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/

accu bristol

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?

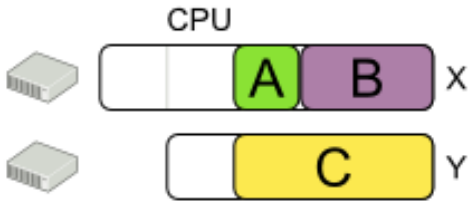
Answer: define a space of all the possible solutions,
and then search through the space.

Hmmmm…. we could exhaustively enumerate over each solution, but search spaces get very big, very quickly…

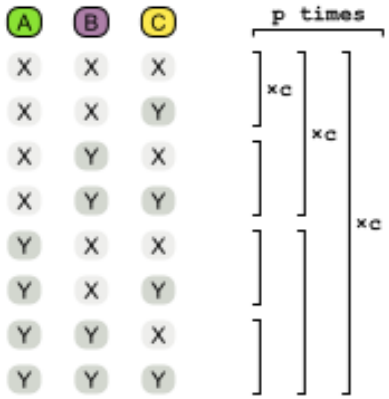
Calculate the size of the search space

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

Cloud balancing



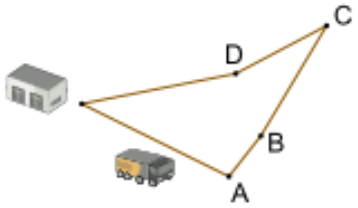
Model: Computer ← Process



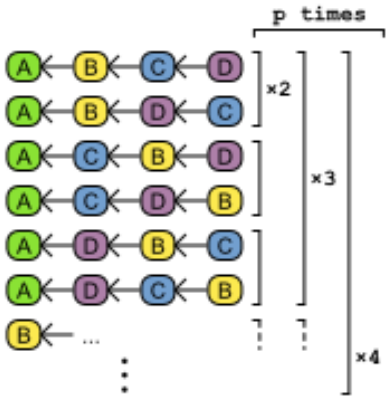
Search space: c^p

# computers	# processes	search space
2	3	8
100	300	10^{600}
200	600	10^{1380}
400	1200	10^{6967}

Traveling salesman (TSP)



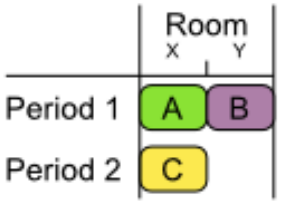
Model: linked list



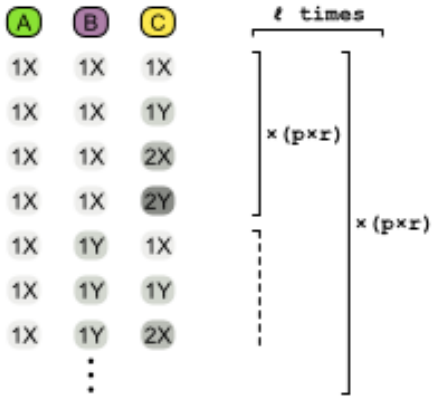
Search space: $n!$

# customers	search space
4	24
100	10^{157}
1000	10^{2567}
10000	10^{35659}

Course scheduling



Model: Period, Room ← Lecture



Search space: $(p \times r)^{\ell}$

# periods	# rooms	# lectures	space
2	2	3	64
36	6	100	10^{233}
36	18	400	10^{1124}
36	36	800	10^{2490}

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!

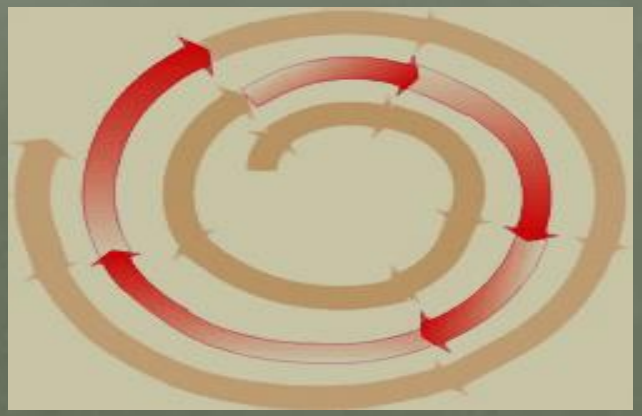
www.flickr.com/photos/armincifuentes/

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

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
```

Eiben, A.E., Smith, J.E. (2015) *Introduction to Evolutionary Computing*, 2nd Edition, Springer.

```
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.

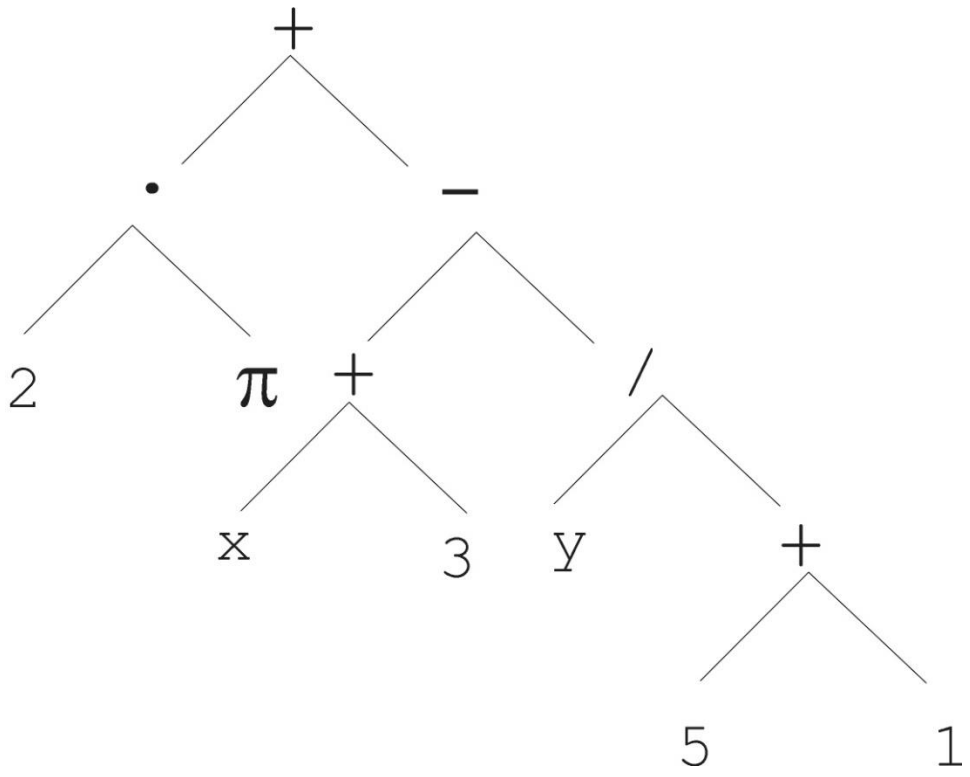
There are two distinct needs….

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:

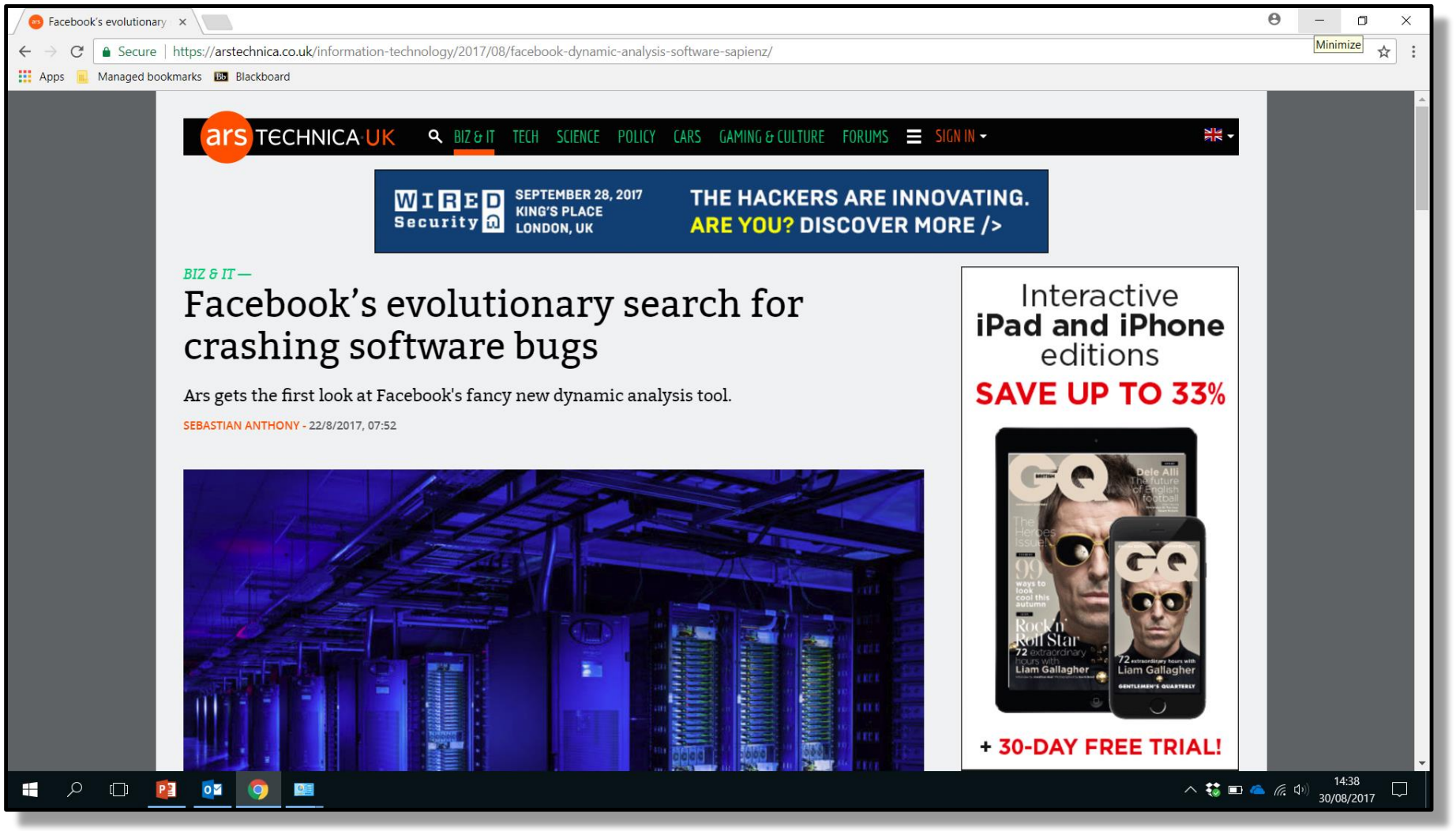
- One fitness measure (single objective)
- 2 or 3 fitness measures (multi-objective)
- > 3 fitness measures (many-objective)

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***

Harman, M. (2012) "The Role of Artificial Intelligence in Software Engineering", *Proceedings of the First International Workshop on Realising Artificial Intelligence in Software Engineering (RAISE)* , pp. 1-6.

Recent example…

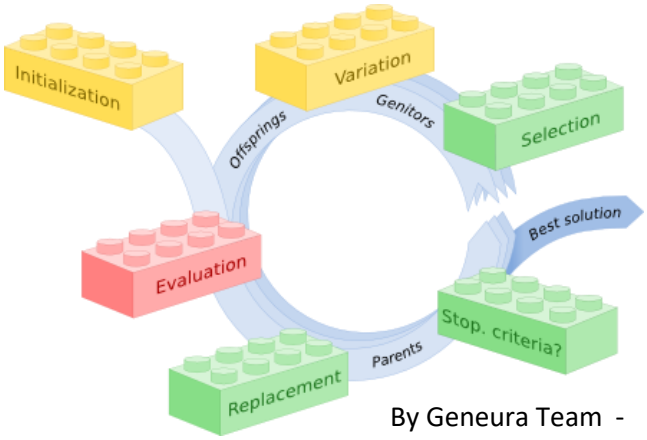


https://arstechnica.co.uk/information-technology/2017/08/facebook-dynamic-analysis-software-sapienz/

Agenda

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

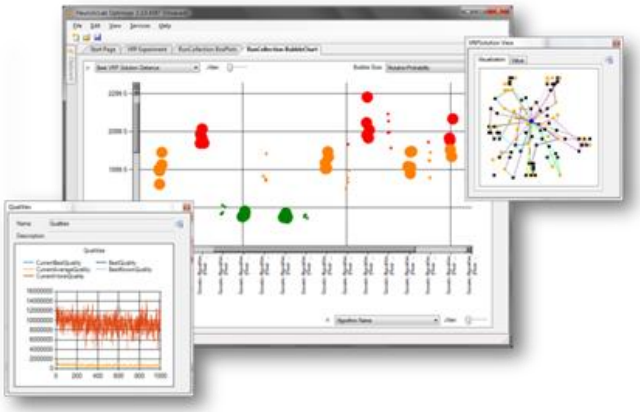
Evolutionary Optimisation Frameworks (EOFs)



By Geneura Team - *http://eodev.sourceforge.net*

*"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"*

Parejo, J.A. *et al.* (2012) "Metaheuristic Optimization Frameworks: A Survey and Benchmarking", *Soft Computing*, vol. 16(3), pp. 527-561.

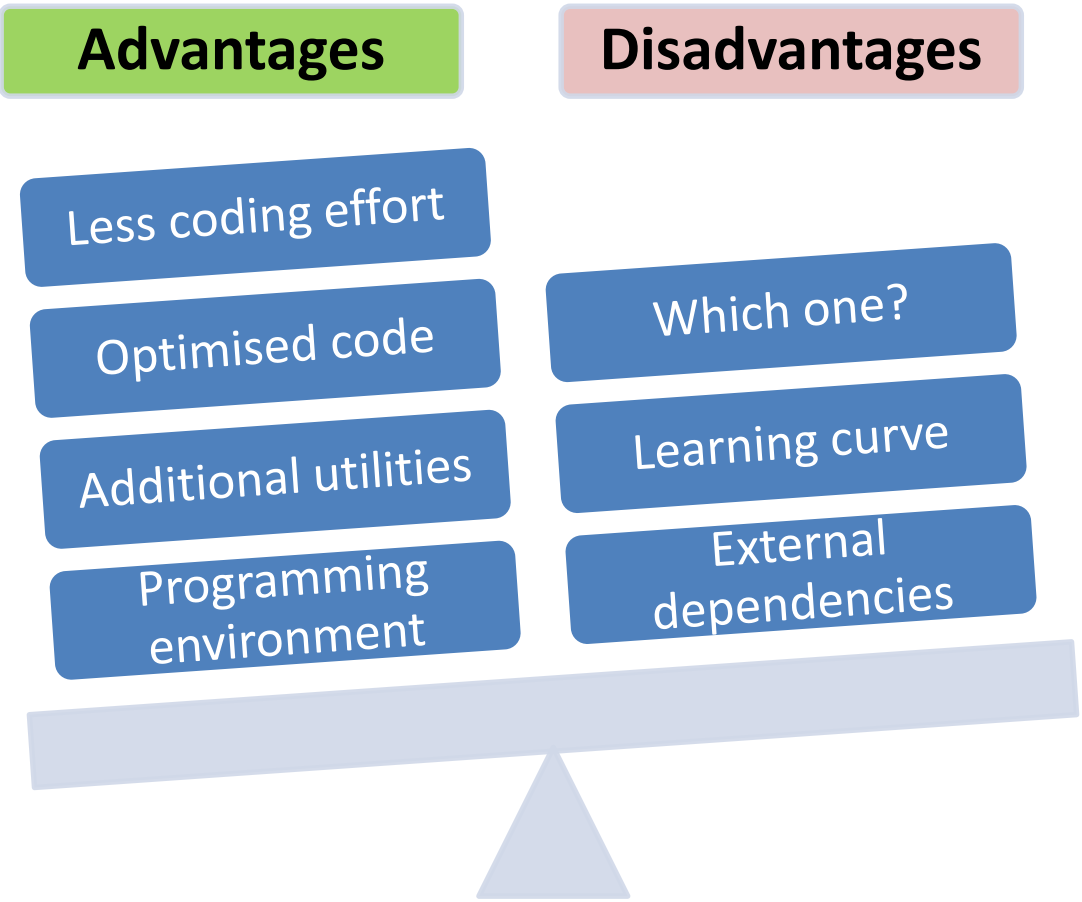


By HEAL - http://dev.heuristiclab.com, GPL, *https://commons.wikimedia.org/w/index.php?curid=18796636*

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)



Adapted from: Parejo, J.A. *et al.* (2012) "Metaheuristic Optimization Frameworks: A Survey and Benchmarking", *Soft Computing*, vol. 16(3), pp. 527-561.

Some EOFs in C++

NAME	WEB	CHARACTERISTICS
ECF 1.4.2 (2017)	http://ecf.zemris.fer.hr/	• Default parameter configuration • Parallelism with MPI
EO 1.3.1 (2012)	http://eodev.sourceforge.net/	• Component-based design • Parallelism with MPI and OpenMD
jMetalCpp 1.7 (2016)	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/	• Exact, heuristic and hybrid algorithms • Minimum inheritance/virtual methods
Open BEAGLE 3.0.3 (2007)	https://github.com/chgagne/beagle	• Configuration files • Advanced EC models (co-evolution)
OptFrame 2.2 (2017)	https://sourceforge.net/projects/optframe/	• Local search and EC algorithms • Support for MPI and MapReduce
ParadisEO 2.0.1 (2012)	http://paradiseo.gforge.inria.fr/	• Four modules: EO, MO, MOEO, SMP • Parallel and distributed algorithms

Some EOFs in Java/C#

NAME	WEB	CHARACTERISTICS
ECJ 25.0 (2017)	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/	• Includes a GUI • Integration with MATLAB
HeuristicLab 3.3.14 (2016) [C#]	http://dev.heuristiclab.com/	• Plugin-based architecture • Complete graphical environment
JCLEC 4.0 (2014)	http://jclec.sourceforge.net/	• Easy integration of user-defined code • Machine learning algorithms
jMetal 5.3 (2017)	https://jmetal.github.io/jMetal/	• Focused on multi-objective problems • Recent algorithms and benchmarks
MOEA Fram. 2.12 (2017)	http://moeaframework.org/	• Catalogue of multi-objective methods • Well documented and tested code
Opt4J 3.1.4 (2015)	http://opt4j.sourceforge.net/	• Highly modular design • 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!

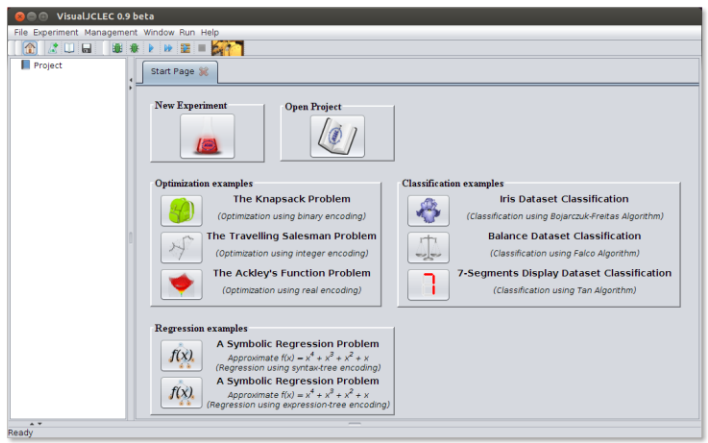
Java Class Library for Evolutionary Computation (JCLEC)

Java and open source

Extension modules

Multi-objective algorithms

Machine learning algorithms



High-level programming style

Interface-oriented design

Design patterns

Easy integration of new code

Configuration with XML files

Java Reflection

Visual environment

(beta)

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

```
execute(): void
pause(): void
terminate(): void
addListener(IAlgorithmListener): void
removeListener(IAlgorithmListener): void
```

PopulationAlgorithm

```
doInit(): void
doSelection(): void
doGeneration(): void
doReplacement(): void
doControl(): void
```

IAlgorithmListener

```
algorithmStarted(AlgorithmEvent): void
algorithmCompleted(AlgorithmEvent): void
```

IEvaluator

```
evaluate(List<IIndividual>): void
getNumberOfEvaluations(): int
```

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>
```

IRecombinator

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

IMutator

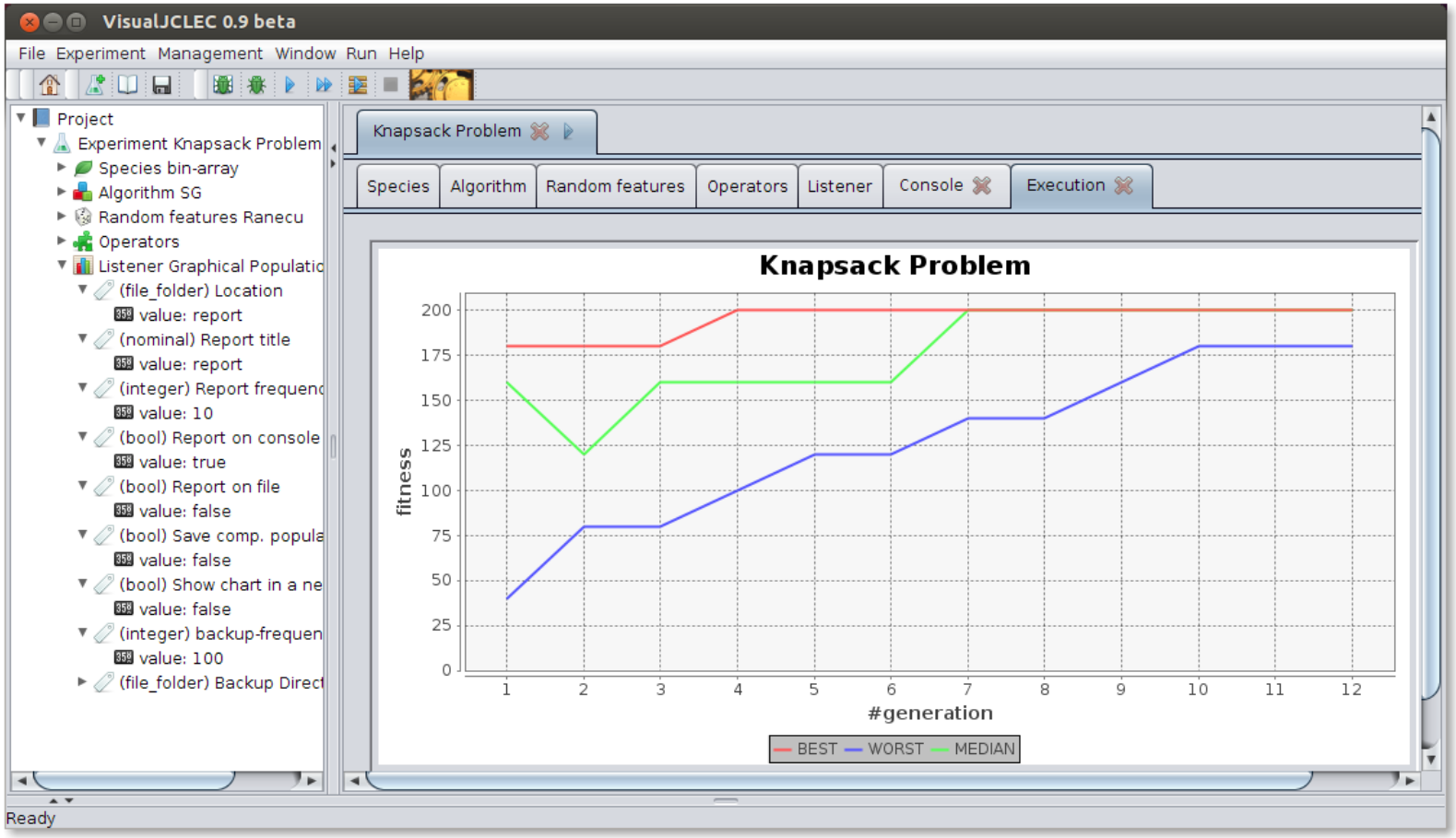
```
mutate(List<IIndividual>): List<IIndividual>
```

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>
```

JCLEC Tutorial

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?***