psel)
  P ← SampleProbabilityDistribution(p(· | Psel))
end while
```

Pseudo-code: EDAs working directly on probability distributions

```

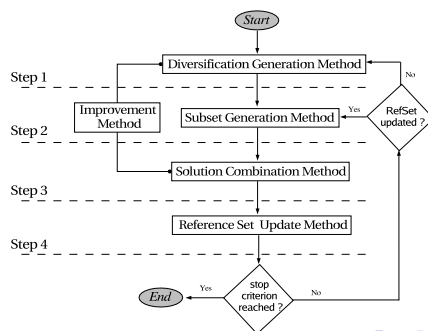
p(·) ← GenerateProbabilityDistribution()
while termination criteria not met do
    P ← SampleProbabilityDistribution(p(·))
    UpdateProbabilityDistribution(P)
end while
    
```

Population-based incremental learning (PBIL)
On the board!!

Some facts

- Provides **explicit intensification and diversification** mechanisms
- **Recombination**:
 1. **Continuous spaces**: Intra- and extrapolation between solutions
 2. **Discrete spaces**: For example, path relinking
- Ideas originally proposed for combining decision rules and constraints in integer programming

Algorithm sketch



Set of reference solutions

Note

The set of reference solutions S_{ref} has the same role as the population in EC algorithms

Management

A solution s can enter S_{ref} iff

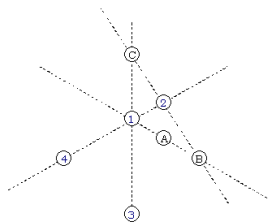
1. Either it is **better than the worst** solution in S_{ref}
2. Or it is more different to the high quality solutions in S_{ref} than other solutions in S_{ref}

Recombination in continuous spaces

Note

Includes **convex** and **non-convex** combinations of existing solutions

Illustration



Differential evolution (DE)

What is DE?

An evolutionary technique for continuous optimization

Algorithm components

- **Population of solutions**: $\vec{x}_i, i = 1, \dots, m$.
Each position $x_{i,j}$ is a real number
- **Mutation**
- **Crossover**
- **Selection**

Mutation in DE

For each solution \vec{x}_i , the following is done:

- ▶ From the current population choose randomly three different solutions: $\vec{x}_r, \vec{x}_s, \vec{x}_l$ ($i \neq r, s, l$)
- ▶ Create a mutated solution: $\vec{x}_i^{mut} := \vec{x}_r + F \cdot (\vec{x}_s - \vec{x}_l)$ with $F \in [0, 2]$

Crossover in DE

For each solution \vec{x}_i do

Create a trial solution \vec{x}_i^{trial} as follows: For all positions $j = 1, \dots, n$

$$x_{i,j}^{trial} := \begin{cases} x_{i,j}^{mut} & : \text{ if } (r(j) \leq q) \text{ or } j = k \\ x_{i,j} & : \text{ otherwise } . \end{cases}$$

Hereby: $q \in [0, 1]$, $r(j)$ are for all j 's random numbers, and k is a random position chosen beforehand

Selection in DE

Possibilities for selection

- ▶ **Most often used:** The trial solution \vec{x}_i^{trial} replaces \vec{x}_i in the population only if better
- ▶ **Also possible:** Even if the trial solution \vec{x}_i^{trial} is worse it still has some probability of replacing \vec{x}_i

Questions?

Questions?