Topics for this Lecture

- Automatic Test Generation Approaches
  - Random Testing
  - Evolutionary Search √√
  - Constraint-Based Testing

Search Based Software Testing (SBST)

- The problem of test data generation is **converted** to a search problem, and search algorithms such as Hill Climbing (**HC)** or Genetic Algorithm (**GA)** are used to derive test data.

- They use a Fitness Function (**FF**) that guides the search toward better solutions.

- The **effectiveness** of the search algorithms is improved as long as the fitness function **distinguishes** between **better** and **worse** solutions.

- The traditional approach is targeting each branch separately.

Search Based Software Testing (SBST)

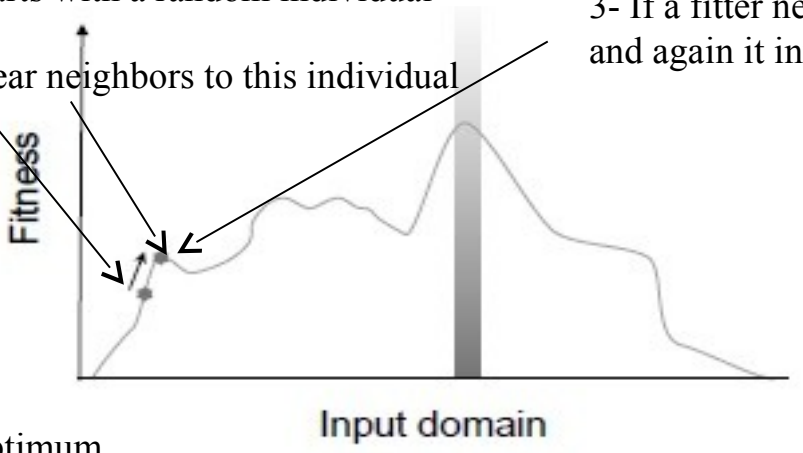
Local Search: Hill Climbing (HC)

- **local search algorithms aim to improve one individual by exploring its neighbors.**

1- Hill Climbing (HC) usually starts with a random individual

2- Then it considers the set of near neighbors to this individual

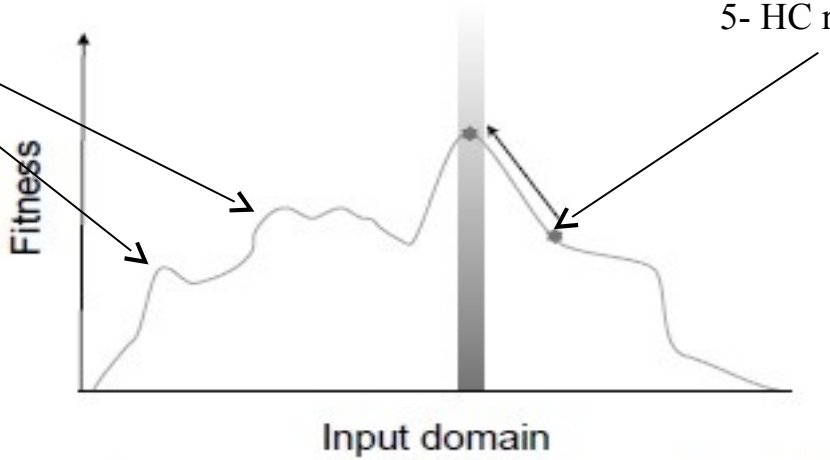
3- If a fitter neighbor is found, HC moves to it and again it investigates its neighbors



(a) Climbing to a local optimum

4- If HC gets trapped in a local optimum, which there is no better neighbor is found

5- HC randomly restarts from a new individual.



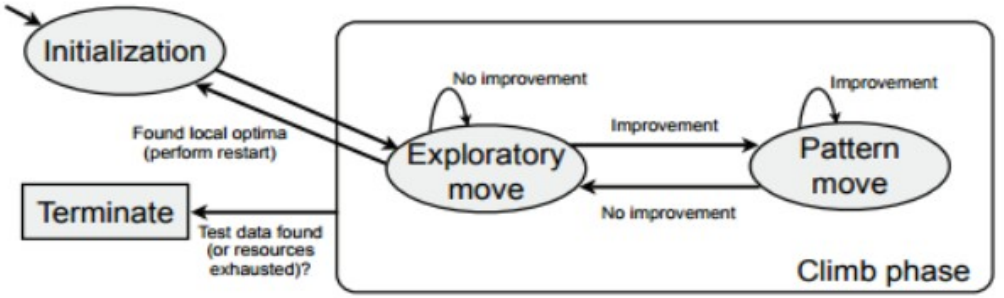
(b) Restarting, on this occasion resulting in a climb to the global optimum

P. McMinn, "Search-based software testing: Past, present and future," 2011

Oregon State University

Local Search: Alternating Variable Method (AVM)

- The **Alternating Variable Method** (**AVM**) is a similar technique to HC
- AVM tries to optimize each input variable in isolation
- The chosen variable is randomly modified by increasing or decreasing a small amount, which is called an **exploratory move.**
- If the changes affect the fitness function, AVM applies a large amount in the same direction, which is called **pattern moves**
- The pattern search is applied in the same direction as long as the fitness function is improved
- Once there are no further improvements of the input variable, the search moves to consider another variable, repeating the same process, until the branch is covered or no more variables can be improved.



- **For example**, in the case of an integer an **exploratory move** consists of adding of **+1 or −1**.

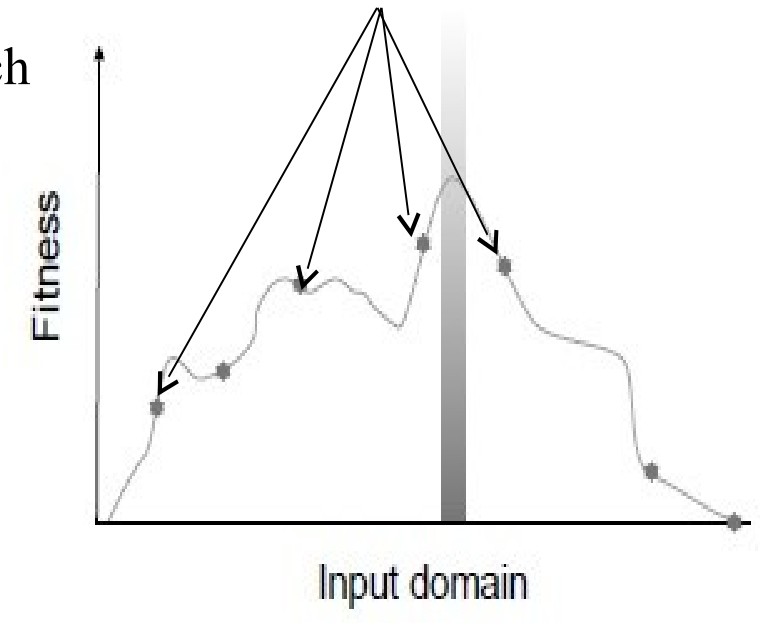
If the **exploratory move** was successful (i.e., the fitness improved), then the search accelerates movement in the direction of improvement with so-called "**pattern moves**". **For example**, in the case of an integer, the search would next try **+2, then +4, etc**.

McMinn, Phil, et al. "Input Domain Reduction through Irrelevant Variable Removal and its Effect on Local, Global and Hybrid Search-Based Structural Test Data Generation."

Oregon State UNIVERSITY

Global Search

- **Global** search, sampling **many points** in the search space at once
- **Global** search is more effective than **local** search, but less efficient, as it is more costly.
- The most commonly applied global search algorithms is a ***Genetic Algorithm*** (GA).
- A GA tries to imitate the natural processes of evolution
- **Evolution** is change in the heritable characteristics of biological populations over successive generations

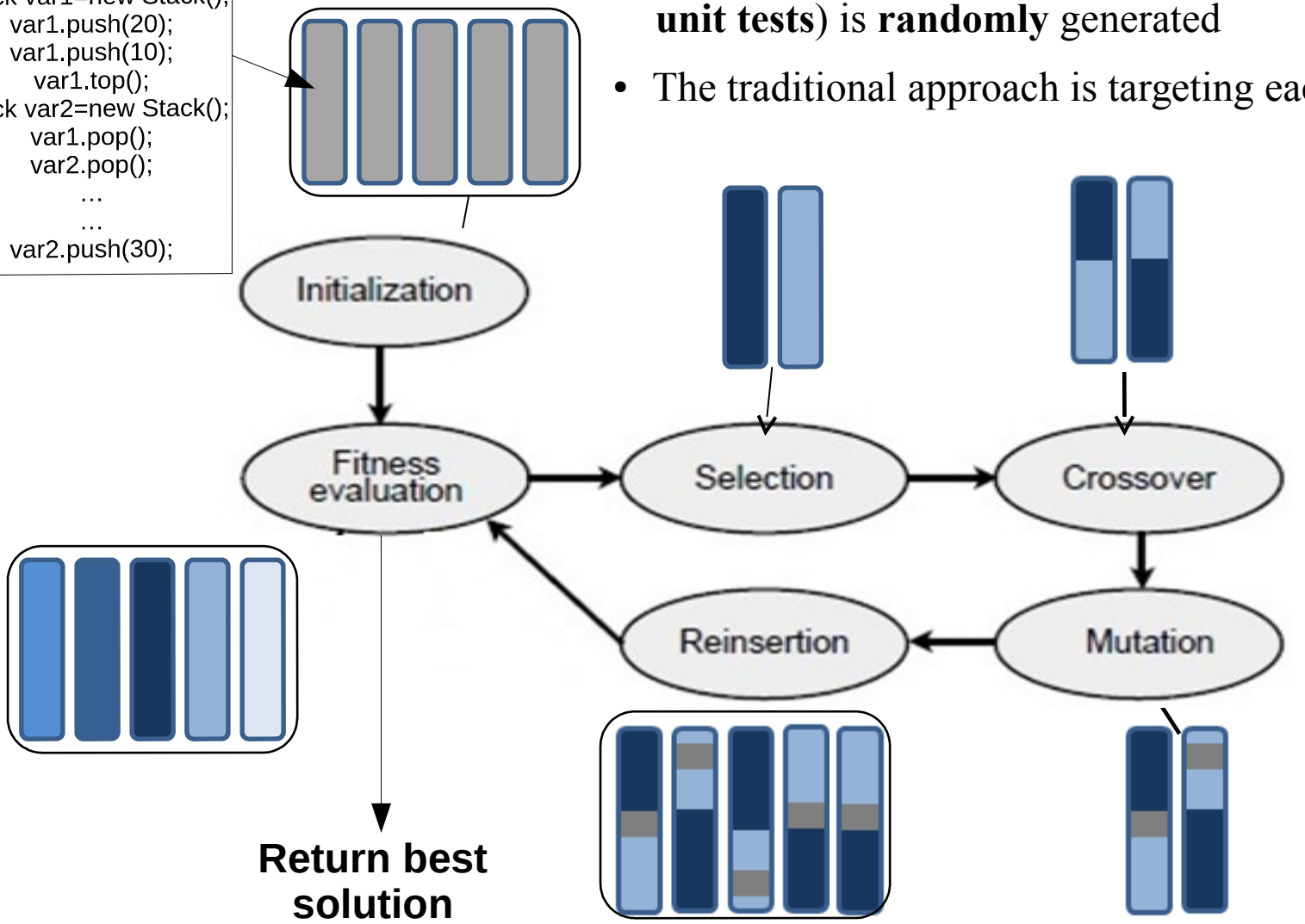


P. McMinn, "Search-based software testing: Past, present and future," 2011

Global Search (Genetic Algorithm (GA))

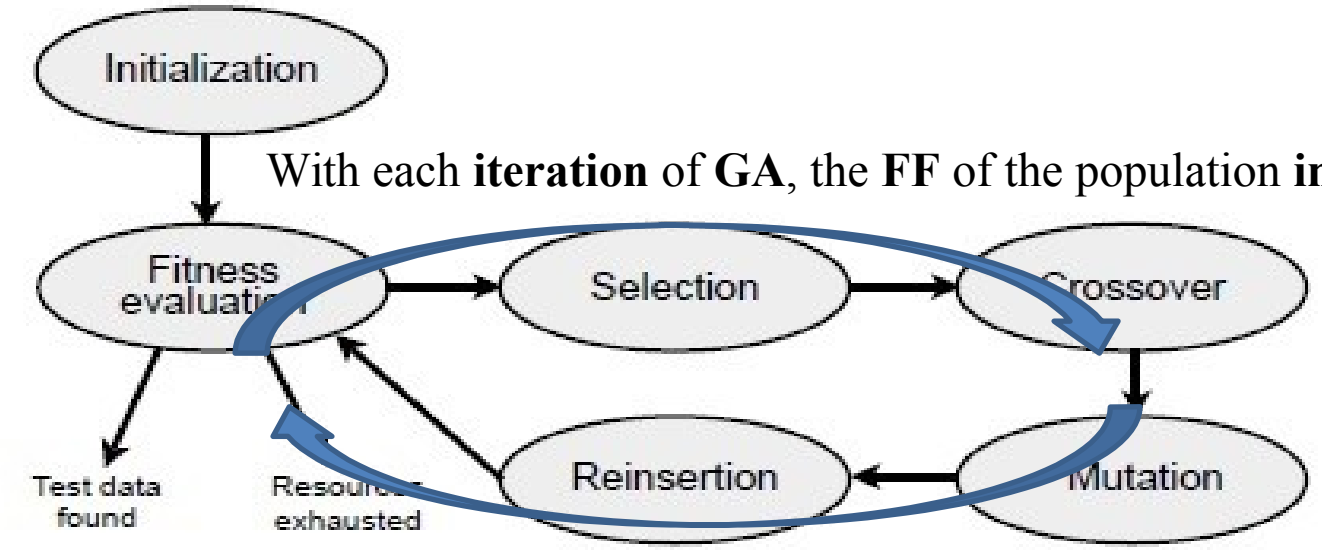
```
Stack var1=new Stack();
var1.push(20);
var1.push(10);
var1.top();
Stack var2=new Stack();
var1.pop();
var2.pop();
…
…
var2.push(30);
```

- **Initial** population (**individuals/chromosomes (i.e., unit tests)** is **randomly** generated
- The traditional approach is targeting each branch separately.



Oregon State University

Global Search (Genetic Algorithm (GA))



This **process** can be **iterated** until either an **optimal solution** is found, or a stopping condition (e.g. **timeout**) holds.

P. McMinn, "Search-based software testing: Past, present and future," 2011

Oregon State University

Initial Population

- **Initial Population**
  - Randomly chosen

- **Individual Representation**: An **individual** (a *Test case*) is a sequence of **constructors** and **method calls**.

Constructor Statement: represents a constructor call

```
Stack var1 = new Stack();
var1.push(−20);
var1.push(  20);
```

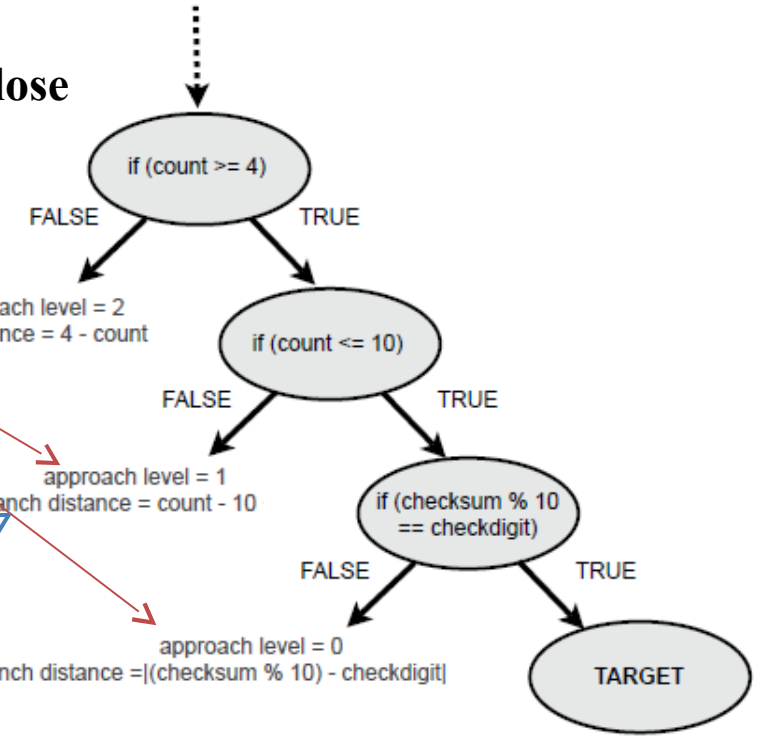
Method Statement: represents a public method call

Fitness Function

- **Fitness Function**: it **estimates** how **close** a candidate solution (**individual**) is to **satisfy** a **coverage goal**. A common FF integrates Approach level and Branch Distance

Approach level:
rewards test cases whose **execution** get **close** in the **CFG** to the **target branch**

```
if (count >= 4)
  if (count <= 10)
    if (checksum % 10 == checkdigit)
      return 0; //target is the true branch
    else return 1;
  else return 2;
else return 3;
```



Branch Distance:
how **near** the input was to **executing** the **required** branch

P. McMinn, "Search-based software testing: Past, present and future," 2011

Fitness Function

- **Fitness Function**: it **estimates** how **close** a candidate solution (**individual**) is to **satisfy** a **coverage goal**. A common FF integrates Approach level and Branch Distance

$$f(t,x) = \mathcal{A}(t,x) + \nu(d(t,x_c)) \ ,$$

Approach level:
rewards test cases whose **execution** get **close** in the **CFG** to the **target branch**

Branch Distance:
how **near** the input was to **executing** the **required** branch

```
public static void foo(int z){
  if(z > 0)
    if(z > 100)
      if(z > 200)
        ; //target
}
```

$$z = 50$$

$$f(t_{50},x_{z>200}) = 1 + \nu(|50 - 100| + 1) = 1 + \nu(51)$$

worse than a test case having $z = 101$

$$f(t_{101},x_{z>200}) = 0 + \nu(|101 - 200| + 1) = 0 + \nu(100)$$

Arcuri, Andrea, and Gordon Fraser. "On the effectiveness of whole test suite generation."

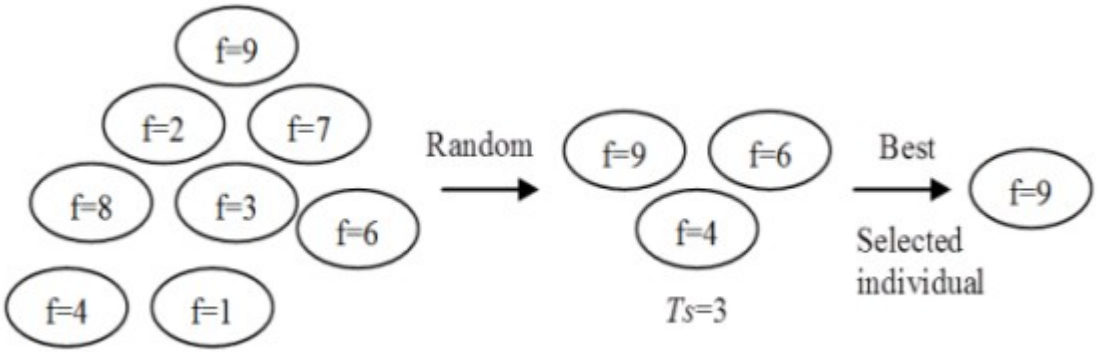
Oregon State UNIVERSITY

SELECTION STRATEGY FOR REPRODUCTION

- The selection strategy addresses on which of the individuals (test cases) in the current generation will be used to reproduce offspring in hopes that next generation will have even higher fitness

A)Tournament Selection

- N individuals are selected randomly from the larger population.
- The selected individuals compete against each other.
- The individual with the highest fitness wins and will be included as one of the next generation population.



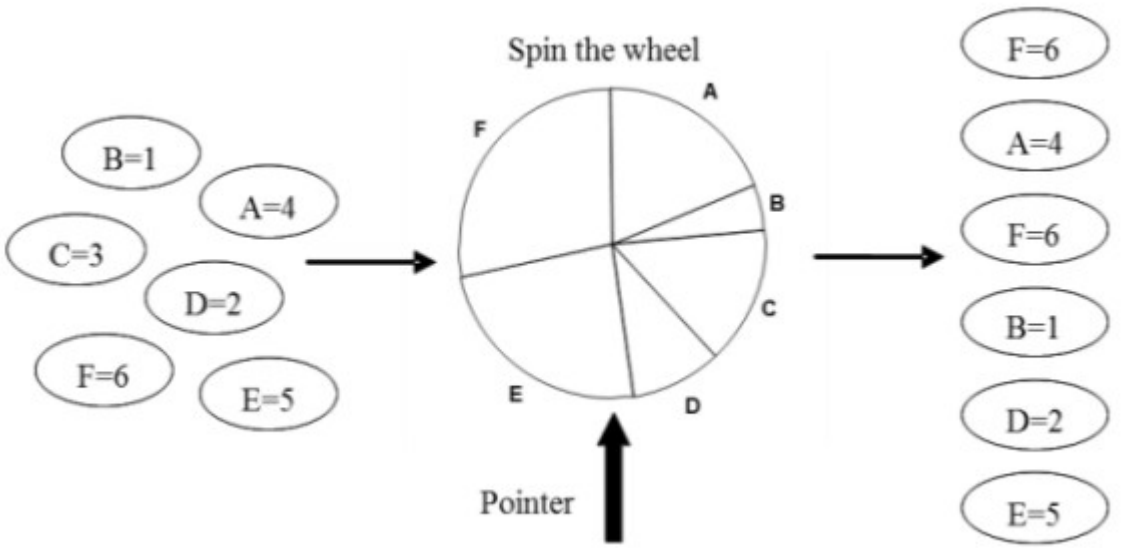
Noraini, Mohd Razali, and John Geraghty. "Genetic algorithm performance with different selection strategies in solving TSP." (2011).

Oregon State University

Proportional Roulette Wheel Selection

- Individuals (i.e., test cases) are selected with a probability that is directly proportional to their fitness values i.e. an individual‟s selection corresponds to a portion of a roulette wheel.
- The fittest individual occupies the largest segment, whereas the least fit have correspondingly smaller segment within the roulette wheel.
- Let $f_1, f_2, \ldots, f_n$ be fitness values of individual 1, 2,…, n. Then the selection probability, P_i for individual i is define as,

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$



Noraini, Mohd Razali, and John Geraghty. "Genetic algorithm performance with different selection strategies in solving TSP." (2011)

Rank-based Roulette Wheel Selection

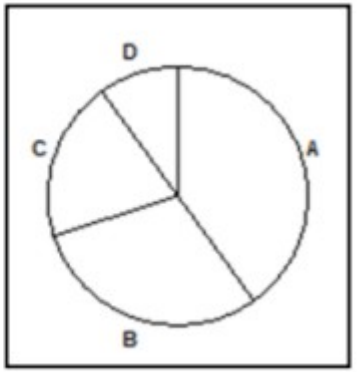
- Rank-based selection schemes first sort individuals in the population according to their fitness and then computes selection probabilities according to their ranks rather than fitness values.

$$Rank(Pos) = 2 - SP + \left(2.(SP-1).\frac{(Pos-1)}{(n-1)} \right)$$

Where Selective pressure 1.0<= **SP**<=2.0

Individual fitness value	Rank	Scaled rank with SP=2.0	Scaled rank with SP=1.1
1	1	0	0.9
3	2	0.2	0.92
4	3	0.4	0.94
7	4	0.6	0.96
8	5	0.8	0.98
9	6	1	1
10	7	1.2	1.02
15	8	1.4	1.04
20	9	1.6	1.06
30	10	1.8	1.08
95	11	2	1.1

Roulette Wheel Selection



Rank-based Selection

Don't use roulette wheel selection!

Oregon State University

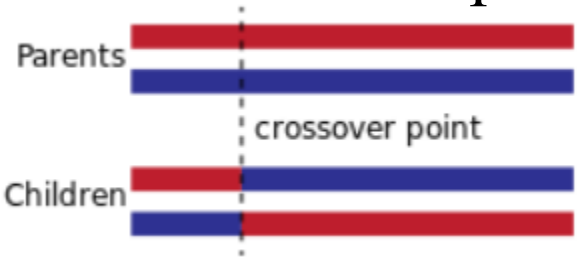
Problems with Selection

- Selective pressure: The higher, the more likely the fittest are chosen
- Stagnation: Selective pressure too small
- Premature convergence: Selective pressure too high
- A good search technique must find a good trade-off between **exploration** and **exploitation** in order to find a global optimum
  - ***Exploration***: Poor solutions must have chance to go to the next generation.
  - ***Exploitation***: Good solutions go to the next generation more frequently than poor solutions.

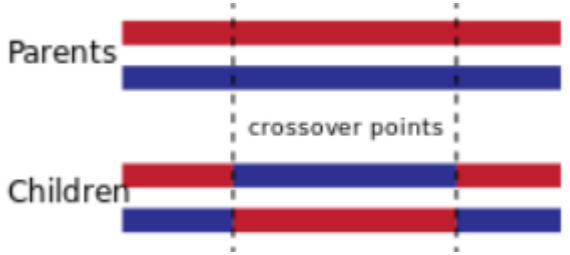
Crossover

- Crossover, it is used to generate new offspring (test case) by combining between them the individuals of the parents.

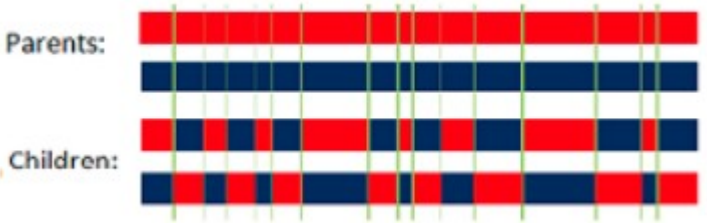
- Single point crossover: Choose one point in parents and split/merge at that point



- Two point crossover: Choose two points in parents and exchange middle parts



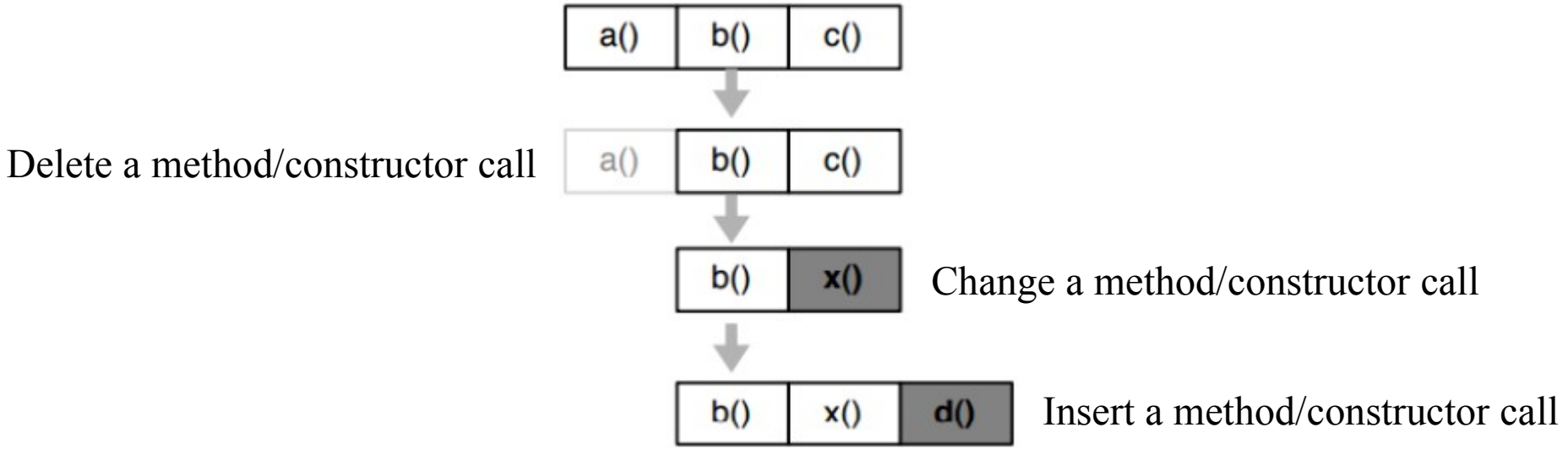
- Uniform crossover: Genes are randomly chosen from either parent individual



Oregon State University

Mutation

- The mutation operator is applied to make small changes in the individuals (i.e., test cases) of the offspring
- Typically, each method/constructor call is considered for mutation with probability 1/k, with k the length of the bit-string



Parameter tuning

- Search algorithms usually depend on several parameters
  - Population size
  - type of selection mechanism
  - crossover rate
  - probability of mutation

- The choice of all these parameters might have a large impact on the performance of a search algorithm.

Some automatic SBST input generation tools for Java

- EvoSuite tool:(http://www.evosuite.org/)
  - Uses one individual as an entire test suite
  - Whole test suites are evolved with the aim of covering all coverage goals at the same time

- AVMFramework (AVMf) tool: (http://avmframework.org/)
  - AVMf is a framework and Java implementation of the Alternating Variable Method (AVM)
  - It is freely available for use

References:

Noraini, Mohd Razali, and John Geraghty. "Genetic algorithm performance with different selection strategies in solving TSP." (2011).
https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)
https://www.st.cs.uni-saarland.de/edu/testingdebugging10/slides/11-SearchBasedTesting