Introduction to Program Synthesis (WS 2024/25)

Chapter 4 - Advanced Methodologies

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany

Announcements

- Exam dates $\rightarrow$ Are about to be scheduled
- Next week $\rightarrow$ Lecture and exercise will be held online (via Zoom)
- July 16 and 17 $\rightarrow$ Online format due to travel

Advanced Methodologies
Evolutionary Algorithms

- Population-based approach to search heuristics $\rightarrow$ Inspired by **biological evolution**
  - $\leadsto$ Adaptation of **Darwinian evolution** $\rightarrow$ Survival of the fittest
  - $\leadsto$ Consider the cost of a **candidate** as **fitness**
  - $\leadsto$ Evolutionary mechanism are performed within an evolutionary algorithms
- Transforming populations of candidate programs to better ones
  - $\leadsto$ *Better* $\rightarrow$ Better fitness
  - $\leadsto$ Apply selection mechanism to mimic natural selection
- Adapting **genetic variation** for **combinatorial search** $\rightarrow$[$\leadsto$] Consider **recombination** and **mutation** as search operators
  - *Evolving* candidate solutions towards an optimum

- Various methodologies have been established in the field of **evolutionary computation**:
  - $\rightsquigarrow$ **Genetic Algorithms** (GA) $\rightarrow$ bitstring representation
  - $\rightsquigarrow$ **Evolutionary Strategies** (ES) $\rightarrow$ real-valued representation
  - $\rightsquigarrow$ **Genetic Programming** (GP) $\rightarrow$ data-structures (trees, graphs, ...)

1954	Barricelli: Evolutionary simulations
1960s - early 1970s	Rechenberg, Schwefel: Evolutionary strategies
same period of time	Fogel: Evolutionary programming
same period of time	Holland: Genetic algorithms
1980s	Forsyth, Cramer, Hicklin: Genetic programming

population	—	a set of individuals
species	—	individuals, which share common characteristics
candidate solution	—	member of the population, part of the search space
individual	—	a candidate, potential solution
breeding	—	the genetic adaption, variation procedure
parent	—	an individual selected for breeding
offspring	—	a candidate solution produced by variation
genotype	—	representation model of an individual, set of genes, vector of numbers
phenotype	—	expression, behavior of the genotype
chromosome	—	a set of genotypes
gene	—	a region of the genotype that encodes functionality
crossover	—	genetic operator, which combines genetic information of two or more parents
	—	to produce new offspring
mutation	—	genetic operator, which varies information on the genome of a individual
	—	(mostly according to a given probability distribution)
selection	—	procedure which choses genomes from a population for later breeding
fitness function	—	an objective function to assess and compare individuals by their fitness
fitness	—	a measurement of the individual's phenotype against the ideal functionality
fitness evaluation	—	a procedure to evaluate the fitness of each individual

Table: List of the important terms which are commonly used in the field of evolutionary algorithms.

Algorithm Example of a simple evolutionary algorithm

```
procedure Evolutionary Algorithm
    initialize(P)                              ▷ Initialize set of candidate solutions
    repeat                                     ▷ Until termination criteria not triggered
        Q ← breed(P)                           ▷ Breed new individuals with crossover and mutation
        Evaluate(Q)                            ▷ Evaluate the fitness of each individual
        if Q meets termination criterion then
            return Q
        end if
        P ← select(P,Q)                        ▷ Select high-fitness individuals
    until P meets termination criterion
    return P
end procedure
```

Advanced Methodologies

Evolutionary Algorithms

- **Evolutionary HC** → Hill-climbing with mutation-based search operator(s)
  - ⇝ Mutation → structural change in a chromosome
  - ⇝ Mutations aggregate across the genome of an individual randomly
- Adaption of mutation for tree-structures → **Subtree mutation**
  - ⇝ Selection of a mutation points by chance
  - ⇝ Exchange of the subsequent subtree with a randomly generated one

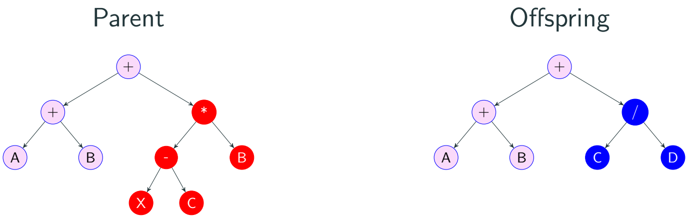


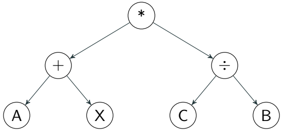
Figure: Subtree mutation

- $(1+\lambda)$ evolutionary strategy $\rightarrow$ Fundamental **search strategy** applied in the field of evolutionary strategies
  - Commonly used in numerical optimization approached with evolutionary computation
  - Breeds λ offspring from one parent in each generation
  - **Elitist** approach $\rightarrow$ **best partial solution** is **selected** from the **population** of $1+\lambda$ candidates
- Evolutionary algorithms that only uses mutation for **breeding** of new candidates
  - The $(1+1)$-ES is the most simple approach
- Adaption of HC to evolutionary computation

Advanced Methodologies
Genetic Programming

Genetic Programming (GP)

- Genetic Programming is a search heuristic.
- Inspired by neo-Darwinian evolution.
- Method for the synthesis of computer programs.
- Traditionally used with parse trees.



$$\mathcal{F} = \{\ +\ ,\ -\ ,\ *,\ \div\ \}$$
$$\mathcal{T} = \{\ A,\ X,\ C,\ B\ \}$$
$$\mathcal{E} = \text{Edges}$$
$$\Psi = (A + X) * (C \div B)$$

Definition (Genetic Program)

A genetic programm $\mathcal{P}$ is an element of $\mathcal{T} \times \mathcal{F} \times \mathcal{E}$:

- $\mathcal{F}$ is a finite non-empty set of functions
- $\mathcal{T}$ is a finite non-empty set of terminals
- $\mathcal{E}$ is a finite non-empty set of edges

Let $\phi : \mathcal{P} \mapsto \Psi$ be a decode function which maps $\mathcal{P}$ to a phenotype Ψ

Definition (Genetic Programming)

Let Θ be a population of $|\Theta|$ individuals and let Ω be the population of the following generation:

- Each individual is represented with a **genetic program** and a **fitness value**.
- Genetic Programming transforms $\Theta \mapsto \Omega$ by the adaptation of **selection**, **recombination** and **mutation**.

Advanced Methodologies
Evolutionary Algorithms

Algorithm $(1{+}\lambda)$ Evolutionary Strategy (ES)

Arguments

λ: Number of offspring

Return

$\mathcal{P}$: Parent individual with best fitness

initialize($\mathcal{P}$) ▷ *Initialize parent individual*
repeat ▷ *Until termination criteria not triggered*
    $\mathcal{Q} \leftarrow$ `breed(`$\mathcal{P}$`)` ▷ *Breed λ offspring by mutation*
    `evaluate(`$\mathcal{Q}$`)` ▷ *Evaluate the fitness of the offspring*
    $\mathcal{Q}^{+} \leftarrow$ `better(`$\mathcal{Q}$,$\mathcal{P}$`)` ▷ *Get individuals which have better fitness as the parent*
    ▷ *If there exist individuals with better fitness*
    if $|\mathcal{Q}^{+}| > 0$ **then**
        $\mathcal{P} \leftarrow$ `best(`$\mathcal{Q}$`)` ▷ *Assign the best offspring as parent*
    end if
until $\mathcal{P}$ *meets termination criterion*
return $\mathcal{P}$ ▷

- Adaption of recombination to tree structures → **Subtree crossover**
- Exchanges subtree's between selected *parents* that have high fitness → Adaption of **propagation of traits** from generation to generation
- Creates one or two offspring → Recombination of **genetic material**
  - ⇝ Genetic material → **Composition** of **nodes, terminals and edges**
  - ⇝ Genes → Non-terminal or terminal **symbol**
  - ⇝ Sum of the genetic material → genotype → tree(s)

- $(\mu + \lambda)$ strategy $\rightarrow$ Extension of the $(1 + \lambda)$ strategy to recombination
- A set of best μ individuals is formed
  - Each parent is selected uniformly at random
  - λ offspring are bred by sub-tree crossover and mutation
- μ parents + λ offspring then form the new population
  - $\rightsquigarrow$ Level of elitism can be controlled by setting of $\mu \rightarrow$ **selection pressure**

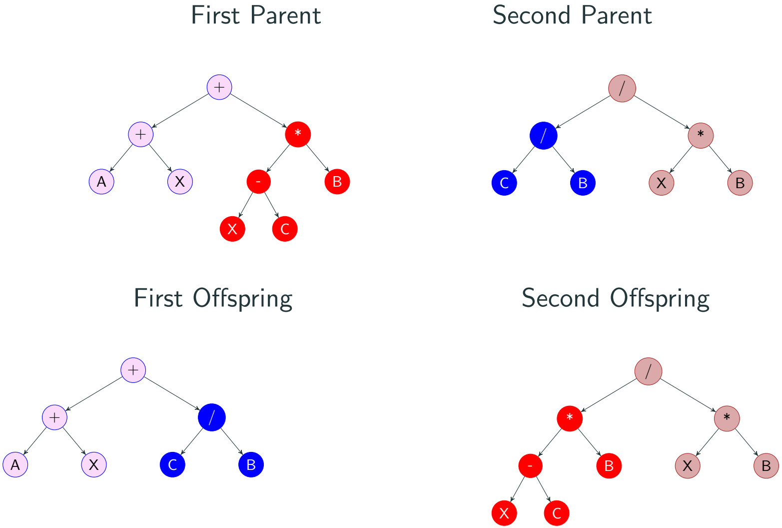


Figure: Subtree crossover

Advanced Methodologies
Evolutionary Algorithms

Algorithm $(\mu + \lambda)$ Evolutionary Strategy (ES)

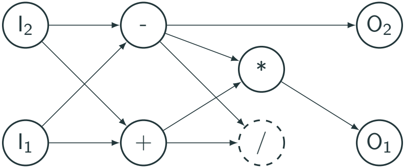
Arguments
μ: Number of parents
λ: Number of offspring
p_c: Recombination probability
p_m: Mutation probability
Return
$\mathcal{B}$: Best individual

initialize($\mathcal{P}$) ▷ *Initialize μ parents*
evaluate($\mathcal{P}$) ▷ *Evaluate the fitness of the parents*
repeat ▷ *Until termination criteria not triggered*
  $\mathcal{Q} \leftarrow$ selection($\mathcal{P}$, μ) ▷ *Select μ parents*
  $\mathcal{Q} \leftarrow$ recombination($\mathcal{P}$, λ, p_c) ▷ *Create λ offspring*
  mutation($\mathcal{Q}$, p_m) ▷ *Mutate the offsprings*
  evaluate($\mathcal{Q}$) ▷ *Evaluate the fitness of the offspring*
  $\mathcal{P} \leftarrow \mathcal{P} + \mathcal{Q}$ ▷ *Form population of next generation*
  $\mathcal{B} \leftarrow$ best($\mathcal{P}$) ▷ *Determine best individual*
until $\mathcal{B}$ *meets termination criterion*
return $\mathcal{B}$ ▷

- Strength of the variation operators is usually controlled with
  - ⇝ Recombination and mutation probabilities
  - ⇝
- Selection pressure can be controlled in various ways
  - ⇝ Elitist vs. non-elitist approaches

Advanced Methodologies
Genetic Programming

- Genetic Programming $\rightarrow$ traditionally tree representation, which is a **well defined form of graph**.
- Graph-based Genetic Programming $\rightarrow$ use of graph representation models that **extend GP beyond trees**.
- In the methods we consider, **Directed Acyclic Graphs** (DAGs), introducing:
  - Reuse of intermediate results.
  - Active and inactive nodes.

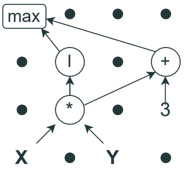


Advanced Methodologies
Genetic Programming

- Application of GP to **evolvable hardware**.
    - Digital circuit design
- Evolution of programs with high degree of **parallelism and distributedness**.
- Discovery of **symbolic**, **neuro-symbolic** and **neural networks**.
- Direct evolution of **machine code**.
- Advantages from the DAG representation features.

- Uses a **direct grid-based** graph representation model
- Each node in the graph is located in a **multi-dimensional** and **evenly spaced** grid
- Prefixed **regular** or **irregular** grid **shape**
- Connections between nodes are limited to be upwards (**feed-forward**)

Grid representation



List representation

```
max     (0,0)   +1  +3

I       (0,1)    0
+       (3,1)   -2   0

*       (1,2)   -1  +1
3       (3,2)

X       (0,3)
Y       (2,3)
```

Cartesian Genetic Programming (CGP)

Representation Model

- Program representation $\rightarrow$ acyclic and directed graph.
- Genotype-phenotype mapping $\rightarrow$ encoding-decoding of the graph
- Predominantly used without recombination $\rightarrow$ mutational $(1+\lambda)$ evolutionary algorithm.

Definition (Cartesian Genetic Program (CP))

A cartesian genetic program $\mathcal{P}$ is an element of $\mathcal{N}_i \times \mathcal{N}_f \times \mathcal{N}_o \times \mathcal{F}$:

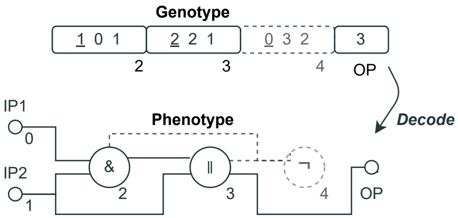
- $\mathcal{N}_i$ is a finite non-empty set of input nodes
- $\mathcal{N}_f$ is a finite set of function nodes
- $\mathcal{N}_o$ is a finite non-empty set of output nodes
- $\mathcal{F}$ is a finite non-empty set of functions

Cartesian Genetic Programming (CGP)

Representation Model

- Nodes of a cartesian genetic program are **continuously indexed**
- Indexing starts with the a value of 0 at the **first input node** and ends at the **last output node**.
- At **each node**, the node number is **increased by one**.
- Let $N = |N_i| + |N_f| + |N_o|$ be the number of nodes of a CP.

Cartesian Genetic Programming
Representation Model



Function Lookup Table

Index	Symbol	Function
0	¬	Negation
1	&	Logical and
2	||	Logical or
3	⊕	Exclusive or

Cartesian Genetic Programming (CGP)
Search Algorithm

- CGP is commonly used with a variant of the $(1 + \lambda)$-EA $\rightarrow$ $(1{+}\lambda)$-CGP
- Implements a modified selection strategy called **neutrality**.
- Adapts a **genetic drift** to provide diversity during the evolutionary run.
- Individuals that have the same fitness are determined, and **one** of these **same-fitness individuals** is returned **uniformly at random**.

Cartesian Genetic Programming (CGP)

Search Algorithm

Algorithm $(1{+}\lambda)$-EA variant used in CGP

initialize($\mathcal{P}$) ▷ *Initialize parent individual*
repeat ▷ *Until termination criteria not triggered*
  $\mathcal{Q} \leftarrow breed(\mathcal{P})$ ▷ *Breed λ offspring by mutation*
  Evaluate($\mathcal{Q}$) ▷ *Evaluate the fitness of the offspring*
  $\mathcal{Q}^{+} \leftarrow best(\mathcal{Q}, \mathcal{P})$ ▷ *Get individuals which have better fitness as the parent*
  $\mathcal{Q}^{=} \leftarrow equal(\mathcal{Q}, \mathcal{P})$ ▷ *Get individuals which have the same fitness as the parent*
  ▷ *If there exist individuals with better fitness*
  if $|\mathcal{Q}^{+}| > 0$ **then**
    ▷ *Choose one individual from $\mathcal{Q}^{+}$ uniformly at random*
    $\mathcal{P} \leftarrow \mathcal{Q}^{+}[\mathrm{r}],\ r \sim U[0,\ |\mathcal{Q}^{+}| - 1]$
    ▷ *Otherwise, if there exist individuals with equal fitness*
  else if $|\mathcal{Q}^{=}| > 0$ **then**
    ▷ *Choose one individual from $\mathcal{Q}^{=}$ uniformly at random*
    $\mathcal{P} \leftarrow \mathcal{Q}^{=}[\mathrm{r}],\ r \sim U[0,\ |\mathcal{Q}^{=}| - 1]$
  end if
until $\mathcal{P}$ *meets termination criterion*
return $\mathcal{P}$ ▷

Cartesian Genetic Programming (CGP)
Mutation

- Standard genetic operator $\rightarrow$ probabilistic **point mutation**.
- Genes are **selected uniformly at random** in the genotype.
- **Exchanges gene values** in the valid range by chance.
- Genetic **variation** of **functionality** and **connectivity**.

Cartesian Genetic Programming (CGP)

Mutation

Algorithm Probabilistic point mutation

Input: Genome $\mathcal{G}$, Function set $\mathcal{F}$, Number of function nodes $\mathcal{N}_f$, Number of input nodes $\mathcal{N}_i$, Mutation rate $\mathcal{P}$
Output: Mutated Genome $\widetilde{\mathcal{G}}$

foreach $g \in \mathcal{G}$ **do** ▷ *Iterate over the genome*
| $X \sim \mathrm{Ber}(\mathcal{P})$ ▷ *Bernoulli random variable*
| **if** $X = 1$ **then** ▷ *Control the mutation strength with X*
| | **if** g is a connection gene **then**
| | | ▷ *Determine the node number of the gene*
| | | $n \leftarrow \mathrm{NodeNumber}(g)$
| | | ▷ *Select g in the range of previous node indexes by chance*
| | | $g \leftarrow r,\ r \sim U[0,\ n-1]$
| | **else if** g is a function gene **then**
| | | ▷ *Select g in the range of the function indexes by chance*
| | | $g \leftarrow r,\ r \sim U[0,\ |F|-1]$
| | **else** ▷ *g is a output gene*
| | | ▷ *Select g in the range of the function and input nodes by chance*
| | | $g \leftarrow r,\ r \sim U[0,\ |N_f| + |N_i| - 1]$
| | **end if**
| **end if**
end foreach
return $\mathcal{G}$ ▷ *Return the mutated genome*

Cartesian Genetic Programming (CGP)
Mutation