CSC4504 : Formal Languages & Applications

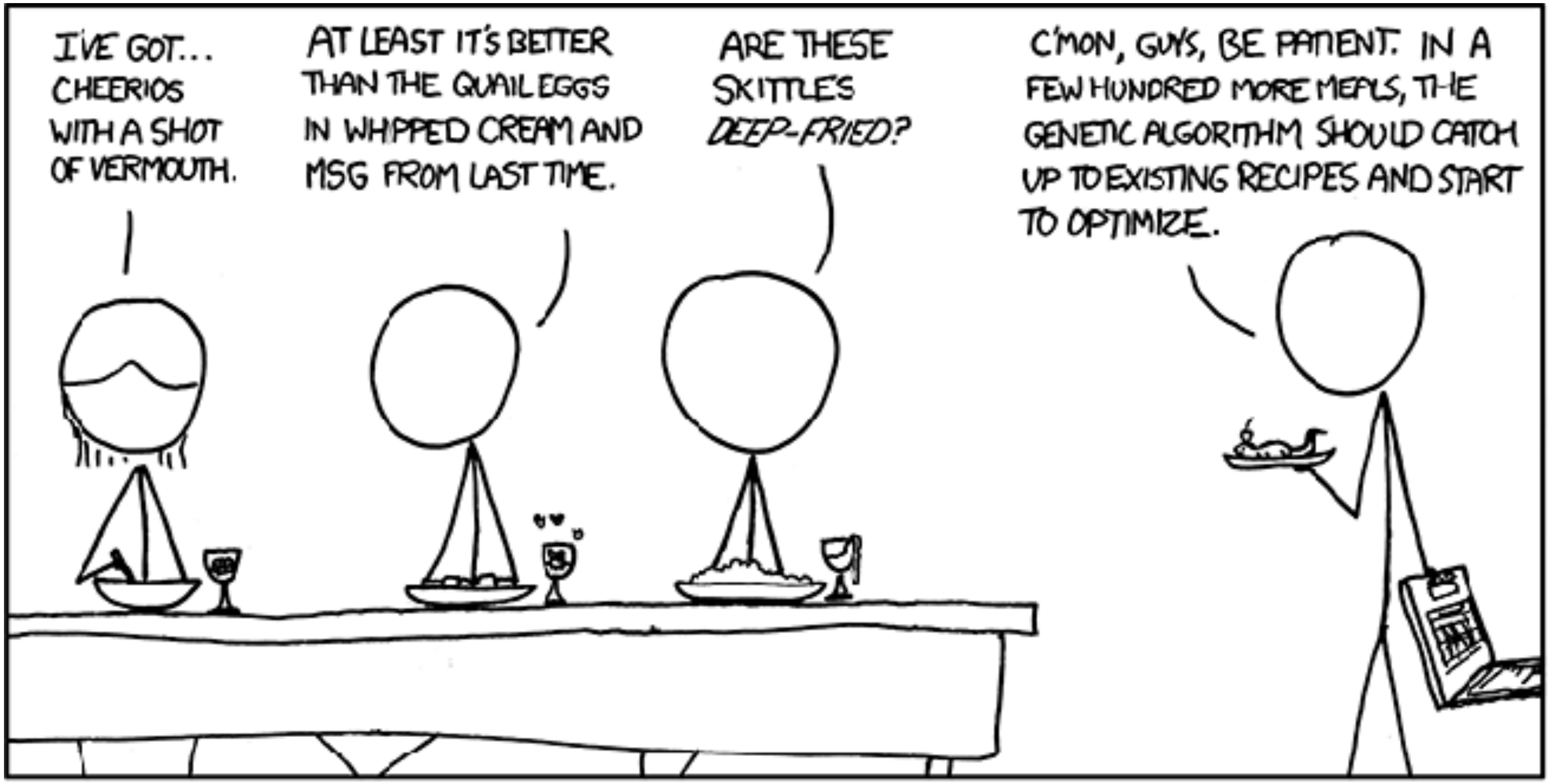
J Paul Gibson D311

paul.gibson@telecom-sudparis.eu

http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC4504/

Genetic Algorithms

/~gibson/Teaching/CSC4504/Problem5-GeneticAlgorithms.pdf


I'VE GOT... CHEERIOS WITH A SHOT OF VERMOUTH.
AT LEAST IT'S BETTER THAN THE QUAIL EGGS IN WHIPPED CREAM AND MSG FROM LAST TIME.
ARE THESE SKITTLES *DEEP-FRIED?*
C'MON, GUYS, BE PATIENT. IN A FEW HUNDRED MORE MEALS, THE GENETIC ALGORITHM SHOULD CATCH UP TO EXISTING RECIPES AND START TO OPTIMIZE.

WE'VE DECIDED TO DROP THE CS DEPARTMENT FROM OUR WEEKLY DINNER PARTY HOSTING ROTATION.

$$\begin{tikzcd}[row sep=large]
\text{Evolutionary Algorithms} \arrow[d] \\
\text{Genetic Algorithms} \arrow[d] \\
\text{Genetic Programming}
\end{tikzcd}$$

Evolutionary Algorithms: *Solve* optimization problems using techniques inspired by natural evolution

Genetic Algorithms: Heuristic_based search, usually based on populations of candidate solutions encoded as strings of characters

Genetic Programming: Members of the population are *computer programs*

Where did the idea come from? - R. M. Friedberg

"Machines would be more useful if they could learn to perform tasks for which they were not given precise methods. Difficulties that attend giving a machine this ability are discussed. It is proposed that the program of a stored-program computer be gradually improved by a learning procedure which tries many programs and chooses, from the instructions that may occupy a given location, the one most often associated with a successful result."

R. M. Friedberg. 1958. A learning machine: part I. *IBM J. Res. Dev.* 2, 1 (January 1958), 2-13. DOI=10.1147/rd.21.0002 http://dx.doi.org/10.1147/rd.21.0002

Where did the idea come from? - Marvin Minsky

“A computer can do, in a sense, only what it is told to do. But even when we do not know how to solve a certain problem, we may program a machine (computer) to search through some large space of solution attempts. Unfortunately, this usually leads to an enormously inefficient process.”

Minsky, Marvin. "Steps toward artificial intelligence." ***Proceedings of the IRE*** **49.1 (1961): 8-30.**

Where did the idea come from? - H. J. Bremermann

"Problems involving vast numbers of possibilities will not be solved by sheer data processing quantity. We must look for quality, for refinements, for tricks, for every ingenuity that we can think of. Computers faster than those of today will be a great help. We will need them. However, when we are concerned with problems in principle, present day computers are about as fast as they ever will be."

H. J. Bremermann. *Optimization through evolution and recombination*. In M. C. Yovits, G. T. Jacobi, and G. D. Golstine, editors, Proceedings of the Conference on Self-Organizing Systems - 1962, pages 93-106, Washington, DC, 1962. Spartan Books..

More Recommended Reading

Goldberg, David E., and John H. Holland. "Genetic algorithms and machine learning." *Machine Learning* 3.2 (1988): 95-99.

Braun, Heinrich. "On solving travelling salesman problems by genetic algorithms." *Parallel Problem Solving from Nature* (1991): 129-133.

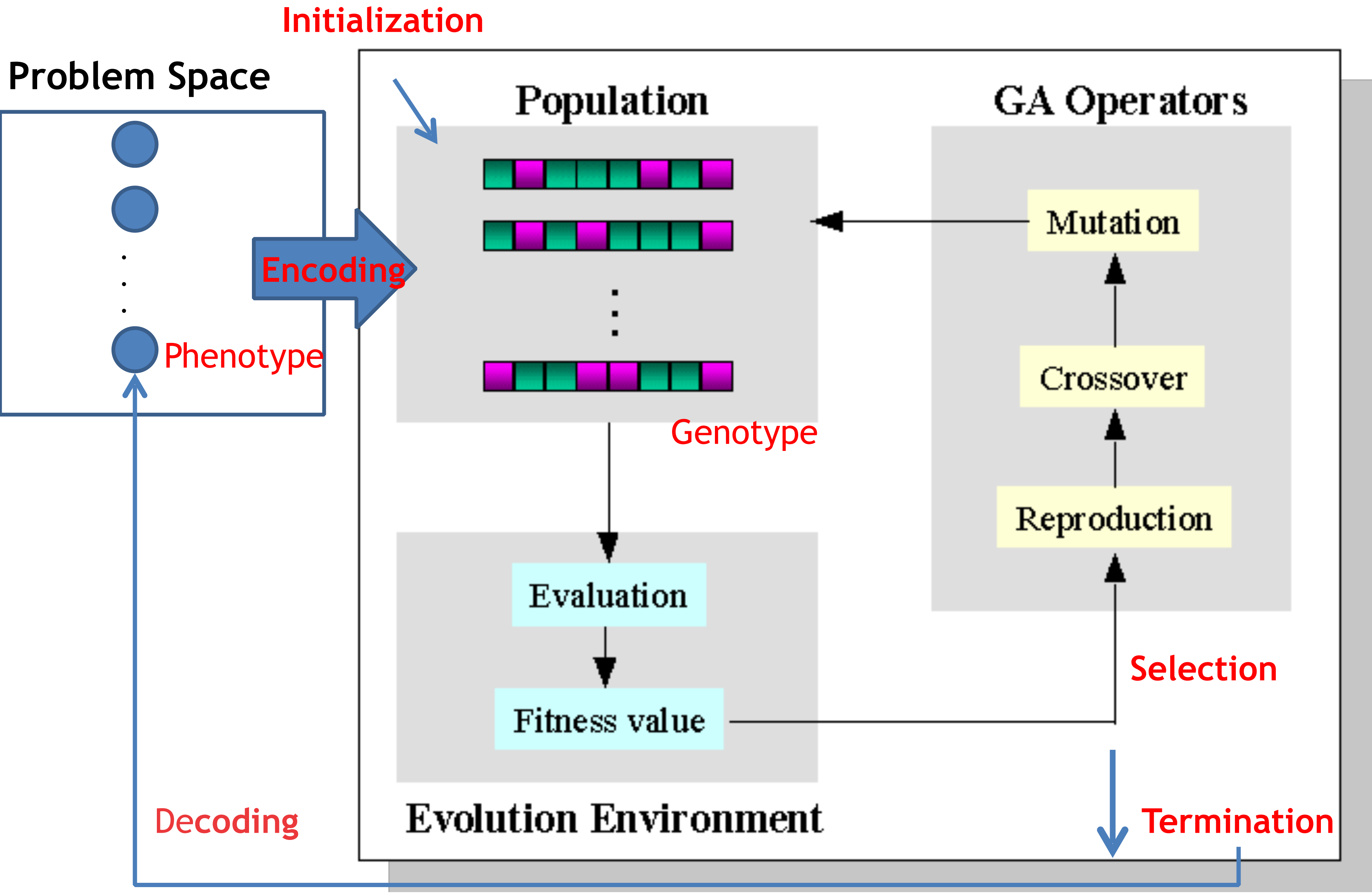
Back, Thomas, Ulrich Hammel, and H-P. Schwefel. "Evolutionary computation: Comments on the history and current state." *Evolutionary computation, IEEE Transactions on* 1.1 (1997): 3-17.

Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." *Evolutionary Computation, IEEE Transactions on* 1.1 (1997): 67-82.

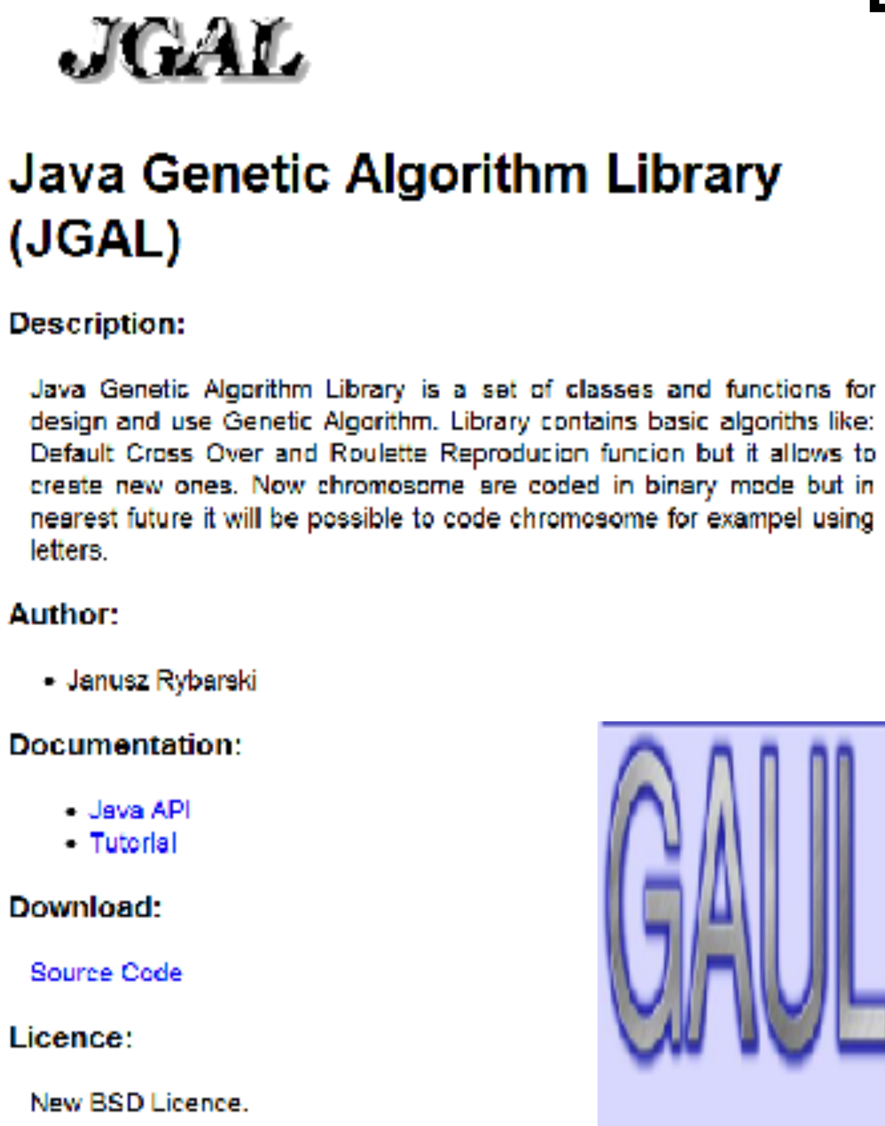
Cantú-Paz, Erick. "A survey of parallel genetic algorithms." *Calculateurs paralleles, reseaux et systems repartis* 10.2 (1998): 141-171.

Banzhaf, Wolfgang, et al. "Genetic programming." *Intelligent Systems and their Applications, IEEE* 15.3 (2000): 74-84.

The Generic Pattern



Lots of Libraries Available



JGAL

Java Genetic Algorithm Library (JGAL)

Description:

Java Genetic Algorithm Library is a set of classes and functions for design and use Genetic Algorithm. Library contains basic algorithms like: Default Cross Over and Roulette Reproducion funcion but it allows to create new ones. Now chromosome are coded in binary mode but in nearest future it will be possible to code chromosome for exampel using letters.

Author:

- Janusz Rybarski

Documentation:

- Java API
- Tutorial

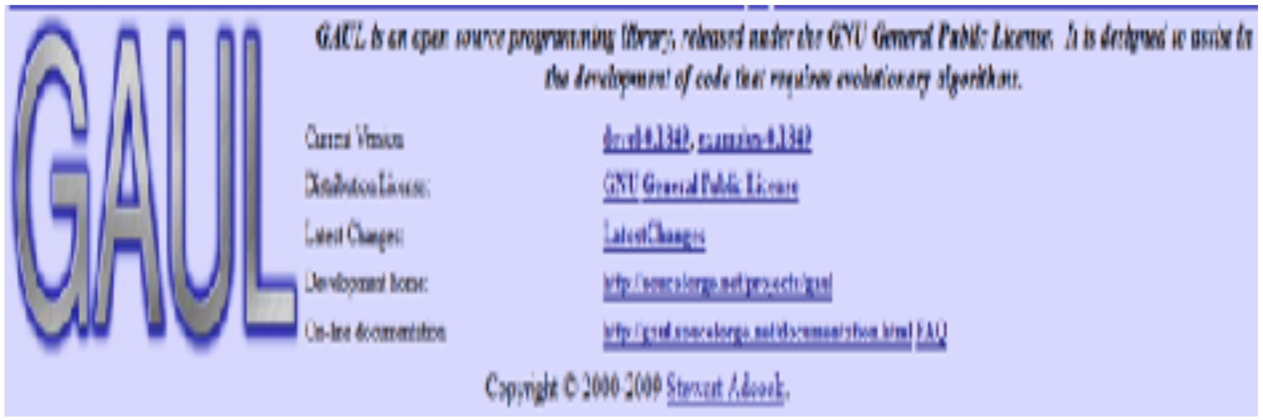
Download:

Source Code

Licence:

New BSD Licence.

pgapack
Parallel genetic algorithm library



GAUL

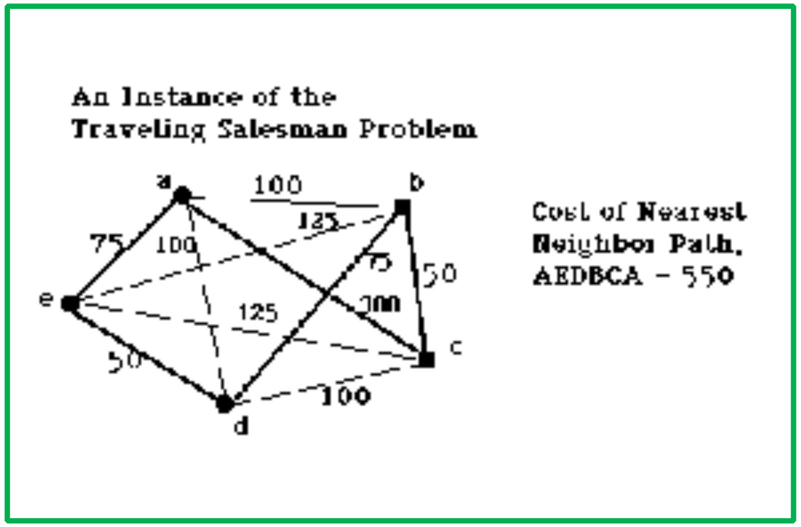
GAUL is an open source programming library, released under the GNU General Public License. It is designed to assist in the development of code that requires evolutionary algorithms.

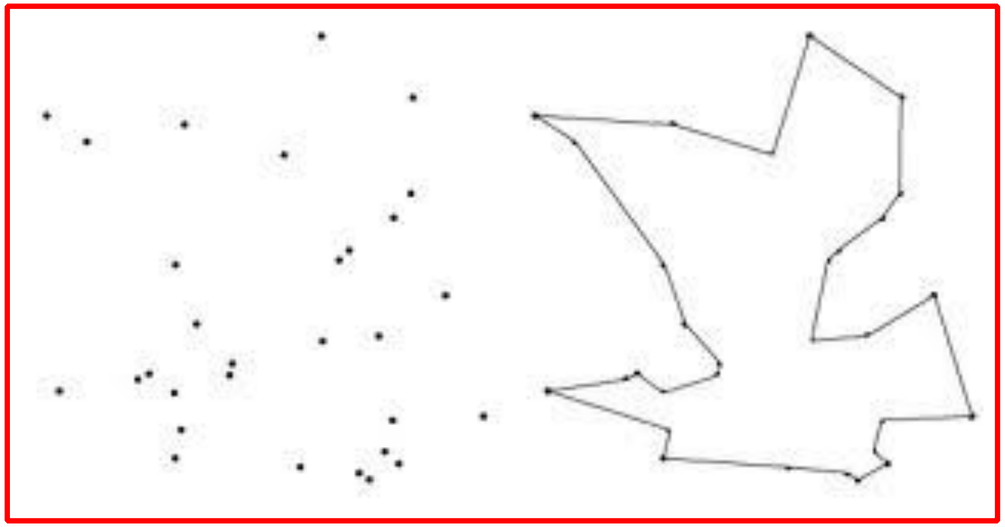
Current Version	devel-0.1849, examples-0.1849
Distribution License:	GNU General Public License
Latest Changes:	LatestChanges
Development home:	http://sourceforge.net/projects/gaul
On-line documentation	http://gaul.sourceforge.net/documentation.html FAQ

Copyright © 2000-2009 Stewart Adcock.

2017: J Paul Gibson

Problem: Write a GA to solve TSP (in Java) <u>without</u> using the generic libraries





Five ‘functions’ to program:

1. Encoding
2. Evaluation
3. Crossover
4. Mutation
5. Decoding

We do not consider this version

Problem: Write a GA to solve TSP (in Java) without using the generic libraries

Five 'functions' to program:

1. Encoding

This is simple: we just use a String to represent the order in which the towns/nodes are visited

As a simplification we can enumerate the towns/nodes so that the string is any permutation of :

1234…n for problems with n nodes.

Note: if the graph is not connected then we can artificially introduce an infinite weighting between any 2 nodes that are not directly connected

Problem: Write a GA to solve TSP (in Java) <u>without</u> using the generic libraries

Five ‘functions’ to program:

2. Evaluation

The fitness function is simple:

just the length of the tour (which we want to minimize)

Problem: Write a GA to solve TSP (in Java) <u>without</u> using the generic libraries

Five 'functions' to program:

3. Crossover

There are lots of possible crossover functions. We will implement a cyclic crossover

```
parent1 = 12345678
parent2 = 85213647

child =   1*******
child =   1******8
child =   1*****78
child =   1**4**78
child =   15243678
```

Choose an arbitrary starting index in parent1 - here we chose 1.

4 is the cycle point

This type of crossover is guaranteed to ensure the ***validity*** of the child's phentotype/genotype (in this case the permutation property)

Problem: Write a GA to solve TSP (in Java) <u>without</u> using the generic libraries

Five ‘functions’ to program:

4. Mutation

Mutation avoids getting stuck in local optimum state spaces which are not globally optimal

After crossover we mutate a percentage of children (defined by the mutation index)

There are many different types of mutation

We choose a simple mutation which inverts a randomly chosen subtour.

For example, child = 12345678, points 3 and 7, mutates to:

child = 12765438

Problem: Write a GA to solve TSP (in Java) <u>without</u> using the generic libraries

Five 'functions' to program:

5. Decoding

This is very simple, once we have a solution that we are happy with then we transform the permutation into a Hamilton circuit.

Problem: Write a GA to solve TSP (in Java) <u>without</u> using the generic libraries

Some additional things to consider:

1) The size of the initial population - risks of too big or too small?
2) The initialization of population state - how to achieve a good distribution?
3) How much mutation - risks of too much or too little?
4) Termination condition - time/iterations/stability
5) Testing - how to test if you don't know the answer?
6) Complexity Analysis - how much time/memory used to find a 'close enough' answer

TO DO: Implement the 5 functions that we have specified as Java methods, and compose into a complete solution.