Applying Genetic Programming to Bytecode and Assembly

Eric Collom

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

29 April '14,
UMM Senior Seminar

The big picture

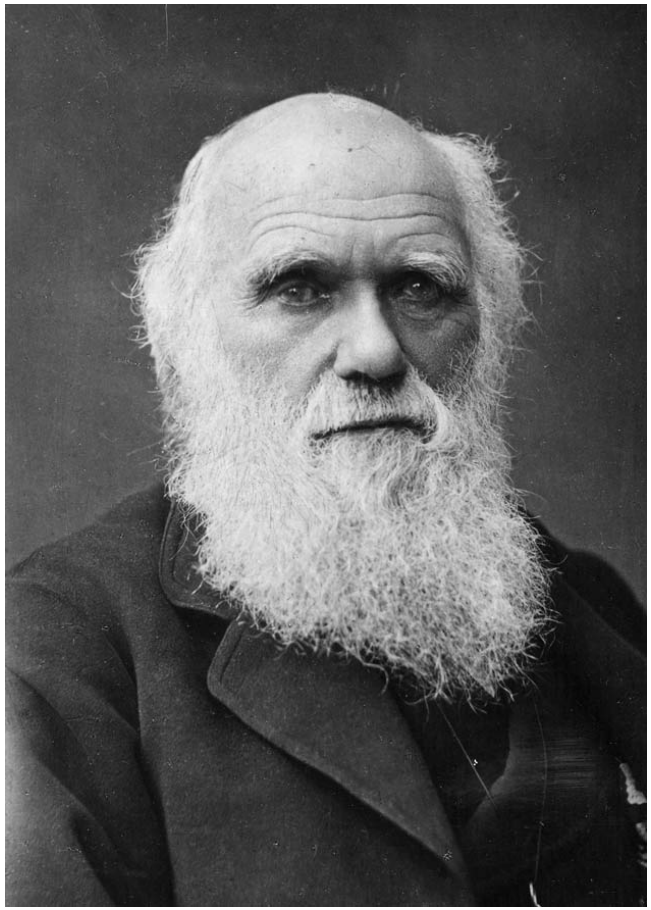
- Evolving whole programs is hard to do with source code.
- Evolving whole programs with bytecode and assembly is not as hard.

Outline

Outline

What is Evolutionary Computation?

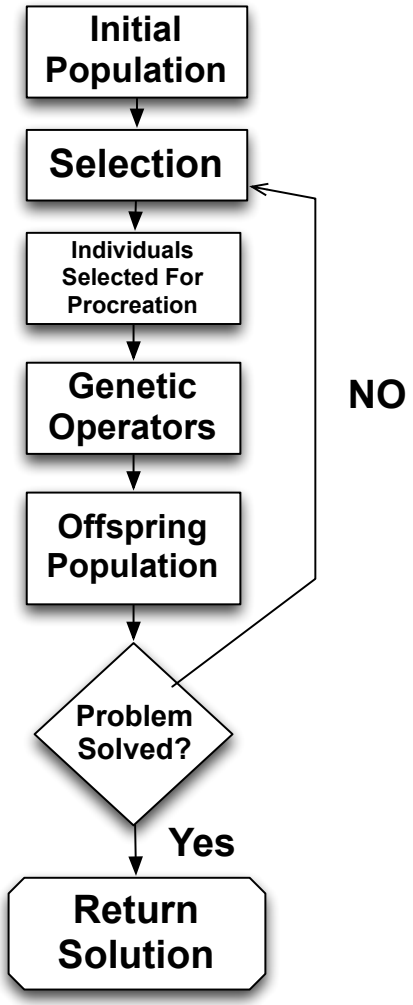
- EC is a a technique that is used to automate computer problem solving.
- Loosely emulates evolutionary biology.



Charles Darwin
http://tinyurl.com/lqwj3wt

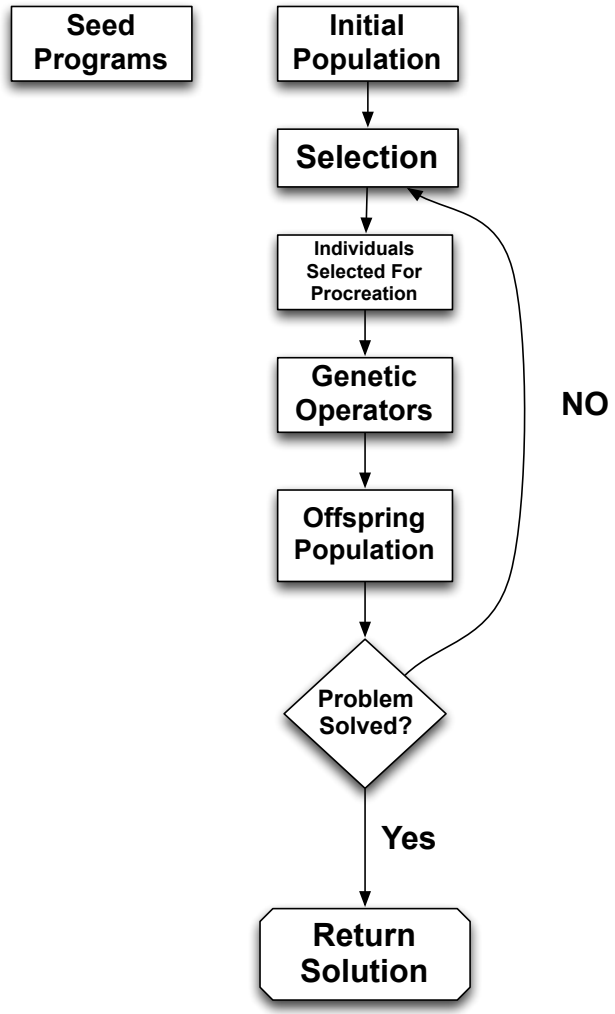
How does it work

- Continuous Optimization
- Selection is driven by the *fitness* of individuals
- Genetic Operators mimic sexual reproduction and mutation



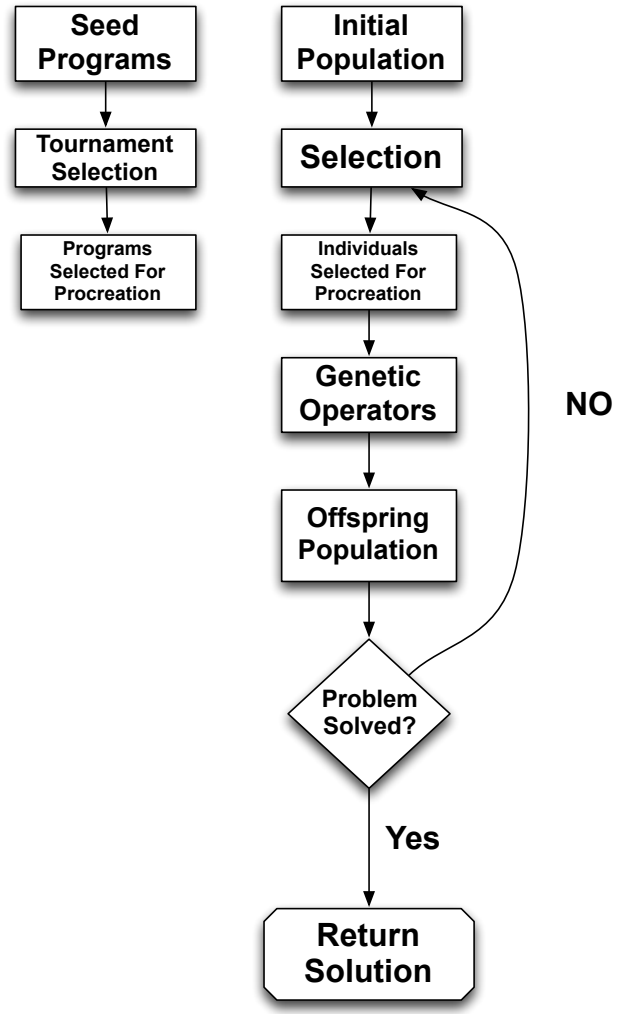
Genetic Programming

- Uses the EC technique to evolve programs
- The population is programs



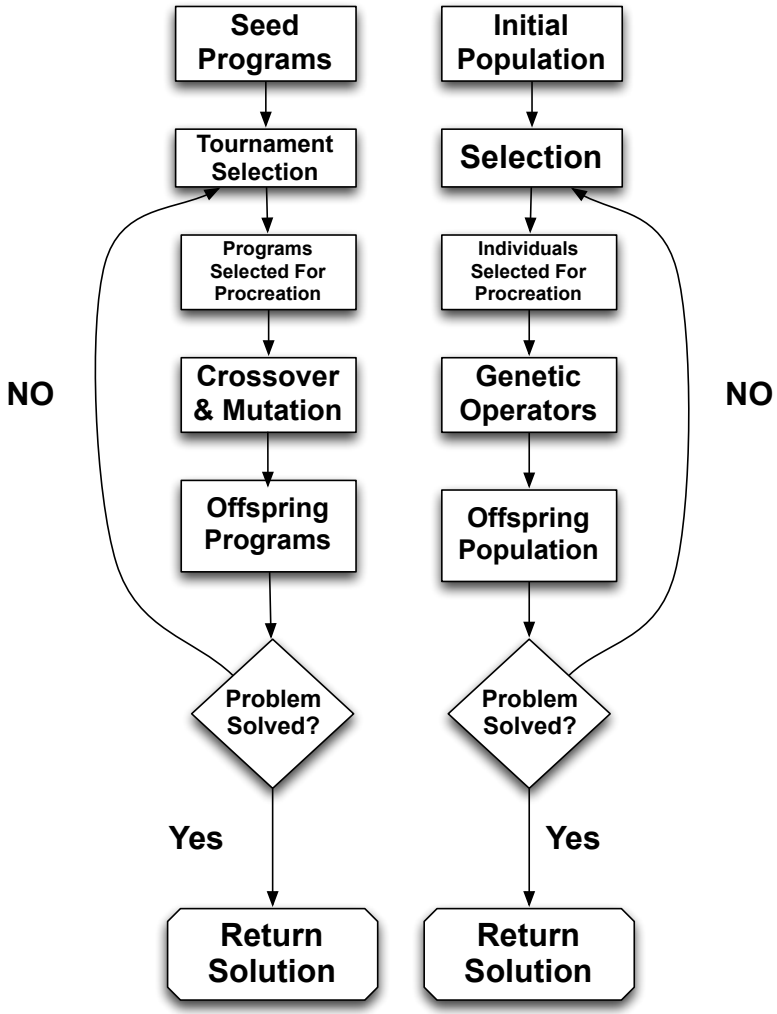
Genetic Programming

- Tournament Selection

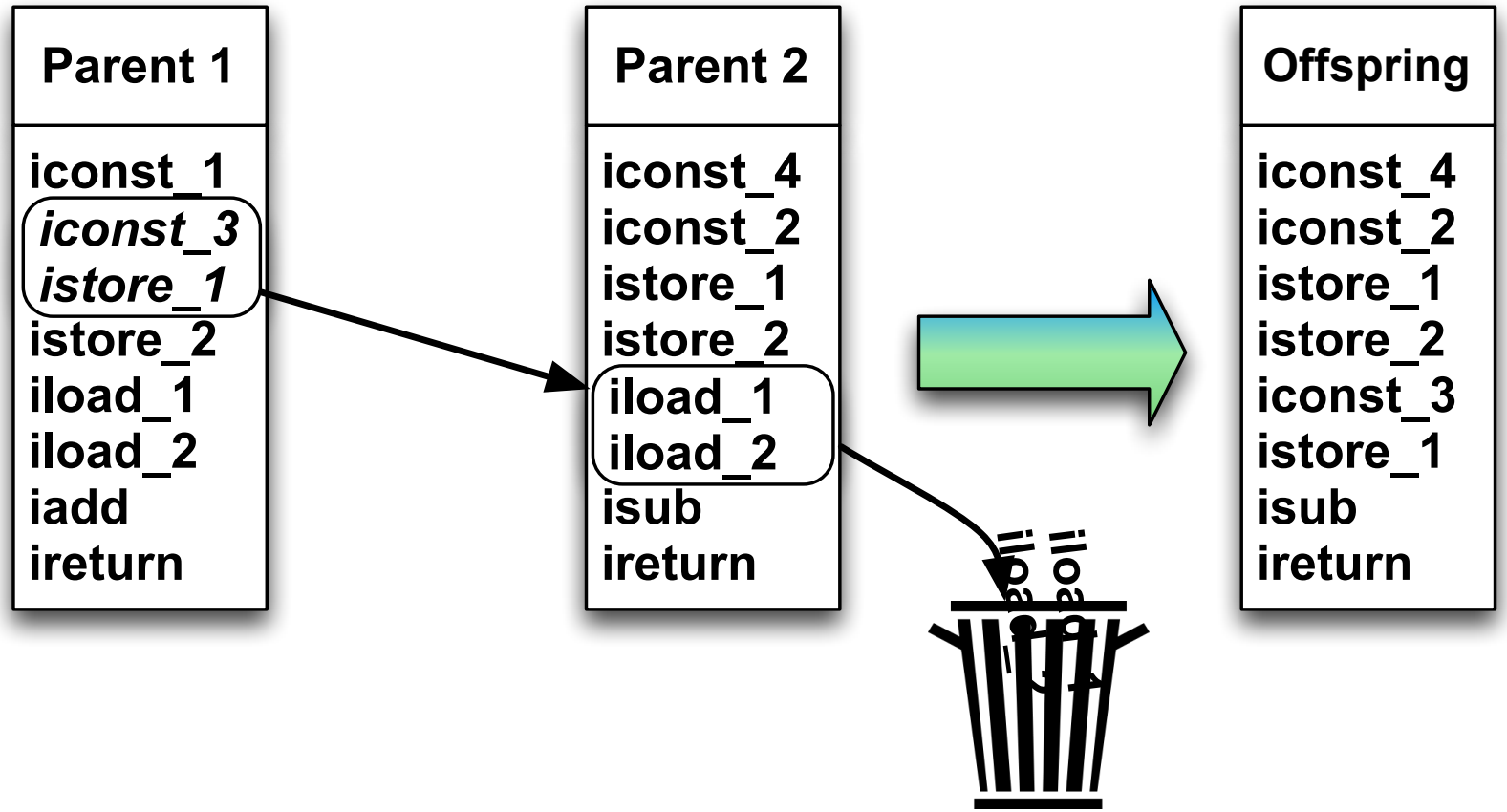


Genetic Programming

- Crossover
- Mutation

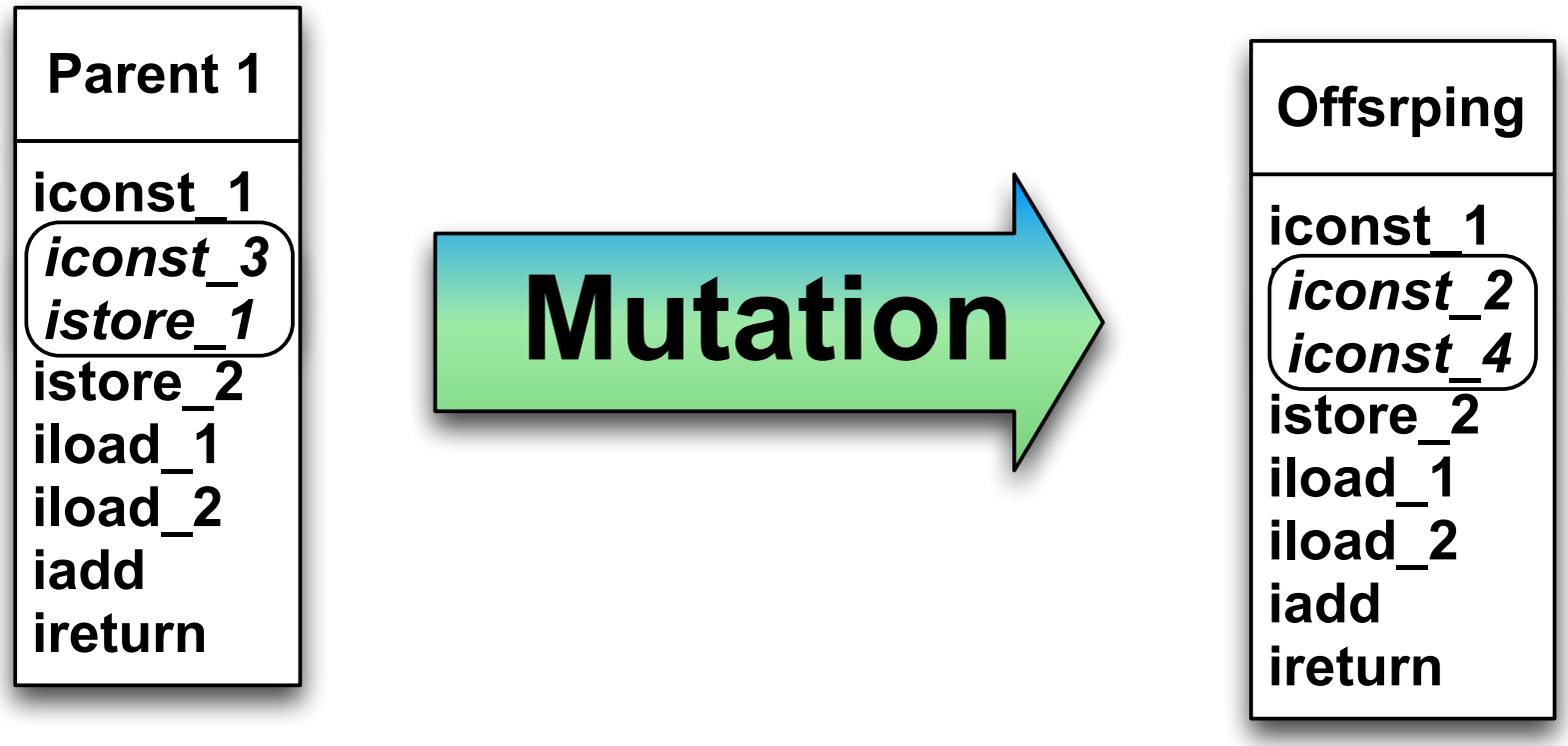


Crossover



Crossover with Java Bytecode

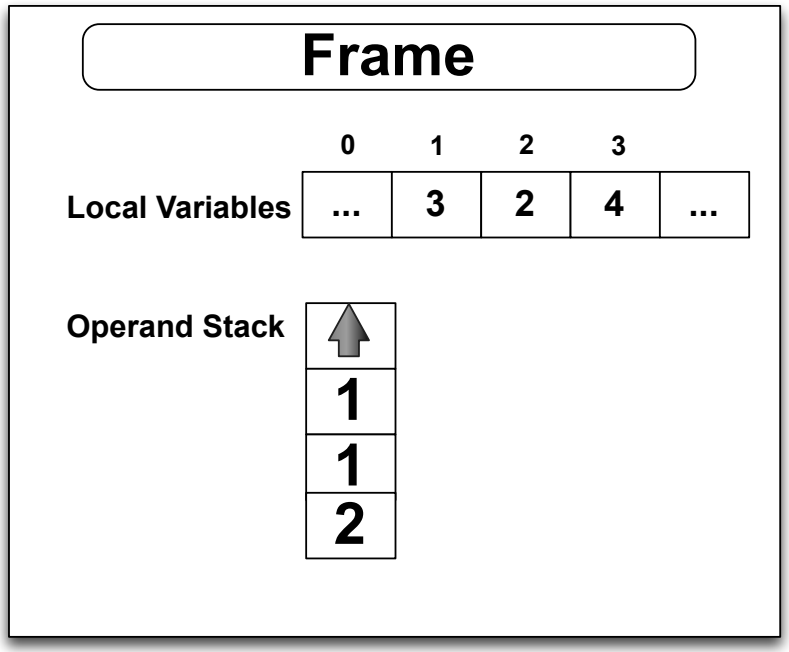
Mutation



Crossover with Java Bytecode

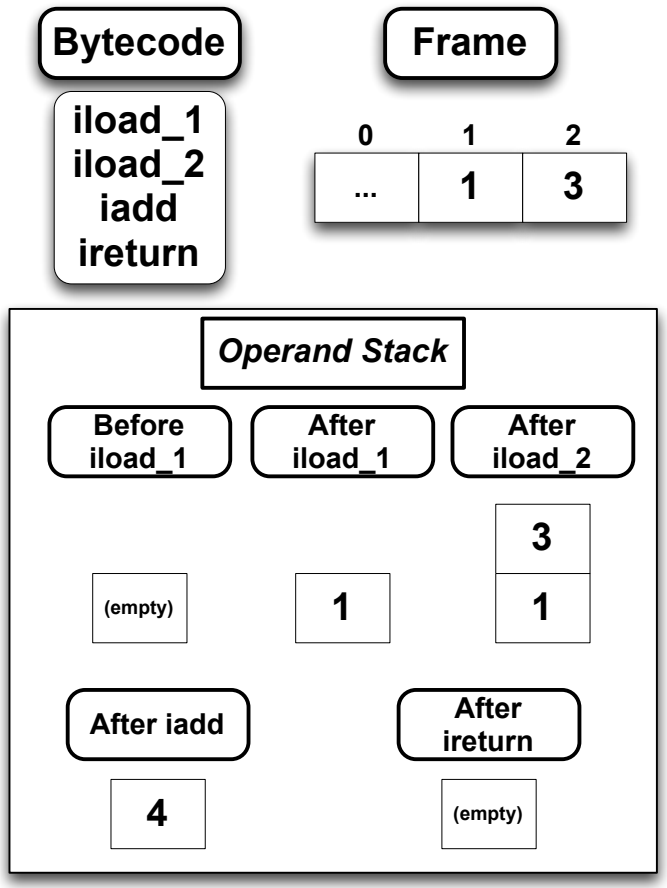
Java Virtual Machine

- Frames
- Array of local variables
- Operand Stack



Java Bytcode and Frames

- Opcodes
- Prefix indicates type



Outline

Source Code Constraints

- While it would be useful, it is difficult to apply evolution to an entire program in source code
  - Source code is made to simplify reading and writing programs
  - Source code does not represent the semantic constraints of the program.

Source Code Constraints

```
float x;  int y = 7;
if (y >= 0)
   x = y;
else
   x = -y;
System.out.println(x);
```

(a)

```
int x = 7;  float y;
if (y >= 0) {
   y = x;
   x = y;
}
System.out.println(z);
```

(b)

Both (a) and (b) are valid syntactically. However (b) is invalid semantically.

Source Code Constraints

- EAs are usually designed to avoid dealing with semantic constraints

```
class Robot{
...
  double robotSpeed(){
      double evolvedVariable = valueFromEA;
      return (robot.location + evolvedVariable)/2;
  }
...
}
```

Instruction-Level Code Constraints

- Consists of a smaller alphabets
- Simpler syntactically
- Less semantic constraints to violate

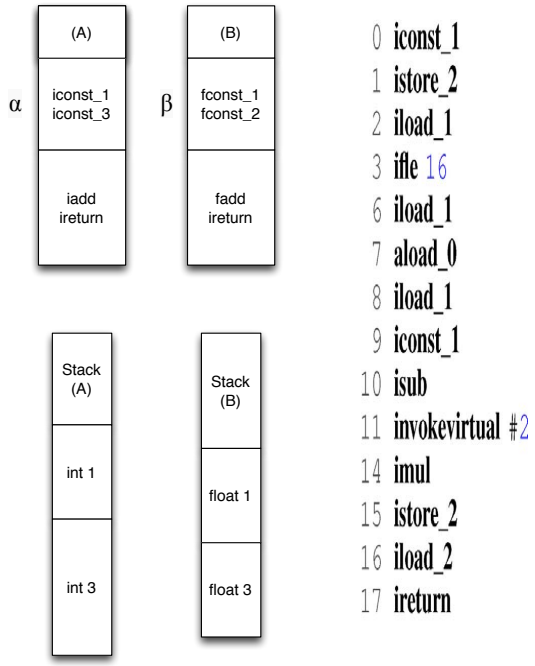
Instruction-level Code Benefits

- Do not need to understand the structure of the program being evolved
- Can evolve a lot from a little
- If there is a compiler for it we can evolve it

Outline

Selecting Offspring

- There is still a chance to produce non-compilable code
- Solution: Add restrictions to code selection.
- Stack and Frame Depth
- Variable Types
- Control Flow



Crossover

Non-Halting Offspring

Outline

Selecting Offspring

Genetic Operators

Non-Halting Offspring

Outline

Conclusions

References