Evolutionary Computing Frameworks for Optimisation

Aurora Ramírez
Department of Computer Science and Numerical Analysis,
University of Córdoba, Córdoba, Spain.
aramirez@uco.es
www.uco.es/users/aramirez/en/

Chris Simons
Department of Computer Science and Creative Technologies,
University of the West of England, Bristol, United Kingdom.
chris.simons@uwe.ac.uk
www.cems.uwe.ac.uk/~clsimons/

13 September 2017
Agenda

• Artificial Intelligence and learning (*briefly*)
• Evolutionary Computation
• Frameworks
• Examples of application
• Discussion: practicalities
• Opportunities for collaboration?
• Pub!
**Assertion:** artificial intelligence is seen in software that appears to learn.

For instance, when we have loads of pre-existing data, software can learn to classify categories in the data (by rule induction, clustering, etc. etc.)

The software *learns a model* of data, based on known inputs and outputs.

The intention is to predict future outputs for unknown inputs.

**Question:** But what happens when there is no pre-existing data, or there is no access to pre-existing data?

Specifically, what happens when we know the sort of outputs we want (e.g. maximise this, minimise that...) but don’t know what sort of inputs that would achieve this?

This is *optimisation*... and it’s a different form of *learning*
**Question:** So how do we develop software to perform optimisation?

**Answer:** define a space of all the possible solutions, and then search through the space.

**Hmmmm....** we could exhaustively enumerate over each solution, but search spaces get very big, very quickly...
Calculate the size of the search space

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

Cloud balancing

Traveling salesman (TSP)

Course scheduling

Model: Computer → Process

Model: linked list

Model: Room → Lecture

Search space: \( n! \)

Search space: \( (p \times r)^l \)

Search space: \( c^p \)

<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^{1380} )</td>
</tr>
<tr>
<td>400</td>
<td>1200</td>
<td>( 10^{6967} )</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^{157} )</td>
</tr>
<tr>
<td>1000</td>
<td>( 10^{2567} )</td>
</tr>
<tr>
<td>10000</td>
<td>( 10^{35659} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># periods</th>
<th># rooms</th>
<th># lectures</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>64</td>
</tr>
<tr>
<td>36</td>
<td>6</td>
<td>100</td>
<td>( 10^{233} )</td>
</tr>
<tr>
<td>36</td>
<td>18</td>
<td>400</td>
<td>( 10^{1124} )</td>
</tr>
<tr>
<td>36</td>
<td>36</td>
<td>800</td>
<td>( 10^{2490} )</td>
</tr>
</tbody>
</table>

http://www.optaplanner.org/blog/2014/03/27/searchSpaceSizeCalculation.png
So what to do?

Take inspiration from nature...
Agenda

• Artificial Intelligence and learning (*briefly*)

• Evolutionary Computation

• Frameworks

• Examples of application

• Discussion: practicalities

• Opportunities for collaboration?

• Pub!
natural evolution

i.e. the change in the inherited characteristics of biological populations over successive generations.

sexual reproduction for population diversity / variety
Evolutionary Algorithms...

Not new...

- Alan Turin (1952)
  - “Computing Machinery and Intelligence” in *Mind* article hints at a “…genetical programming…”
- Alex Fraser (1957)
  - Computational simulation of natural evolution
- Fogel *et al.* (1966)
  - *Evolutionary programming* (finite state machines)
- Rechenburg (1973)
  - *Evolutionary Strategies*
- Holland (1975)
  - *Genetic Algorithms*
- Kosa (1992)
  - *Genetic Programming*

*And many more*
...via computational evolution

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

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

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

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

There are two distinct needs....

1 - Enable effective variation and diversity (i.e. exploration)
2 - Enable fitness measures for evaluation (i.e. exploitation)
Representation – tree example

$$2 \cdot \pi + \left( (x + 3) - \frac{y}{5 + 1} \right)$$
initialise population at random
while( not done )
   evaluate each individual
   select parents
   recombine pairs of parents
   mutate new candidate individuals
   select candidates for next generation
end while

Fitness measures enable comparison of solution individuals in the population.

Typically,
*Either maximise fitness* e.g. quality, desirability, etc.

*Or minimise cost* e.g. time, resource, money, etc.

Evaluation can comprise:
- One fitness measure (single objective)
- 2 or 3 fitness measures (multi-objective)
- > 3 fitness measures (many-objective)
Challenges for Evolutionary Computation for developers?

- Searching for strategies rather than instances
- Exploiting many-core computing
- **Giving insight to software developers**
- Optimising compilation and deployment
- ‘AI-friendly’ software development and deployment

Facebook’s evolutionary search for crashing software bugs

Ars gets the first look at Facebook’s fancy new dynamic analysis tool.

SEBASTIAN ANTHONY - 22/8/2017, 07:52
Agenda

• Artificial Intelligence and learning (*briefly*)
• Evolutionary Computation
• **Frameworks**
• Examples of application
• Discussion: practicalities
• Opportunities for collaboration?
• Pub!
Evolutionary Optimisation Frameworks (EOFs)

“A set of software tools that provide a correct and reusable implementation of a set of metaheuristics, and the basic mechanisms to accelerate the implementation of its partner subordinate heuristics (possibly including solution encodings and technique-specific operators), which are necessary to solve a particular problem instance using techniques provided”

Evolutionary Optimisation Frameworks (EOFs)

- Adaptable search components (algorithms, operators...)
- Easy integration of problem-specific knowledge (fitness, solution encoding...)
- Mechanisms to configure and monitor the execution
- General utilities to conduct experiments
- Object-oriented paradigm
Evolutionary Optimisation Frameworks (EOFs)

Advantages
- Less coding effort
- Optimised code
- Additional utilities
- Programming environment

Disadvantages
- Which one?
- Learning curve
- External dependencies

## Some EOFs in C++

<table>
<thead>
<tr>
<th>NAME</th>
<th>WEB</th>
<th>CHARACTERISTICS</th>
</tr>
</thead>
</table>
| ECF 1.4.2 (2017) | [http://ecf.zemris.fer.hr/](http://ecf.zemris.fer.hr/) | • Default parameter configuration  
• Parallelism with MPI |
• Parallelism with MPI and OpenMD |
| jMetalCpp 1.7 (2016) | [https://github.com/jMetal/jMetalCpp](https://github.com/jMetal/jMetalCpp) | • C++ (partial) version of jMetal  
• Single and multi-objective problems |
| MALLBA 2.0 (2009) | [http://neo.lcc.uma.es/software/mallba/](http://neo.lcc.uma.es/software/mallba/) | • Exact, heuristic and hybrid algorithms  
• Minimum inheritance/virtual methods |
| Open BEAGLE 3.0.3 (2007) | [https://github.com/chgagne/beagle](https://github.com/chgagne/beagle) | • Configuration files  
• Advanced EC models (co-evolution) |
| OptFrame 2.2 (2017) | [https://sourceforge.net/projects/optframe/](https://sourceforge.net/projects/optframe/) | • Local search and EC algorithms  
• Support for MPI and MapReduce |
• Parallel and distributed algorithms |
# Some EOFs in Java/C#

<table>
<thead>
<tr>
<th>NAME</th>
<th>WEB</th>
<th>CHARACTERISTICS</th>
</tr>
</thead>
</table>
| ECJ 25.0 (2017)     | [https://cs.gmu.edu/~eclab/projects/ecj/](https://cs.gmu.edu/~eclab/projects/ecj/) | • Variety of search algorithms  
                     |                                                             | • Documentation and contributions  |
| EvA 2.2.0 (2015)    | [http://www.ra.cs.uni-tuebingen.de/software/eva2/](http://www.ra.cs.uni-tuebingen.de/software/eva2/) | • Includes a GUI  
                     |                                                             | • Integration with MATLAB  |
                     |                                                             | • Complete graphical environment  |
| JCLEC 4.0 (2014)    | [http://jclec.sourceforge.net/](http://jclec.sourceforge.net/) | • Easy integration of user-defined code  
                     |                                                             | • Machine learning algorithms  |
| jMetal 5.3 (2017)   | [https://jmetal.github.io/jMetal/](https://jmetal.github.io/jMetal/) | • Focused on multi-objective problems  
                     |                                                             | • Recent algorithms and benchmarks  |
                     |                                                             | • Well documented and tested code  |
                     |                                                             | • Includes a GUI  |
Some other EOFs

- DEAP (Python)
  - https://github.com/DEAP

- GEATbx (Matlab)
  - http://www.geatbx.com/

- Pygmo (Python)
  - http://esa.github.io/pygmo/
Agenda

• Artificial Intelligence and learning (*briefly*)
• Evolutionary Computation
• Frameworks
• **Examples of application**
• Discussion: practicalities
• Opportunities for collaboration?
• Pub!
Java Class Library for Evolutionary Computation (JCLEC)

Java and open source

Extension modules
Multi-objective algorithms
Machine learning algorithms

High-level programming style
Interface-oriented design

Easy integration of new code
Configuration with XML files

Design patterns

Visual environment (beta)

Java Reflection
Java Class Library for Evolutionary Computation (JCLEC)

*initialise population at random*

while( not done )
    evaluate each individual
    select parents
    recombine pairs of parents
    mutate new candidate individuals
    select candidates for next generation
end while

**IPopulation**

- getSpecies(): ISpecies
- getEvaluator(): IEvaluator
- getGeneration(): int
- getInhabitants(): List<IIndividual>

**IIndividual**

- getFitness(): IFitness
- setFitness(IFitness): void
- copy(): IIndividual
- equals(): boolean

**IFitness**

- getValue(): double
- setValue(double): void
- isAcceptable(): boolean
- copy(): IFitness

**IProvider**

- provide(int): List<IIndividual>

**ISpecies**

- createIndividual(T[]): IIndividual
Java Class Library for Evolutionary Computation (JCLEC)

initialise population at random

while( not done )
    evaluate each individual
    select parents
    recombine pairs of parents
    mutate new candidate individuals
    select candidates for next generation
end while

---

**IAlgorithm**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute()</td>
<td>void</td>
</tr>
<tr>
<td>pause()</td>
<td>void</td>
</tr>
<tr>
<td>terminate()</td>
<td>void</td>
</tr>
<tr>
<td>addListener(IAlgorithmListener)</td>
<td>void</td>
</tr>
<tr>
<td>removeListener(IAlgorithmListener)</td>
<td>void</td>
</tr>
</tbody>
</table>

**PopulationAlgorithm**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>doInit()</td>
<td>void</td>
</tr>
<tr>
<td>doSelection()</td>
<td>void</td>
</tr>
<tr>
<td>doGeneration()</td>
<td>void</td>
</tr>
<tr>
<td>doReplacement()</td>
<td>void</td>
</tr>
<tr>
<td>doControl()</td>
<td>void</td>
</tr>
</tbody>
</table>

**IAlgorithmListener**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithmStarted()</td>
<td>void</td>
</tr>
<tr>
<td>algorithmCompleted()</td>
<td>void</td>
</tr>
</tbody>
</table>

**IEvaluator**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaluate(List&lt;IIndividual&gt;)</td>
<td>void</td>
</tr>
<tr>
<td>getNumberOfEvaluations()</td>
<td>int</td>
</tr>
</tbody>
</table>
Java Class Library for Evolutionary Computation (JCLEC)

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

ISelector

select(List<IIndividual>): List<IIndividual>
select(List<IIndividual>, int): List<IIndividual>
select(List<IIndividual>, int, boolean): List<IIndividual>

IRedecominator

recombine(List<IIndividual>): List<IIndividual>

IMutator

mutate(List<IIndividual>): List<IIndividual>
Java Class Library for Evolutionary Computation (JCLEC)

```
<experiment>
  <process algorithm-type="net.sf.jclec.algorithm.classic.SGE">
    <rand-gen-factory type="net.sf.jclec.util.random.RanecuFactory" seed="123"/>
    <population-size>100</population-size>
    <max-of-generations>100</max-of-generations>
    <species type="net.sf.jclec.binarray.BinArrayIndividualSpecies"
      genotype-length="100"/>
    <evaluator type="tutorial.Knapsack">
      (...)
    </evaluator>
    <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>10</report-frequency>
      <report-on-file>true</report-on-file>
      <save-complete-population>false</save-complete-population>
      <report-title>Knapsack</report-title>
    </listener>
  </process>
</experiment>
```
Java Class Library for Evolutionary Computation (JCLEC)

VisualJCLEC webpage
Agenda

• Artificial Intelligence and learning (*briefly*)
• Evolutionary Computation
• Frameworks
• Examples of application
• **Discussion: practicalities**
• Opportunities for collaboration?
• Pub!
Got to find the right sort of problem…
What? Allocations, permutations, sequences, etc.
Why? for optimisation, insight discovery

Got to find the right scale of problem…
small problems can be solved with exhaustive search.
so generally large scale problems that can’t be solved otherwise

Just getting something that’s ‘good enough’ can be great!
often can’t prove that you’ve got the single best solution (single objective)
can never prove this for multi/many objective problems

Benchmarking performance – few agreed, standard benchmarks
evolutionary computation OK if execution in ‘reasonable time’?
some comparative surveys available

What happens after you’ve evolved a population of ‘optimal’ solutions?
which one(s) do you select? the developer has to choose?
so could the developer be involved – programmer intuition??

Anything else??
Agenda

• Artificial Intelligence and learning (*briefly*)
• Evolutionary Computation
• Frameworks
• Examples of application
• Discussion: practicalities

• **Opportunities for collaboration?**
• Pub!
Agenda

• Artificial Intelligence and learning (*briefly*)
• Evolutionary Computation
• Frameworks
• Examples of application
• Discussion: practicalities
• Opportunities for collaboration?

• Pub! …Brewdog?? Emyr?