Teach Your Computer to Code FizzBuzz

Frances Buontempo

@fbuontempo
frances.buontempo@gmail.com
overload@accu.org

Chris Simons

@chrislsimons
chris.simons@uwe.ac.uk
www.fet.uwe.ac.uk/~clsimons/
Workshop materials can be found at

http://www.fet.uwe.ac.uk/~clsimons/ACCU2019/

JCLEC requires Java SE Development Kit, e.g. version 8

https://www.oracle.com/technetwork/java/javase/overview/index.html
Evolutionary computing

Frameworks for evolutionary computing

Java Class Library for Evolutionary Computing (JCLEC)

Optimisation problems:
  1 - ‘OneMax’ Problem
  2 - How to program your way out of a paper bag
  3 - FizzBuzz

Genetic Programming, Genetic Improvement
Evolutionary computing

Frameworks for evolutionary computing

Java Class Library for Evolutionary Computing (JCLEC)

Optimisation problems:
1 - ‘OneMax’ Problem
2 - How to program your way out of a paper bag
3 - FizzBuzz

Genetic Programming, Genetic Improvement
When we want to optimise:

• We might want to maximise ‘quality’, and/or
• We might want to minimise ‘cost’...

We want to find the ‘best’ (for whatever ‘best’ means for our situation)
So what’s the problem? **combinatorial explosion**

Calculate the size of the search space

*Given a Solution model, how many different combinations can it represent?*

**Cloud balancing**

<table>
<thead>
<tr>
<th>Model:</th>
<th>Computer</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>

**Traveling salesman (TSP)**

<table>
<thead>
<tr>
<th>Model:</th>
<th>linked list</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
</tr>
</tbody>
</table>

**Course scheduling**

<table>
<thead>
<tr>
<th>Model:</th>
<th>Period</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Room</td>
<td></td>
</tr>
</tbody>
</table>

Search space:

- **Cloud balancing**: \(c^p\)
- **Traveling salesman (TSP)**: \(n!\)
- **Course scheduling**: \((p \times r)^t\)

<table>
<thead>
<tr>
<th># computers</th>
<th># processes</th>
<th>search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>100</td>
<td>300</td>
<td>(10^{600})</td>
</tr>
<tr>
<td>200</td>
<td>600</td>
<td>(10^{1800})</td>
</tr>
<tr>
<td>400</td>
<td>1200</td>
<td>(10^{5987})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># customers</th>
<th>search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>100</td>
<td>(10^{155})</td>
</tr>
<tr>
<td>1000</td>
<td>(10^{1367})</td>
</tr>
<tr>
<td>10000</td>
<td>(10^{10593})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># periods</th>
<th># rooms</th>
<th># lectures</th>
<th>search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>64</td>
</tr>
<tr>
<td>36</td>
<td>9</td>
<td>100</td>
<td>(10^{273})</td>
</tr>
<tr>
<td>36</td>
<td>18</td>
<td>400</td>
<td>(10^{12114})</td>
</tr>
<tr>
<td>36</td>
<td>36</td>
<td>800</td>
<td>(10^{4890})</td>
</tr>
</tbody>
</table>

http://www.optaplanner.org/blog/2014/03/27/searchSpaceSizeCalculation.png
natural evolution
i.e. the change in the inherited characteristics of biological populations over successive generations.

environment
Selection of fittest individuals

sexual reproduction for population diversity / variety

What can we do about it?
Computational evolution

**Representation** of an “individual” solution
e.g. arrays, trees, models, code etc. etc.

initialise population at random
evaluate each individual
while( not done )
   select parents
   recombine pairs of parents
   mutate new candidate individuals
   evaluate each individual
   select candidates for next generation
end while
Ideas from biology (1)

Information concerning the characteristics of a solution individual is encoded in ‘genes’ – all the gene values of an individual is known as the genotype.

Typically, many individuals make up a population.

Individuals can become parents from whom offspring are created. The offspring help to form the new generation, and can themselves become parents in the next generation. Evolutionary algorithms can run for many generations, until some termination condition.
**Ideas from biology (2)**

*Evaluation* of a solution *individual* gives some *fitness* value or *cost* value that is to be optimised, either *maximised* or *minimised*.

Only the fittest solution *individuals* are *selected* to breed *offspring*; individuals can enter a *tournament*, the fittest wins the right to breed.

*Diversity* in the *population* is maintained by:

**Recombination** (sexual reproduction)

| 1 1 1 1 1 | 1 1 1 0 0 |
| 0 0 0 0 0 | 0 0 0 1 1 |

parents | offspring

**Mutation** (asexual reproduction)

| 0 0 0 0 0 | 0 0 0 1 0 |
| 0 0 0 1 1 | |

parent | offspring
Many applications of Evolutionary Computing

Examples include many well-known optimisation problems such as

- course timetabling,
- nurse rostering,
- process scheduling,
- network routing,
- vehicle delivery scheduling,
- load balancing,
- Etc. etc.

The 2006 NASA ST5 spacecraft antenna. This complicated shape was found by an evolutionary computer design program to create the best radiation pattern.

Evolutionary computing

**Frameworks for evolutionary computing**

Java Class Library for Evolutionary Computing (JCLEC)

Optimisation problems:
  1 - ‘OneMax’ Problem
  2 - How to program your way out of a paper bag
  3 - FizzBuzz

Genetic Programming, Genetic Improvement
Frameworks for Evolutionary Computing

Characteristics include:

- mechanisms for the integration of problem-specific knowledge, such as problem constraints and fitness function(s);
- components to configure and monitor the execution, allowing the user to set execution parameters and visualise intermediate results;
- designed with *best practices* and *design patterns* in mind.
<table>
<thead>
<tr>
<th>Language</th>
<th>Framework</th>
<th>version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>Evolutionary Computation Framework (ECF)</td>
<td>1.4.2</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Evolving Objects (EO)</td>
<td>1.3.1</td>
<td>2012</td>
</tr>
<tr>
<td></td>
<td>jMetalCpp</td>
<td>1.7</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>Malba</td>
<td>2.0</td>
<td>2009</td>
</tr>
<tr>
<td></td>
<td>Open Beagle</td>
<td>3.0.3</td>
<td>2007</td>
</tr>
<tr>
<td></td>
<td>OptFrame</td>
<td>2.2</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>PaGMO</td>
<td>2.5</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>ParadisEO</td>
<td>2.0.1</td>
<td>2012</td>
</tr>
<tr>
<td>Java</td>
<td>Java-based Evolutionary Computation Research System (ECJ)</td>
<td>24.0, 25.0</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Evolutionary Algorithms Workbench (EvA)</td>
<td>2.2.0</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>Java Class Library for Evolutionary Computation (JCLEC)</td>
<td>4.0</td>
<td>2014</td>
</tr>
<tr>
<td></td>
<td>jMetal</td>
<td>5.3</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Multi-Objective Evolutionary Algorithm (MOEA) Framework</td>
<td>2.12</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Opt4J</td>
<td>3.1.4</td>
<td>2015</td>
</tr>
<tr>
<td>C#</td>
<td>GeneticSharp (A C# Genetic Algorithm Library)</td>
<td>On-going</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>HeuristicLab (A Paradigm-Independent and Extensible Environment for Heuristic Optimization)</td>
<td>3.3.14</td>
<td>2016</td>
</tr>
<tr>
<td>Python</td>
<td>Distributed Evolutionary Algorithms in Python (DEAP)</td>
<td>1.1.0</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>jMetalPy</td>
<td>On-going</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Pyevolve</td>
<td>0.6rc_1</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>PyGMO</td>
<td>On-going</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Pyvolution</td>
<td>1.1</td>
<td>2012</td>
</tr>
<tr>
<td>Matlab</td>
<td>Genetic and Evolutionary Algorithm Toolbox for Matlab (GEATbx)</td>
<td>3.8</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Global Optimisation Toolbox</td>
<td>R2017b</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td>Matlab Platform for Evolutionary Multi-objective Optimisation (PlatEMO)</td>
<td>1.3</td>
<td>2017</td>
</tr>
</tbody>
</table>

Many frameworks available!

For further details, see Overload 142

https://accu.org/index.php/journals/c380/
Evolutionary computing

Frameworks for evolutionary computing

**Java Class Library for Evolutionary Computing (JCLEC)**

Optimisation problems:

1. ‘OneMax’ Problem
2. How to program your way out of a paper bag
3. FizzBuzz

Genetic Programming, Genetic Improvement
Time to look at an example of a evolutionary computing framework

http://jclec.sourceforge.net
Time to download the framework [http://jclec.sourceforge.net]

We need this **FIRST**

But we don’t need this

And we need this **SECOND**
You can copy an Eclipse or NetBeans project, or

From `jclec4_base.zip`, copy extracted JCLEC source files to an IDE of your choosing, e.g.
From `jclec4_base.zip`, extract and let the IDE know about the required libraries, e.g.
From `jclec4-tutorial.zip`, place tutorial source files in a package called ‘tutorial’.
Each JCLEC project has an XML-based configuration file to specify:

1. Evolutionary algorithm components used, and
2. parameter set up

XML configuration files have a `.cfg` suffix
Also from **jclec4-tutorial.zip**, copy tutorial example configuration files to a folder called `examples`, e.g.
Note

• For the Travelling Salesman Problem (TSP) example, you may also need `orderarray` package from `jclec4-tutorial.zip`
Evolutionary computing

Frameworks for evolutionary computing

Java Class Library for Evolutionary Computing (JCLEC)

Optimisation problems:

1 - ‘OneMax’ Problem
2 - How to program your way out of a paper bag
3 - FizzBuzz

Genetic Programming, Genetic Improvement
The “hello world” of evolutionary computing!

The OneMax problem consists of maximising the number of ones in a bitstring.

Let’s take a length of 100 bits for the bitstring.

e.g. http://tracer.lcc.uma.es/problems/onemax/onemax.html

Yes, I know, we can do this in our heads : ) but it’s a good example of getting going with the framework...
Algorithm design and parameter set up – let’s apply some patterns...

**Representation**
- how to encode a candidate solution?
  
a binary array

**Fitness**
- how to evaluate the fitness of a candidate solution?
  
count the number of 1s

**Diversity**
- how to make offspring different to parents?
  
crossover and mutation

**Initialisation:** random

**Evolution:** simple generational with elitism (SGE)

... and suggested parameters

**Population size:** 100 individuals

**Stop Criterion:** 50 generations

**Parent selection:** tournament of 2 individuals
Make a new file "OneMax.java"

And make a new folder “OneMax” in ”examples” for the configuration file – “OneMax.cfg”
Don’t forget to invoke the executable with “OneMax.cfg” as an argument
Enjoy the programming!

Here’s one way to solve the OneMax optimisation problem...
Here’s one possible configuration:

```xml
<experiment>
  <process algorithm-type="net.sf.jclec.algorithm.classic.SGE">
    <rand-gen-factory type="net.sf.jclec.util.random.RanecuFactory" seed="987328938"/>
    <population-size>100</population-size>
    <max-of-generations>50</max-of-generations>
    <species type="net.sf.jclec.binarray.BinArrayIndividualSpecies" genotype-length="100"/>
    <evaluator type="tutorial.OneMax"/>
    <provider type="net.sf.jclec.binarray.BinArrayCreator"/>
    <parents-selector type="net.sf.jclec.selector.TournamentSelector">
      <tournament-size>2</tournament-size>
    </parents-selector>
    <recombinator type="net.sf.jclec.binarray.rec.UniformCrossover" rec-prob="0.9" />
    <mutator type="net.sf.jclec.binarray.mut.OneLocusMutator" mut-prob="0.2" />
    <listener type="net.sf.jclec.listener.PopulationReporter">
      <report-frequency>5</report-frequency>
      <report-on-file>true</report-on-file>
      <save-complete-population>true</save-complete-population>
      <report-title>"OneMax-"</report-title>
    </listener>
  </process>
</experiment>
```
One way of solving the fitness evaluation:

```java
@Override
protected void evaluate( IIndividual ind ) {
    // Individual genotype
    byte[ ] genotype = ( (BinArrayIndividual)ind).getGenotype( );
    int bitCount = 0;

    for( int i = 0; i < genotype.length; i++ ) {
        if( genotype[ i ] == 1 ) {
            bitCount++;
        }
    }

    ind.setFitness( new SimpleValueFitness( bitCount ) );
}
```
Demonstration
Evolutionary computing

Frameworks for evolutionary computing

Java Class Library for Evolutionary Computing (JCLEC)

**Optimisation problems:**
1. ‘OneMax’ Problem
2. *How to program your way out of a paper bag*
3. FizzBuzz

Genetic Programming, Genetic Improvement
Let’s suppose there’s a canon in a paper bag

Let’s also suppose:
width of bag is 10.0,
height of bag is 5.0.

• *Overload*, 21(118):7–9, December 2013
  • [http://accu.org/index.php/journals/1821](http://accu.org/index.php/journals/1821)
Given a bag with bottom left corner at \((k, 0)\), of width \(w\), and height \(h\), assuming the projectile is smaller than the bag, the cannon is a point of no size, and given the acceleration due to gravity, \(g\), after time \(t\) the projectile will be at point \((x, y)\) where

\[
x = k + \frac{1}{2}w + vt \cos \theta
\]
\[
y = vt \sin \theta - \frac{1}{2}gt^2
\]

\(x\) is the horizontal displacement and \(y\) the vertical displacement. The projectile will just escape when \(y \geq h\) and \(x < k\) or \(x > k + w\).

\(g\) will be taken as 9.81 m/s\(^2\). For simplicity, the code will assume \(k\) is zero.
Fitness evaluation

• Launching at random, something either ends in or out of the bag
• But some fail cases are less bad than others

• 3 escape
• 2 on left get “close”
• Could “close” mean height (at edge of bag)?
  • Fitness = height
Diversity Preservation

**Recombination**

\[ \theta_0 \, v_0 \quad \theta_1 \, v_1 \quad \theta_2 \, v_2 \quad \theta_3 \, v_3 \]

Choose two:

\[ \theta_1 \, v_1 \quad \theta_2 \, v_2 \]

Combine:

\[ \theta_1 \, v_2 \]

**Mutation**

\[ \theta_0 \, v_0 \quad \theta_1 \, v_1 \quad \theta_2 \, v_2 \quad \theta_3 \, v_3 \]

Choose:

\[ \theta_2 \, v_2 \]

Mutate:

\[ \theta_2^1 \, v_2^1 \]
Algorithm design and parameter set up – let’s again apply some patterns...

**Representation**
- how to encode a candidate solution in the population?

**Fitness**
- how to evaluate the fitness of a candidate solution?

**Diversity**
- how to make offspring different to parents?

**Initialisation**: random?

**Evolution**: simple generational with elitism (SGE)?

... and suggested parameters

**Population size**: 12 individuals

**Stop Criterion**: 20 generations

**Parent selection**: tournament of 2 individuals
<experiment>
  <process algorithm-type="net.sf.jclec.algorithm.classic.SGE">
    <population-size>12</population-size>
    <max-of-generations>20</max-of-generations>
    <rand-gen-factory type="net.sf.jclec.util.random.RanecuFactory" seed="124321453"/>
    <species type="net.sf.jclec.realarray.RealArrayIndividualSpecies">
      <genotype-schema>
        <locus type="net.sf.jclec.util.range.Interval" left="0.0" right="20.0"
          closure="closed-closed"/>
        <locus type="net.sf.jclec.util.range.Interval" left="0.0" right="180.0"
          closure="closed-closed"/>
      </genotype-schema>
    </species>
    <evaluator type="tutorial.PaperBag"/>
    <provider type="net.sf.jclec.realarray.RealArrayCreator"/>
    <parents-selector type="net.sf.jclec.selector.TournamentSelector"
      tournament-size="2"/>
    <mutator type="net.sf.jclec.realarray.mut.NonUniformMutator" mut-prob="0.15"/>
    <recombinator type="net.sf.jclec.realarray.rec.BLXAlphaCrossover" rec-prob="0.9"
      alpha="0.3"/>
    <listener ...</listener>
  </process>
</experiment>
Don’t forget to invoke the executable with "PaperBag.cfg” as an argument
Enjoy the programming!

Hint – think about converting the angle theta to radians before applying $\sin()$ and $\cos()$

Here’s one way to solve the ‘Out of a Paper Bag’ optimisation problem...
Let’s start with a ‘Point’ class (with public x & y attributes for convenience):

```java
public class Point {
    public double x;
    public double y;

    public Point() {
        x = 0.0;
        y = 0.0;
    }
}
```
public class PaperBag extends AbstractEvaluator {

    protected boolean maximize = true;

    private Comparator<IFitness> COMPARATOR;

    /* list of x,y points of the projectile trajectory */
    private List<Point> pointsList;

    private DecimalFormat df;  // for debugging

    private static final double GRAVITY = 9.81; // gravity i.e. 9.81 m/sec^2
    private static final double WIDTH = 10.0; // width of the paper bag
    private static final double HEIGHT = 5.0; // height of the paper bag
    private static final double STEP = 0.1; // the "time step"

    // ...
One way of solving the fitness evaluation:

```java
protected void evaluate( IIndividual ind ) {
    // Individual genotype
    double[ ] genotype = ((RealArrayIndividual)ind).getGenotype( );
    double velocity = genotype[ 0 ];
    double theta = genotype[ 1 ];

    pointsList = new ArrayList< >( ); // clear out the list of points

    // calculate the points of the parabolic trajectory
    for( double time = 0.0; time < END; time += STEP ){
        Point p = getPointAtTime( time, velocity, theta );
        pointsList.add( p );
    }

    // ...

    Continued....
```
double fitness = 0.0;

// calculate fitness value from the parabolic trajectory points
for( Point p : pointsList ) {
    if( p.x <= 0.0 || p.x >= WIDTH ) {
        fitness = p.y;
        break;
    }
}

ind.setFitness( new SimpleValueFitness( fitness ) );
The `getPointAtTime()` method:

```java
private Point getPointAtTime( final double time, final double velocity,
final double theta )
{
    double inRadians = Math.toRadians( theta ); // convert to radians
    double xTemp = 0.5 * WIDTH;
    double xIncrement = velocity * time * Math.cos( inRadians );
    xTemp += xIncrement;

    double yTemp = velocity * time * Math.sin( inRadians );
    double yIncrement = 0.5 * GRAVITY * ( time * time );
    yTemp -= yIncrement;
    // can't have a negative y value - this is the ground!
    if( yTemp < 0.0 ) yTemp = 0.0;

    Point p = new Point( );
p.x = xTemp; p.y = yTemp;
    return p;
}
```
Demonstration
<table>
<thead>
<tr>
<th>Generation</th>
<th>Report</th>
<th>Best Individual</th>
<th>Worst Individual</th>
<th>Median Individual</th>
<th>Average Fitness</th>
</tr>
</thead>
</table>
Evolutionary computing

Frameworks for evolutionary computing

Java Class Library for Evolutionary Computing (JCLEC)

Optimisation problems:

1 - ‘OneMax’ Problem
2 - How to program your way out of a paper bag
3 - FizzBuzz

Genetic Programming, Genetic Improvement
Rules of FizzBuzz

A typical game of FizzBuzz involves counting through a sequence of numbers starting at one, but multiples of three are substituted with ‘Fizz’, multiples of five are substituted with ‘Buzz’, and multiples of fifteen are substituted with ‘FizzBuzz’.
Let’s start by evolving a sequence of 100 integers...

• First define the actual sequence in the configure method in a `FizzBuzz` class
  • Here are some constants that might be useful...

```java
private List<Integer> sequence; // to evolve against

private static final int SEQUENCE_SIZE = 100;

private static final int FIZZ = 3;
private static final int BUZZ = 5;
private static final int FIZZ_BUZZ = 15;

private static final Integer FIZZ_IDENTIFIER = 101;
private static final Integer BUZZ_IDENTIFIER = 102;
private static final Integer FIZZ_BUZZ_IDENTIFIER = 103;
```
Enjoy the programming!

Here’s one way to solve the ‘FizzBuzz as a sequence’ problem...
public void configure(Configuration settings) {
    sequence = new ArrayList< >();

    for( int i = 1; i <= SEQUENCE_SIZE; i++ ) {
        sequence.add( new Integer( i ) );
    }

    for( int i = 1; i < sequence.size( ); i++ ) {
        int number = sequence.get( i );
        if( number % FIZZ_BUZZ == 0 ) {
            sequence.set( i, new Integer( FIZZ_BUZZ_IDENTIFIER ) );
        } else if( number % BUZZ == 0 ) {
            sequence.set( i, new Integer( BUZZ_IDENTIFIER ) );
        } else if( number % FIZZ == 0 ) {
            sequence.set( i, new Integer( FIZZ_IDENTIFIER ) );
        }
    }
}
Here’s the fizzbuzz.cfg file (1 of 2):

```xml
<experiment>
    <process algorithm-type="net.sf.jclec.algorithm.classic.SGE">
        <rand-gen-factory type="net.sf.jclec.util.random.RanecuFactory"
                          seed="123456789"/>
        <population-size>100</population-size>
        <max-of-generations>3000</max-of-generations>

        <provider type="net.sf.jclec.intarray.IntArrayCreator" />
        <species type="net.sf.jclec.intarray.IntArrayIndividualSpecies"
                 genotype-length="100">
            <genotype-schema>
                <locus type="net.sf.jclec.util.intset.Interval"
                        left="1" right="103" closure="closed-closed" />
                <locus type="net.sf.jclec.util.intset.Interval"
                        left="1" right="103" closure="closed-closed" />

                ... Etc. for each integer in the array of 100 integers ...
            </genotype-schema>
        </species>
    </process>
</experiment>
```

Continued...
Here’s the fizzbuzz.cfg file (2 of 2):

```xml
<evaluator type="tutorial.FizzBuzz"> </evaluator>

<parents-selector type="net.sf.jclec.selector.TournamentSelector">
    <tournament-size>2</tournament-size>
</parents-selector>

<mutator type="net.sf.jclec.intarray.mut.OneLocusMutator" mut-prob="0.2" />
<recombinator type="net.sf.jclec.intarray.rec.OnePointCrossover" rec-prob="0.9"/>

<listener type="net.sf.jclec.listener.PopulationReporter">
    <report-frequency>10</report-frequency>
    <report-on-file>false</report-on-file>
    <save-complete-population>false</save-complete-population>
    <report-title>FizzBuzz-</report-title>
</listener>
</process>
</experiment>
```
The evaluation( ) method is almost trivial – something like ‘OneMax’?

protected void evaluate( IIndividual ind ) {
    int [] genotype = ( ( IntArrayIndividual ) ind ).getGenotype( );

    int matchCount = 0;

    for( int i = 0; i < genotype.length; i++ ) {
        if( genotype[ i ] == sequence.get( i ) ) {
            matchCount++;
        }
    }

    ind.setFitness( new SimpleValueFitness( matchCount ) );
}
Demonstration
But there’s a problem with this approach....

• How do you write the fitness function without writing code to generate the correct values?
• What properties do we need?
• Listen to @KevlinHenney
every result is 'Fizz', 'Buzz', 'FizzBuzz' or a decimal string, every decimal result corresponds to its ordinal position, every third result contains 'Fizz', every fifth result contains 'Buzz', every fifteenth result is 'FizzBuzz', the ordinal position of every 'Fizz' result is divisible by 3, the ordinal position of every 'Buzz' result is divisible by 5, the ordinal position of every 'FizzBuzz' result is divisible by 15
actual = [fizzbuzz(n) for n in range(1, 101)]

truths = [
    all(a in {'Fizz', 'Buzz', 'FizzBuzz'} or a.isdecimal() for a in actual),
    all(int(a) == n for n, a in enumerate(actual, 1) if a.isdecimal()),
    all('Fizz' in a for a in actual[2::3]),
    all('Buzz' in a for a in actual[4::5]),
    all(a == 'FizzBuzz' for a in actual[14::15]),
    all(n % 3 == 0 for n, a in enumerate(actual, 1) if a == 'Fizz'),
    all(n % 5 == 0 for n, a in enumerate(actual, 1) if a == 'Buzz'),
    all(n % 15 == 0 for n, a in enumerate(actual, 1) if a == 'FizzBuzz')
]

assert all(truths)
Change the array to a tree

• Previous genotypes were “list/array”
• What if we use trees?
• What do trees make?
• Code
But JCLEC doesn’t specifically offer AST representations... and that looked a bit like Python...

Distributed Evolutionary Algorithms in Python (DEAP)

https://deap.readthedocs.io/en/master/
So let’s evolve a tree


• You add a primitive set,
  pset = PrimitiveSet("main", 2)

• Add function and number operators:
  pset.addPrimitive(max, 2)
  pset.addPrimitive(operator.add, 2)
  pset.addPrimitive(operator.neg, 1)

• Or values:
  pset.addTerminal(3)

• Then generate trees:
  expr = genFull(pset, min_=1, max_=3)
  tree = PrimitiveTree(expr)
For Fizz Buzz

```python
def if_then_else(x, y, z):
    if x:
        return y
    else:
        return z

def mod3(x):
    return operator.mod(x, 3) == 0

def mod5(x):
    return operator.mod(x, 5) == 0

def mod15(x):
    return operator.mod(x, 15) == 0

def both(x, y):
    return x and y

def either(x, y):
    return x or y

pset = gp.PrimitiveSet("MAIN", 1)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(both, 2)
pset.addPrimitive(either, 2)
pset.addPrimitive(operator.mod, 2)
pset.addPrimitive(if_then_else, 3)
pset.addPrimitive(mod3, 1)
pset.addPrimitive(mod5, 1)
pset.addPrimitive(mod15, 1)
pset.addTerminal("Buzz")
pset.addTerminal("Fizz")
pset.addTerminal("FizzBuzz")

Can get it to find numbers, e.g.
pset.addEphemeralConstant("rand101", lambda: random.randint(-1,1))
But don’t have a working example yet
```
Evolve...

creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", gp.PrimitiveTree,
    fitness=creator.FitnessMax)

toolbox = base.Toolbox()

toolbox.register("expr", gp.genHalfAndHalf,
    pset=pset, min_=1, max_=2)

toolbox.register("individual", tools.initIterate,
    creator.Individual, toolbox.expr)

toolbox.register("population", tools.initRepeat, list,
    toolbox.individual)

toolbox.register("compile", gp.compile, pset=pset)

# or genFull or genGrow
# genHalfAndHalf does grow 50% or
# time, Full 50%
def fizz_buzz(func, points):
    passed = 0
    def safe_run(func, x):
        try:
            return func(x)
        except:
            return -1
    results = [safe_run(func, x) for x in points
    if every_result_is_Fizz_Buzz_FizzBuzz_or_a_decimal(results):
        passed += 1
    if every_decimal_result_corresponds_to_its_ordinal_position(results):
        passed += 1
    if every_third_result_contains_Fizz(results):
        passed += 1
    if every_fifth_result_contains_Buzz(results):
        passed += 1
    if every_fifteenth_result_contains_FizzBuzz(results):
        passed += 1
    if the_ordinal_position_of_every_Fizz_result_is_divisible_by_3(results):
        passed += 1
    if the_ordinal_position_of_every_Buzz_result_is_divisible_by_5(results):
        passed += 1
    if the_ordinal_position_of_every_FizzBuzz_result_is_divisible_by_15(results):
        passed += 1
    return passed
def register(fn):
    # e.g. our fizz buzz tests fitness function

def eval(individual, points):
    # This is our custom evaluation function
    func = toolbox.compile(expr=individual)
    return fn(func, points),

toolbox.register("evaluate", eval, points=range(101)) # and we register it here

toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("mate", gp.cxOnePoint)
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=pset)

toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"), max_value=17))
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"), max_value=17))

def main():
    random.seed(318)
    register(fizz_buzz)
    pop = toolbox.population(n=4000)
    hof = tools.HallOfFame(1) # The best

    pCrossover = 0.75
    pMutation = 0.5
    nGen = 75
    pop, log = algorithms.eaSimple(pop, toolbox, pCrossover, pMutation, nGen, halloffame=hof)
    return pop, log, hof
Ta-da!

The Hof

if_then_else(mod15(if_then_else(if_then_else(mod15(x), 'FizzBuzz'), 'Fizz', 'Buzz'), x, if_then_else('Buzz', 'FizzBuzz', mod3(x)))), 'FizzBuzz', if_then_else(both(if_then_else(if_then_else(mod15(x), either('FizzBuzz', 'FizzBuzz'), 'FizzBuzz'), if_then_else('FizzBuzz', mod15(mod5(x)), 'Buzz'), 'Buzz'), if_then_else('Fizz', 'Buzz', if_then_else('Buzz', if_then_else(if_then_else('Buzz', if_then_else(if_then_else(mod3(x), x, 'FizzBuzz'), if_then_else(x, x, either('Buzz', 'Buzz'))), x), 'Fizz'), 'Fizz', x), if_then_else(either(if_then_else(x, x, mod3(x)), 'FizzBuzz'), 'Fizz', 'Fizz')))), if_then_else(mod15(x), either('FizzBuzz', either('Buzz', x)), if_then_else(mod3(x), 'Fizz', x)), 'Buzz'))
if_then_else(mod15(if_then_else(if_then_else(mod15(x), 'FizzBuzz'), 'Fizz', 'Buzz'), x, if_then_else('Buzz', 'FizzBuzz', if_then_else(both(if_then_else(if_then_else(mod15(x), either('FizzBuzz', 'Buzz'), 'FizzBuzz'), if_then_else('FizzBuzz', mod15(mod5(x)), 'Buzz'), 'Buzz'), if_then_else('Fizz', 'Buzz', if_then_else('FizzBuzz', if_then_else(if_then_else('Buzz', if_then_else(if_then_else(mod3(x), x, 'FizzBuzz'), if_then_else(x, x, either('Buzz', 'Buzz')), x), 'Fizz'), 'Fizz', x), if_then_else(either(if_then_else(x, x, mod3(x)), 'FizzBuzz'), 'Fizz', 'Fizz')))), if_then_else(mod15(x), either('FizzBuzz', either('Buzz', x)), if_then_else(mod3(x), 'Fizz', x)), 'Buzz')))}
Evolutionary computing

Frameworks for evolutionary computing

Java Class Library for Evolutionary Computing (JCLEC)

Optimisation problems:
1 - ‘OneMax’ Problem
2 - How to program your way out of a paper bag
3 - FizzBuzz

Genetic Programming, Genetic Improvement
Genetic Programming (GP) - evolution of a tree structure

Evolves the ‘innards’ (white box) of a function or expression

Each tree node is an **operator** or **variable**, or a **terminal node**.

Used widely to evolve functions for
- Curve fitting
- Circuit board design
- Data modelling
- Symbolic regression
- Feature selection
- Classification

https://en.wikipedia.org/wiki/Genetic_programming#/media/File:Genetic_Program_Tree.png
Could we use GP to evolve source code e.g. for FizzBuzz?

Well, using AST representations, as we saw above, yes!

**BUT** GP doesn’t always scale well (e.g. due to ‘code bloat’ with crossover) and code doesn’t look like what a human programmer would produce!!!!

So in practice, maybe better to start with an existing body of code and use *‘Genetic Improvement’ (GI)* e.g.
For further information:

Genetic Algorithms and Machine Learning for Programmers

Create AI Models and Evolve Solutions

A Systematic Review of Interaction in Search-Based Software Engineering

Aurora Ramírez, José Raúl Romero, Member, IEEE, and Christopher L. Simons

Abstract—Search-Based Software Engineering (SBSE) has been successfully applied to automate a wide range of software development activities. Nevertheless, in those software engineering problems where human evaluation and preference are crucial, such insights have proved difficult to characterize in search, and solutions might not look natural when that is the expectation. In an attempt to address this, an increasing number of researchers have reported the incorporation of the “human-in-the-loop” during search and interactive SBSE has attracted significant attention recently. However, reported results are fragmented over different development phases, and a great variety of novel interactive approaches and algorithmic techniques have emerged. To better integrate these results, we have performed a systematic literature review of interactive SBSE. From a total of 669 papers, 28 primary studies were identified. To enable their analysis, we formulated a classification scheme focused on four crucial aspects of interactive search, i.e., the problem formulation, search technique, interactive approach, and the empirical framework. Our intention is that the classification scheme affords a methodological approach for interactive SBSE. Lastly, as well as providing a detailed cross analysis, we identify and discuss some open issues and potential future trends for the research community.

Index Terms—Search-Based Software Engineering, Interaction, Systematic Literature Review, Optimization

1 INTRODUCTION

The design and development of complex, large-scale software systems can be non-trivial and challenging for the software engineer to perform. In an attempt to based problems in the software development lifecycle, this may also be challenging [6] (chapter 9).

In addition, results of automated search approaches
Thanks!

Fran Buontempo

@fbuontempo
frances.buontempo@gmail.com
overload@accu.org

Chris Simons

@chrislsimons
chris.simons@uwe.ac.uk
www.fet.uwe.ac.uk/~clsimons