

# Evolutionary Computation for Problems in Computational Statistics

Robin Nunkesser<sup>1</sup>   Thorsten Bernholt<sup>1</sup>   Oliver Morell<sup>2</sup>  
Holger Schwender<sup>2</sup>

<sup>1</sup>Department of Computer Science, University of Dortmund

<sup>2</sup>Department of Statistics, University of Dortmund

Statistical Computing 2007

## 1 Introduction

## 2 Application

- FrEAK
- Genetic Association Study
- Robust Regression

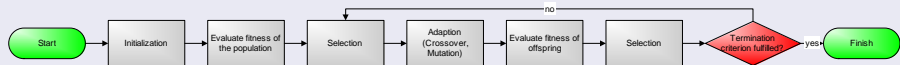
- General term for the usage of search heuristics inspired by natural evolution
  - possible solutions are represented by *individuals*
  - a set of individuals (a *population*) undergoes variation (*crossover* and *mutation*)
  - the *fitness* of the individuals is evaluated
  - a new generation is derived after a *selection* process
- Black-box optimization
- Often used for problems that are not easy to solve by conventional methods
  - Combinatorial optimization
  - Learning
  - ... (Banzhaf et al., 1998 alone lists 178 application examples of GP up to 1997)



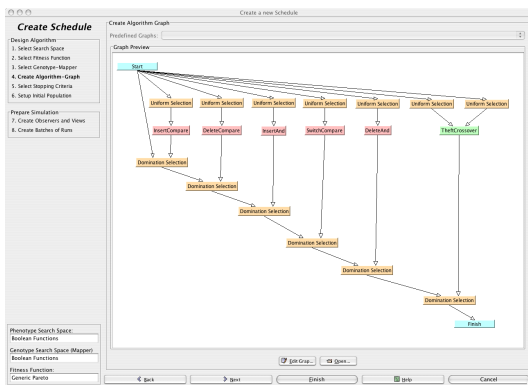
# Layout of Algorithms for Evolutionary Computation

- 1 Create an initial random population.
- 2 Evaluate the fitness values of the population.
- 3 Perform the following steps on the current generation:
  - 1 Select individuals in the population based on a selection scheme.
  - 2 Adapt the selected individuals.
  - 3 Evaluate the fitness value of the adapted individuals.
  - 4 Select adapted individuals for the next generation according to a selection scheme.
- 4 If the termination criterion is fulfilled, then output the final population. Otherwise, set the next generation as current and go to step 3.

## Graphical Representation



- Base of implementation: Free Evolutionary Algorithm Kit (FrEAK)  
<http://sourceforge.net/projects/freak427/>
- Adaptions to easily access FrEAK from R with rJava



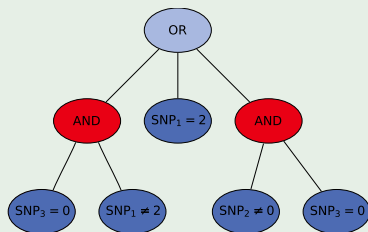
# Genetic Programming for Genetic Association Studies

- Goal: Identification of SNPs and SNP interactions leading to a higher disease risk
- Search for logic expressions as predictors

## Structure of the Individuals

- We grow trees representing logic expressions in disjunctive normal form
- Multi-valued logic expressions, because our input can take three states
  - homozygous reference (0)
  - heterozygous (1)
  - homozygous variant (2)

## Example of an individual

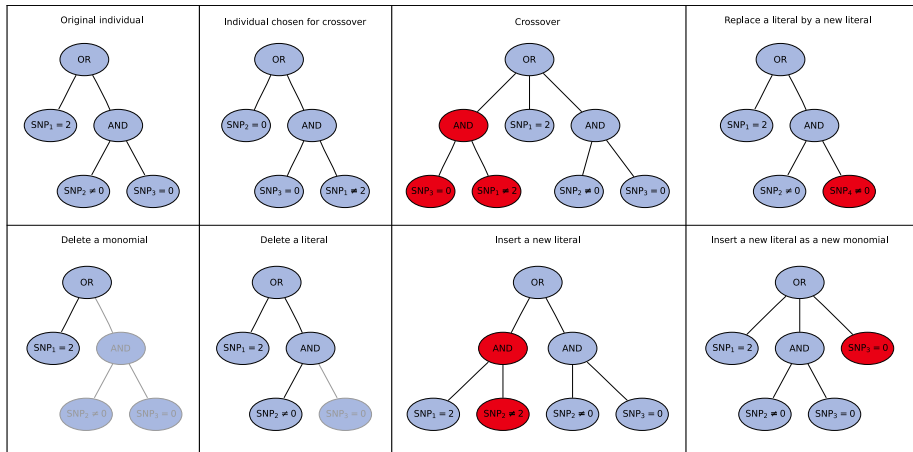


# Genetic Programming Algorithm

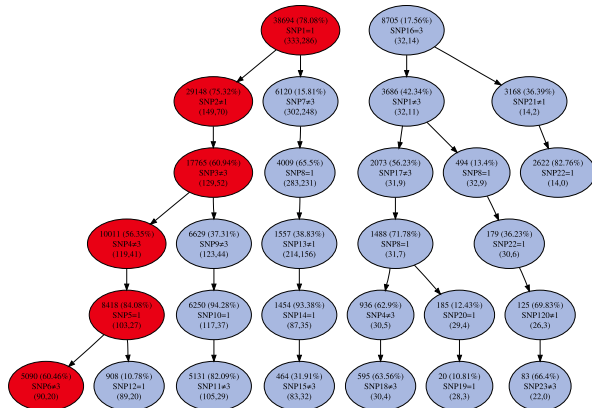
- 1 Create an initial random population consisting of two individuals.
- 2 Perform the following steps on the current generation:
  - 1 Select all individuals in the population, and reproduce them.
  - 2 Conduct mutations and crossovers to uniformly at random selected individuals in the population.
  - 3 Evaluate the fitness value of the adapted and reproduced individuals with fitness functions evaluating
    - 1 Number of predicted controls
    - 2 Number of predicted cases
    - 3 Size of the logic expression
  - 4 Select all adapted and reproduced individuals that are not pareto dominated for the next generation.
- 3 If the termination criterion is fulfilled, then output the final population. Otherwise, set the next generation as current and go to step 2.



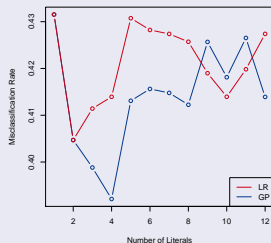
# Crossover and Mutations



# New Visualization to detect interesting interactions



## MCR of LR and the GP algorithm for a restricted number of variables



## MCR of discrimination for GENICA and HapMap data set

	GP Algorithm	Logic Regression	CART	Bagging	Random Forests
GENICA	0.392	0.405	0.429	0.457	0.450
HapMap	0.011	0.144	0.356	0.022	0.011

# Evolutionary Algorithm for Robust Regression

- Subset of observations is suitable for many robust regression methods
- Regression methods with exact fit property are NP-hard to compute

## Definition (Rousseeuw (1984))

The Least Trimmed Squares (LTS) estimator is defined by

$$\hat{\beta}_{\text{LTS}} = \arg \min_{\hat{\beta} \in \mathbf{R}^p} \sum_{i=1}^h (r^2)_{i:n} ,$$

i.e. it minimizes the sum of the  $h$  smallest squared residuals.

- Subsets of  $p$  observations suffice to uniquely determine possible solutions



# Evolutionary Algorithm

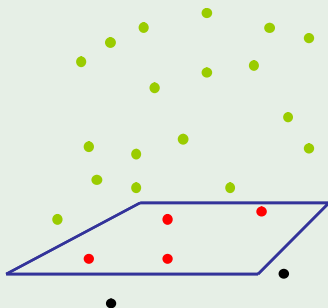
- 1 Randomly select  $p$  observations.
- 2 Perform the following steps on the current individual:
  - 1 Conduct one of the following adaptations:
    - 1 Exchange one of the selected observations with one of the not selected.
    - 2 Conduct a problem specific adaptation
  - 2 Evaluate the fitness value, i.e. the sum of squared residuals for LTS.
  - 3 Proceed with the adapted individual, if it exceeds the original individual.
- 3 If the termination criterion is fulfilled, then output the final individual. Otherwise go to step 2.



# Specific Adaption

- 1 Choose one of the currently not selected observations.
- 2 Move the hyperplane build by the  $p$  selected observations to the new observation.
- 3 Choose the  $p$  observations that are nearest to the new hyperplane.

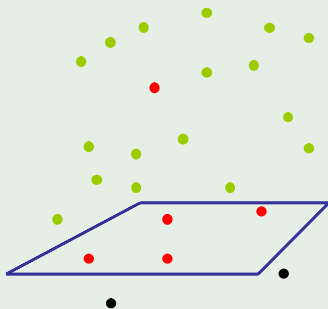
## Example



# Specific Adaption

- 1 Choose one of the currently not selected observations.
- 2 Move the hyperplane build by the  $p$  selected observations to the new observation.
- 3 Choose the  $p$  observations that are nearest to the new hyperplane.

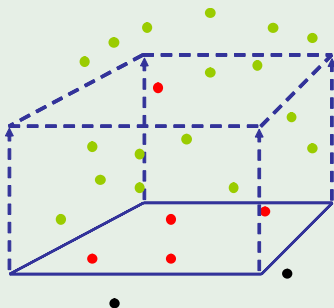
## Example



# Specific Adaption

- 1 Choose one of the currently not selected observations.
- 2 Move the hyperplane build by the  $p$  selected observations to the new observation.
- 3 Choose the  $p$  observations that are nearest to the new hyperplane.

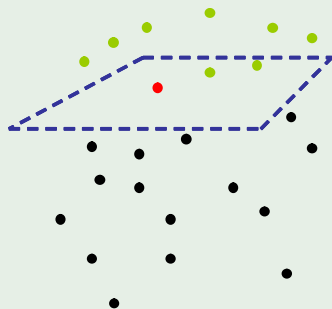
## Example



# Specific Adaption

- 1 Choose one of the currently not selected observations.
- 2 Move the hyperplane build by the  $p$  selected observations to the new observation.
- 3 Choose the  $p$  observations that are nearest to the new hyperplane.

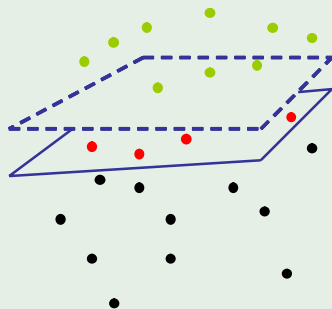
## Example



# Specific Adaption

- 1 Choose one of the currently not selected observations.
- 2 Move the hyperplane build by the  $p$  selected observations to the new observation.
- 3 Choose the  $p$  observations that are nearest to the new hyperplane.

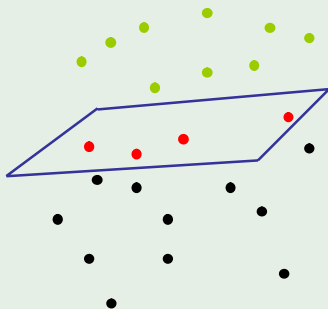
## Example



# Specific Adaption

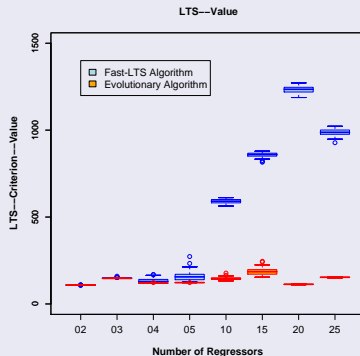
- 1 Choose one of the currently not selected observations.
- 2 Move the hyperplane build by the  $p$  selected observations to the new observation.
- 3 Choose the  $p$  observations that are nearest to the new hyperplane.

## Example



- Simulations based on methods from design of experiments
- Comparison with Fast-LTS (Rousseeuw and Van Driessen, 2005))

## Objective value for an increasing number of regressors






- Running time is substantially longer for the EA

- Brief introduction into Evolutionary Computation
- Application in Association Studies
- Application in Robust Regression

- Further possibilities in association studies, classification, clustering, . . .
- Useful in all areas, where the hitherto used search heuristics are not satisfactory

Thank you!

-  Banzhaf, W., Francone, F. D., Keller, R. E., Nordin, P., 1998. Genetic programming: an introduction: on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
-  Rousseeuw, P. J., 1984. Least median of squares regression. Journal of the American Statistical Association 79, 871–880.
-  Rousseeuw, P. J., Van Driessen, K., 2006. Computing lts regression for large data sets. Data Mining and Knowledge Discovery 12 (1), 29–45.

