Master of Technology

Computational Intelligence II

Other Evolutionary Computation Techniques

Dr. Zhu Fangming
Institute of Systems Science,
National University of Singapore
Email: isszfm@nus.edu.sg

© 2018 NUS. The contents contained in this document may not be reproduced in any form or by any means, without the written permission of ISS, NUS other than for the purpose for which it has been supplied.

© 2018 NUS. All Rights Reserved.

Objectives

- Upon the completion of this lecture, the students will be able
  - To explain the basic operation and characteristics of the other evolutionary computation techniques
    - Evolutionary programming
    - Evolution strategies
    - Genetic programming
    - Particle swarm optimization

© 2018 NUS. All Rights Reserved.

Topics

- Evolutionary Programming
- Evolution Strategies
- Genetic Programming
- Particle Swarm Optimization (PSO)

© 2018 NUS. All Rights Reserved.

Evolutionary Programming

- Evolutionary programming (EP) is a stochastic optimization strategy similar to genetic algorithms where mutation is the sole genetic operator.
- EP operates on a variety of representational structures, frequently real-valued objective variables although more complex structures have been used.
- The representation of the solutions are abstracted as vectors of behavioral traits.

© 2018 NUS. All Rights Reserved.

Evolutionary Programming

- This also means that there is no need for a decoding function.
- The significance of this is that a decoding function may introduce additional non-linearities and other mathematical difficulties, which can hinder the search process substantially.

© 2018 NUS. All Rights Reserved.

Evolutionary Programming

- The basic EP method involves
  - Choose an initial population of trial solutions at random.
  - Each solution is mutated to produce one or more solutions. The exact form of the mutation operator is dependent on the representation, but the degree to which a solution is mutated is related to the solution's fitness.
  - Each offspring solution is assessed by computing its fitness. A stochastic tournament is held to determine N solutions to be retained for the population of solutions. There is no requirement that the population size be held constant, nor that only a single offspring be generated from each parent.

© 2018 NUS. All Rights Reserved.

Differences from GA

- Representation
  - GA uses fixed length strings (chromosomes)
  - EP uses the representation of the problem and is not limited by length or form

- Genetic operators
  - GA uses both crossover and mutation
  - EP uses only mutation and is effected based on statistical distribution. The magnitude of the mutation is reduced as search progresses i.e. optimum is reached.

© 2018 NUS. All Rights Reserved.

Evolution Strategies (ES)

- Conceived in 1960s for solving engineering optimization problems like designing optimal airfoils i.e. to optimize real-valued parameters of devices.
- The problem is defined as finding the fixed-length real-valued vector **x** that is associated with the optimizing function F(**x**).

© 2018 NUS. All Rights Reserved.

ES technical summary

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	(μ,λ) or $(\mu+\lambda)$
Specialty	Self-adaptation of mutation step sizes

© 2018 NUS. All Rights Reserved.

Evolution Strategies

- An initial population of parent vectors, $\mathbf{x}_i$, i=1,2,...,P is selected at random from a feasible range in each dimension.
- Offspring are created through mutation and recombination.
- Survivor selection then determines which of these vectors to maintain based on their fitness i.e. entirely deterministic. The P vectors that possess the least error become the new parents for the next generation.
- The process continues until a sufficient solution is reached.

© 2018 NUS. All Rights Reserved.

Mutation in ES

- An offspring vector xi' is created from each parent xi by adding a Gaussian random variable with zero mean and preselected standard deviation to each component of xi.

$$x'i = xi + N(0,\sigma)$$

- Mutation step size σ can coevolve with the solution x as well.

© 2018 NUS. All Rights Reserved.

Recombination in ES

- Creates one offspring
- Acts per variable / position by either
  - Averaging parental values (intermediary), or
  - Selecting one of the parental values (discrete)
- From two or more parents by either:
  - Using two selected parents to make an offspring (local)
  - Selecting two parents for each position anew (global)

© 2018 NUS. All Rights Reserved.

Survivor Selection in ES

- Applied after creating λ offspring from the μ parents by mutation and recombination
- Deterministically chops off the “bad stuff”
- Basis of selection is either:
  - The set of offspring only: (μ,λ)-selection
  - The set of parents and offspring: $(\mu+\lambda)$-selection

© 2018 NUS. All Rights Reserved.

Survivor Selection in ES

- $(\mu+\lambda)$-selection
  - Parent population of size μ.
  - Generate λ offspring from randomly chosen parents.
  - Next population is μ best among **parents and offspring**.
- (μ,λ)-selection (where $\lambda > \mu$)
  - Parent population of size μ.
  - Generate λ offspring from randomly chosen parents.
  - Next population is μ best among **offspring**.

© 2018 NUS. All Rights Reserved.

Genetic Programming

- Genetic programming (GP) is an extension of the conventional genetic algorithm in which each individual in the population is a computer program.
- GP is an attempt to deal with one of the central questions in computer science, namely
  - How can computers learn to solve problems without being explicitly programmed?
  - OR, in other words
  - How can computers be made to do what needs to be done without being told exactly how to do it?

© 2018 NUS. All Rights Reserved.

Genetic Programming

- All computer programs can be viewed as a sequence of applications of functions (operations) to arguments (values).
- GP is most easily understood if one thinks about it in terms of a programming language that views a computer program as a sequence of applications of functions to arguments.

© 2018 NUS. All Rights Reserved.

Genetic Programming

- GP initially creates computer programs at random and then manipulates the programs by various genetically motivated operations till an acceptable program is found.
- GP may be implemented in a conceptually straightforward way in a programming language that permits a computer program to be easily manipulated as data and then permits the new created data to be immediately executed as a program.
  - LISP programming language
  - Scheme

© 2018 NUS. All Rights Reserved.

Applying Genetic Programming

- Steps to be taken in applying GP include
  - Specifying the representation scheme
    1. Identify the set of terminals
       - The terminals can be viewed as the inputs to the as-yet-undiscovered computer program
       - The terminals are the ingredients from which GP attempts to construct a computer program to solve, or approximately solve, the problem.
       - Typical terminals: $t \in \mathbf{T}$

         Constants: integers, real numbers, etc
         Variables: x, y, z (state variables of system)

© 2018 NUS. All Rights Reserved.

Applying Genetic Programming

2. Identify the set of functions that are to be used to generate the mathematical expression that attempts to fit the given finite sample of data.
   - Typical functions: f ∈ **F**

     Arithmetic: +, -, *, /,
     Mathematical: sin, cos, tan, exp,...
     Logical: And, Or, Not, If-then-else, ..
     Iterative: do-until, for...
     Recursion: functions permitting recursion
     Special: domain specific functions
3. Each computer program is a composition of functions from the function set **F** and terminals from the terminal set **T**.

© 2018 NUS. All Rights Reserved.

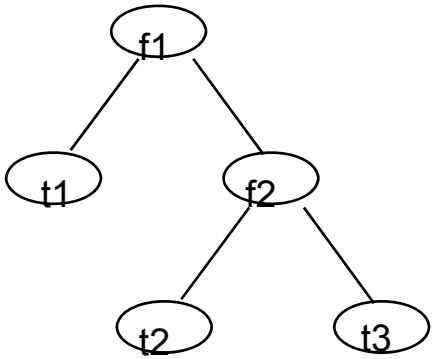
Applying Genetic Programming

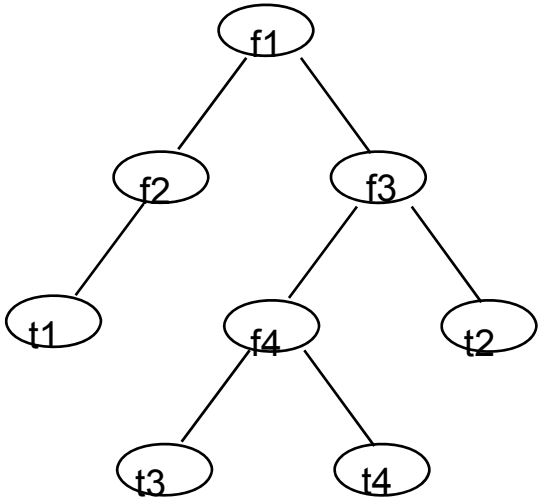
- Steps in applying genetic programming
  - Fitness measure
    - Each individual computer program in the population is measured in terms of how well it performs in the particular problem environment.
  - Parameters for controlling the run
    - Crossover operation is implemented by exchanging randomly selected subtrees in the individuals
    - Various mutation operations are also used to increaser population diversity.
  - Method for designating a result and the criterion for terminating a run
    - The best individual appearing in any generation of a run is typically designated as the result product by the run of GP.

© 2018 NUS. All Rights Reserved.

Problem Encodings

- Structures
  - The structures undergoing adaptation in GP are active.
    - Structures are inherently hierarchical.
  - They are not passive encodings, a fixed-length character string as in GA.
  - They are interpreted as programs.



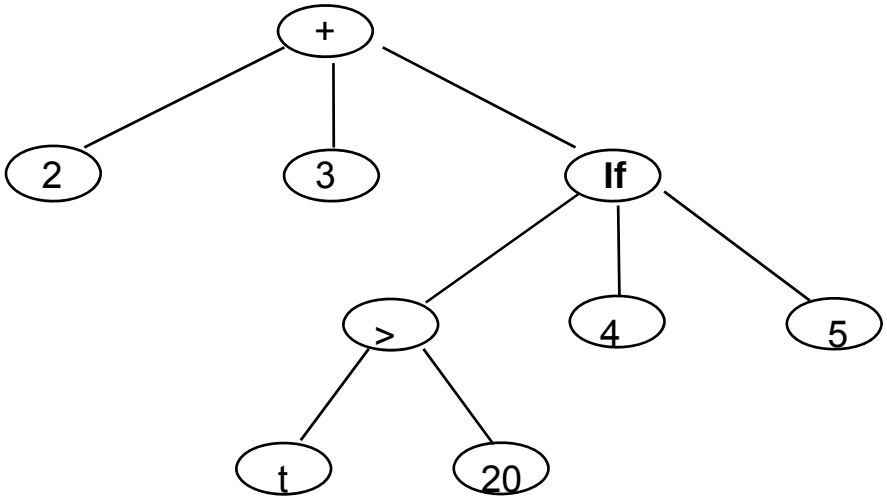


© 2018 NUS. All Rights Reserved.

Problem Encodings

- Tree encoding
  - Trees are open-ended
  - Trees can grow large in uncontrolled ways
  - Trees can be very difficult to understand and to simplify

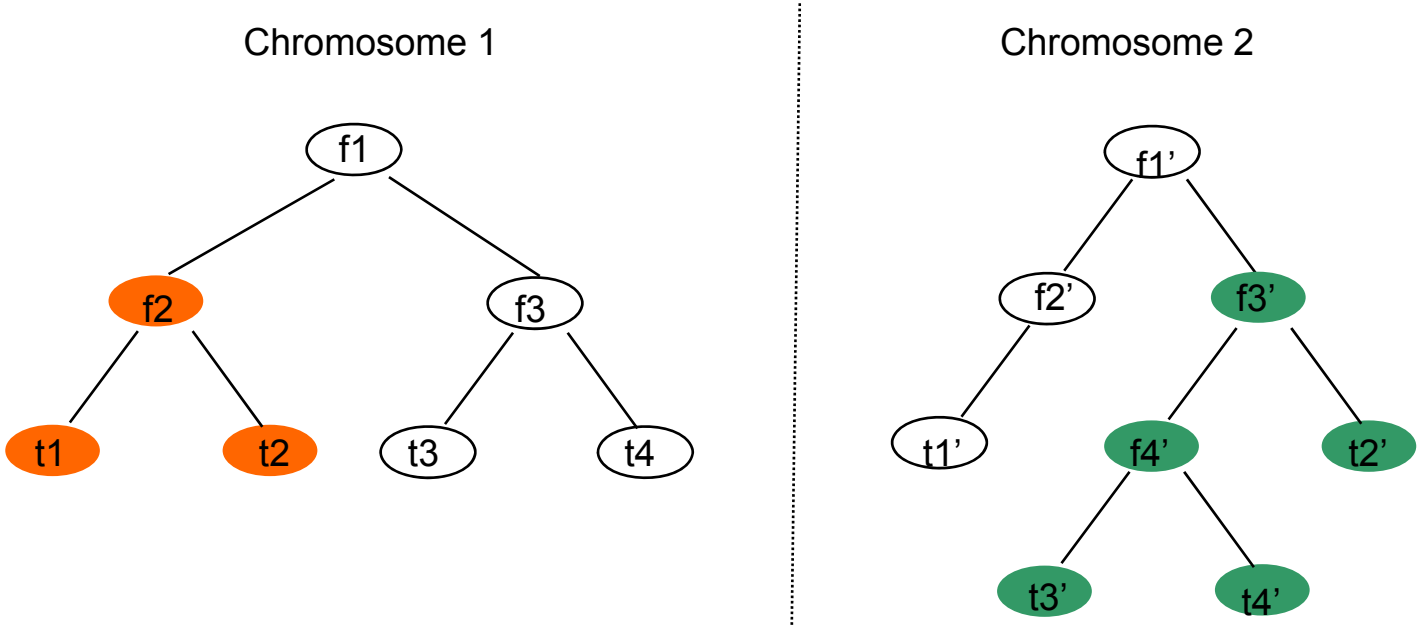
Example: (+ 2 3 (**If** (> t 20) 4 5)) which states
"If t>20, returns 9, otherwise returns 10".



© 2018 NUS. All Rights Reserved.

Crossover Operators

- Sub-trees of two parents are exchanged to produce two offsprings



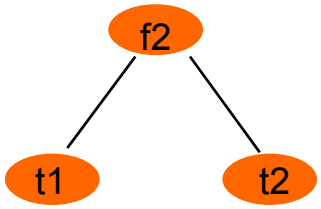
Two parental computer programs

© 2018 NUS. All Rights Reserved.

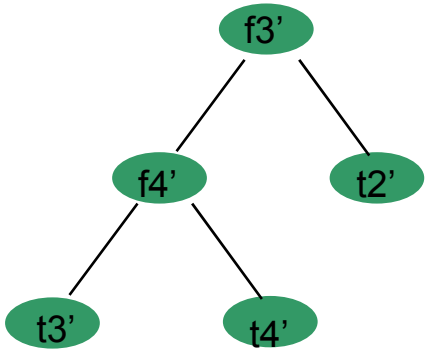
Crossover Operators

Two crossover fragments

Fragment from chromosome 1

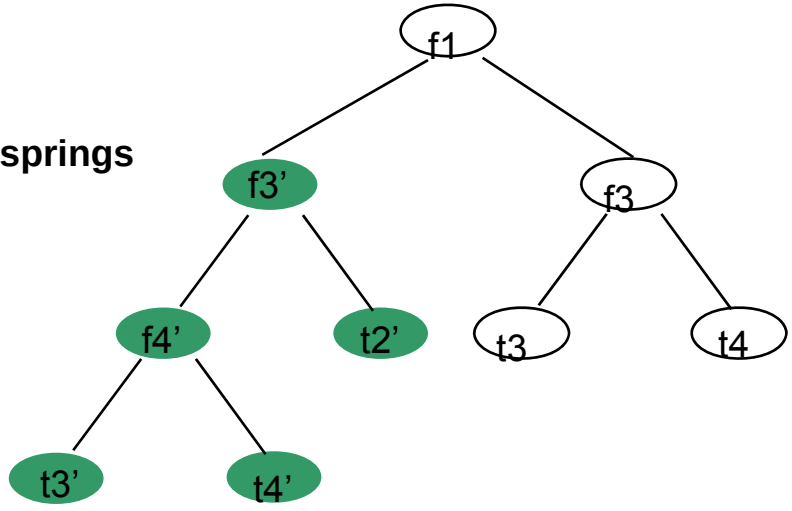


Fragment from chromosome 2

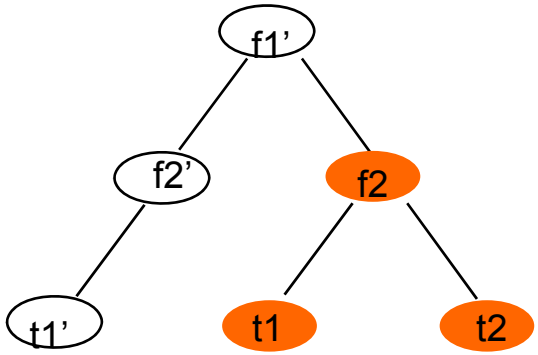


Offsprings

Chromosome 1' after crossover



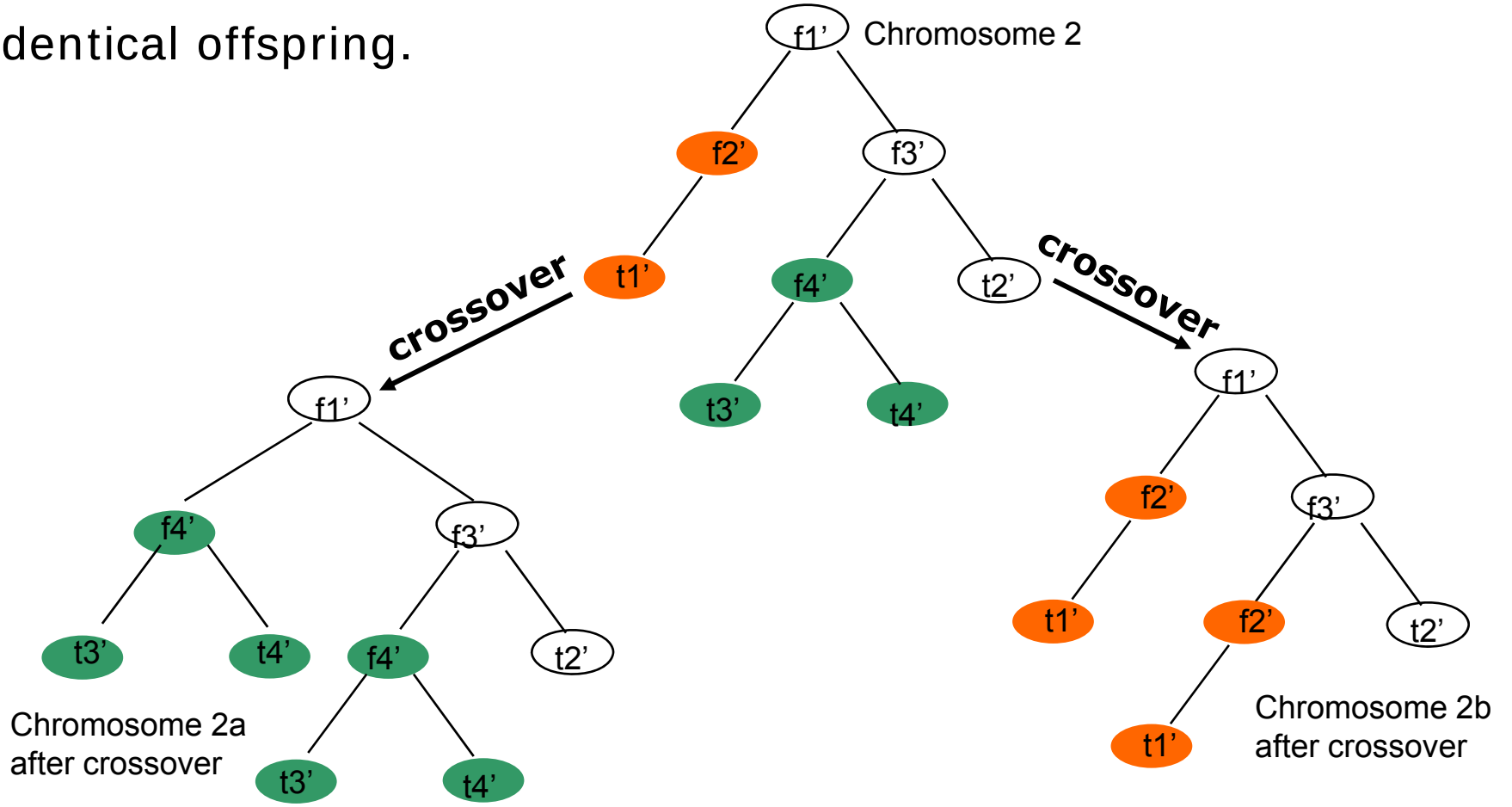
Chromosome 2' after crossover



© 2018 NUS. All Rights Reserved.

Crossover Operators

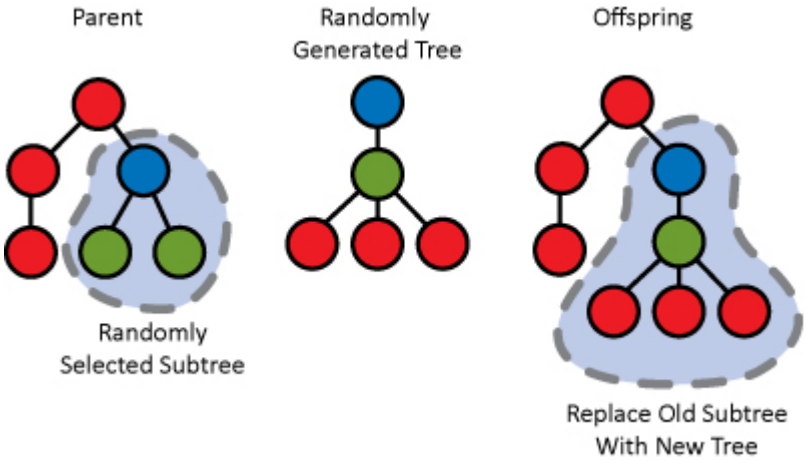
In genetic programming identical parents can yield different offspring, while in genetic algorithms identical parents would yield identical offspring.



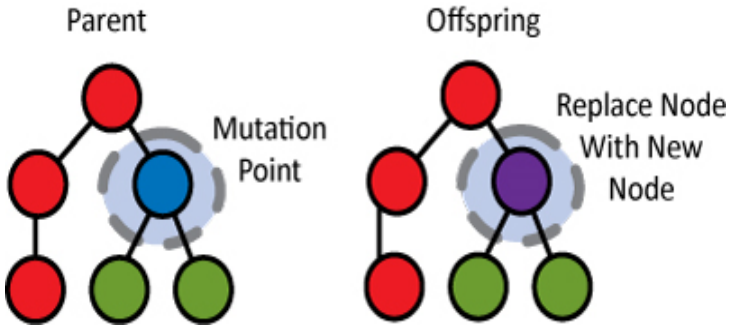
© 2018 NUS. All Rights Reserved.

Mutation Operators

Subtree Mutation:



Node Replacement Mutation:

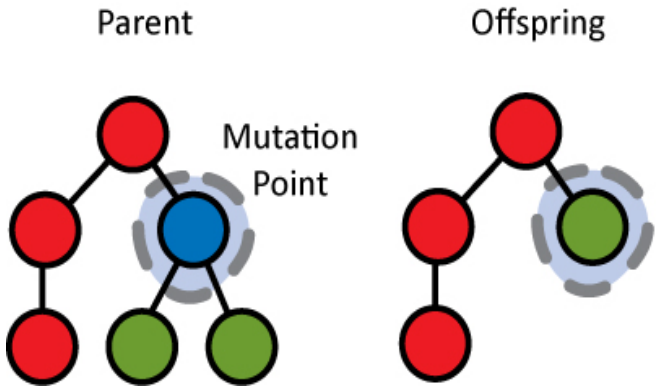


Source: http://geneticprogramming.us/Genetic_Operations.html

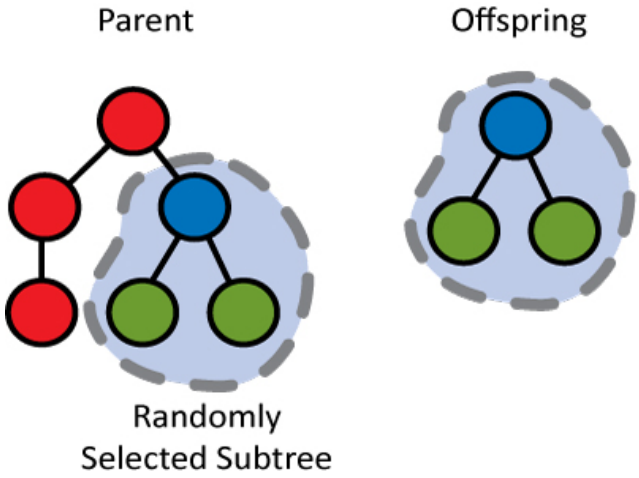
© 2018 NUS. All Rights Reserved.

Mutation Operators

Shrink Mutation:



Hoist Mutation:



Source: http://geneticprogramming.us/Genetic_Operations.html

© 2018 NUS. All Rights Reserved.

Information on Genetic Programming

- LISP programming language is especially well-suited for GP, however it should be recognized that GP does not require LISP for its implementation and is not in any way based on LISP.
- GP has succeeded in evolving correct programs to solve a large number of simple (and some not-so-simple) problems in regression, optimal control, planning, image compression, robotics, etc.
- GP demo
  - http://alphard.ethz.ch/gerber/approx/default.html

© 2018 NUS. All Rights Reserved.

Particle Swarm Optimization (PSO)

- Particle Swarm Optimization (PSO) was developed by Kennedy & Eberhart in 1995.
- It was inspired from the nature social behavior and dynamic movements with communications of insects, birds and fish
- It is related to evolution-inspired problem solving techniques such as genetic algorithms.
- Swarm intelligence-based optimization
- Population-based optimization

© 2018 NUS. All Rights Reserved.

Particle Swarm Optimization (PSO)

- Swarm : a set of particles
- Particle: a potential solution
  - Position: X_i
  - Velocity: V_i
- Each particle maintains
  - Individual best position: p_i; $pbest_i = f(p_i)$
- Swarm maintains its global best:

  p_g; $gbest = f(p_g)$

© 2018 NUS. All Rights Reserved.

PSO Algorithm

- Basic PSO algorithm:
  1. Initialize the swarm from the solution space
  2. Evaluate fitness of each particle
  3. Update individual and global bests
  4. Update velocity and position of each particle
  5. Go to step 2, and repeat until termination condition

© 2018 NUS. All Rights Reserved.

PSO Algorithm (cont.)

- Original velocity update equation:

$$V_i^{t+1} = V_i^t + \varphi_1 . r_1 (P_i - X_i^t) + \varphi_2 . r_2 (P_g - X_i^t)$$

Inertia **Personal Inference** **Social Inference**

  - $r_1, r_2 \sim U(0,1)$: random number
  - φ_1, φ_2 : acceleration constant

- Position Update:

$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

© 2018 NUS. All Rights Reserved.

PSO Algorithm (cont.)

- Number of particles usually between 10 and 50
- φ_1 is the importance of personal best value
- φ_2 is the importance of global best value
- Usually $\varphi_1 + \varphi_2 = 4$ (empirically chosen value)
- If velocity is too low → algorithm too slow
- If velocity is too high → algorithm too unstable

© 2018 NUS. All Rights Reserved.

PSO: Comparison with GA

- Both are initialized with a random population/ agents, and a search is made for the optimum by updating generations. Both use the idea of fitness evaluation.
- Both update the population and search for the optimium with random techniques. Both do not guarantee success.
- PSO does not use evolutionary operators like crossover and mutation. There is no selection in PSO. In PSO, the particles fly through the space by following current optimum particles.
- The information sharing mechanism in PSO is significantly different from GA.

© 2018 NUS. All Rights Reserved.