

# ***Fuzzy Logic and Soft Computing: Theory and Applications***

***Dr. Piero P. Bonissone***

***GE Global Research  
Schenectady, NY, USA***

***Email: [Bonissone@crd.ge.com](mailto:Bonissone@crd.ge.com)***

**Part 5**

# Notes Outline

## 1. Soft Computing Overview

- PR, FL, NN, EA
- Model Generation
- Integrating Knowledge and Data

## 2. Background

- FL: Operations, Extension Principle, Relations, Reasoning

## 3. Prediction

- ANFIS
- Case Study: Prediction of Web Breakage

## 4. Classification

- Tree Based Classifiers: CART
- Instance Based Classifiers: Fuzzy CBR
- Case Study: Digital Underwriting

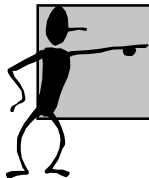
## 5. Optimization & Search

- **Evolutionary Algorithms: Search**
- **Case Study: Scheduling of Maintenance Tasks for Constellation of LEO satellites**
- **Case Study: Tuning of Fuzzy Controller for Automated Train Handling**
- **Multi-objective DM and Optimization: EMOO + Preferences**

## 6. Model Lifecycle

- Case Study: Digital Underwriting

## 7. Conclusions



# Outline

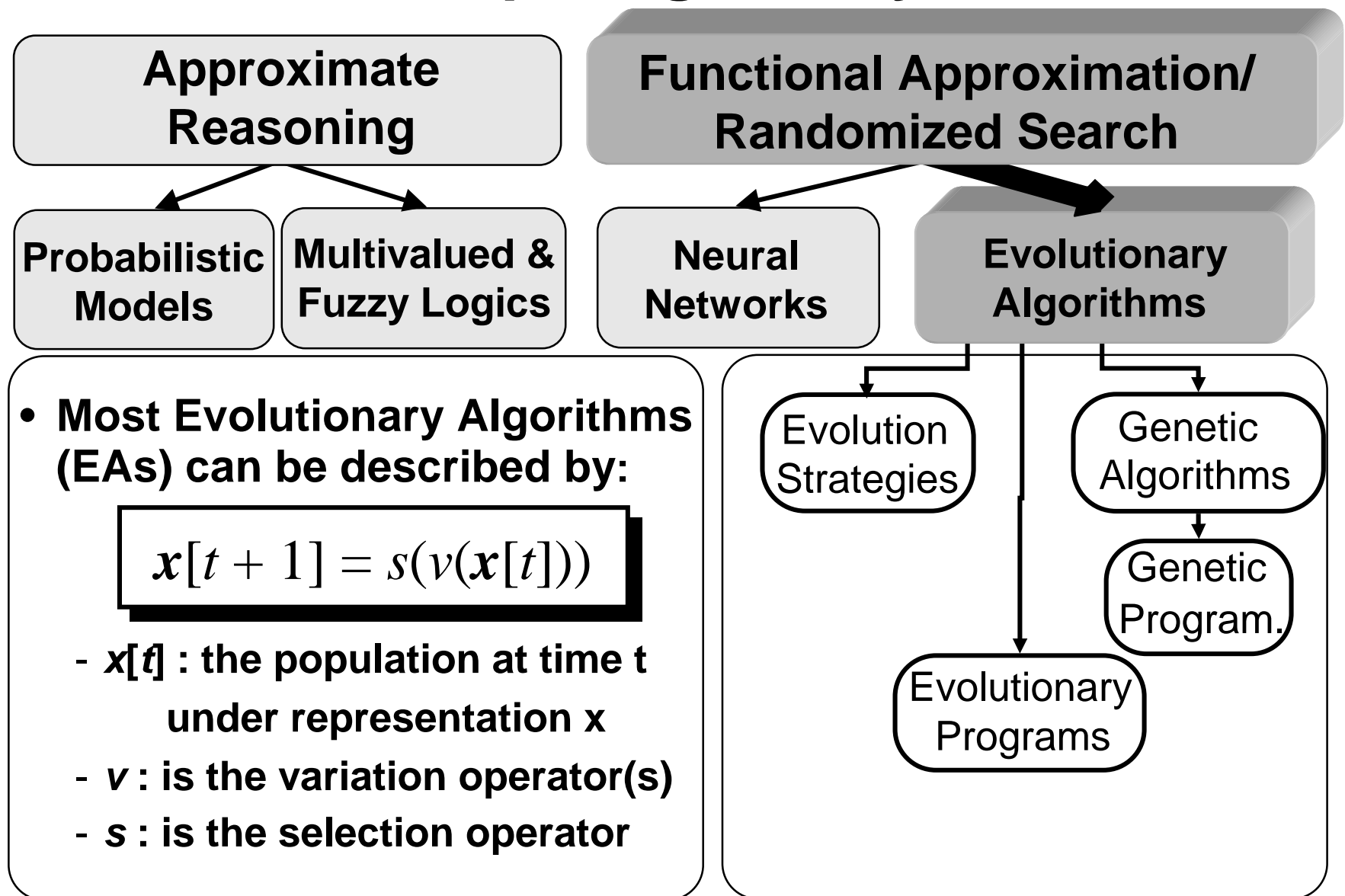


- **Evolutionary Computation**
  - Derivative Free Optimization and NFL Theorem
  - Process Cycle
  - Components (ES, EP, GA, GP)
- **Genetic Algorithms**
  - General Characteristics
  - Representation
  - Evaluation & Constraints
  - Operators
  - Components Summary
  - Functional Optimization Example
  - Evolution Stages

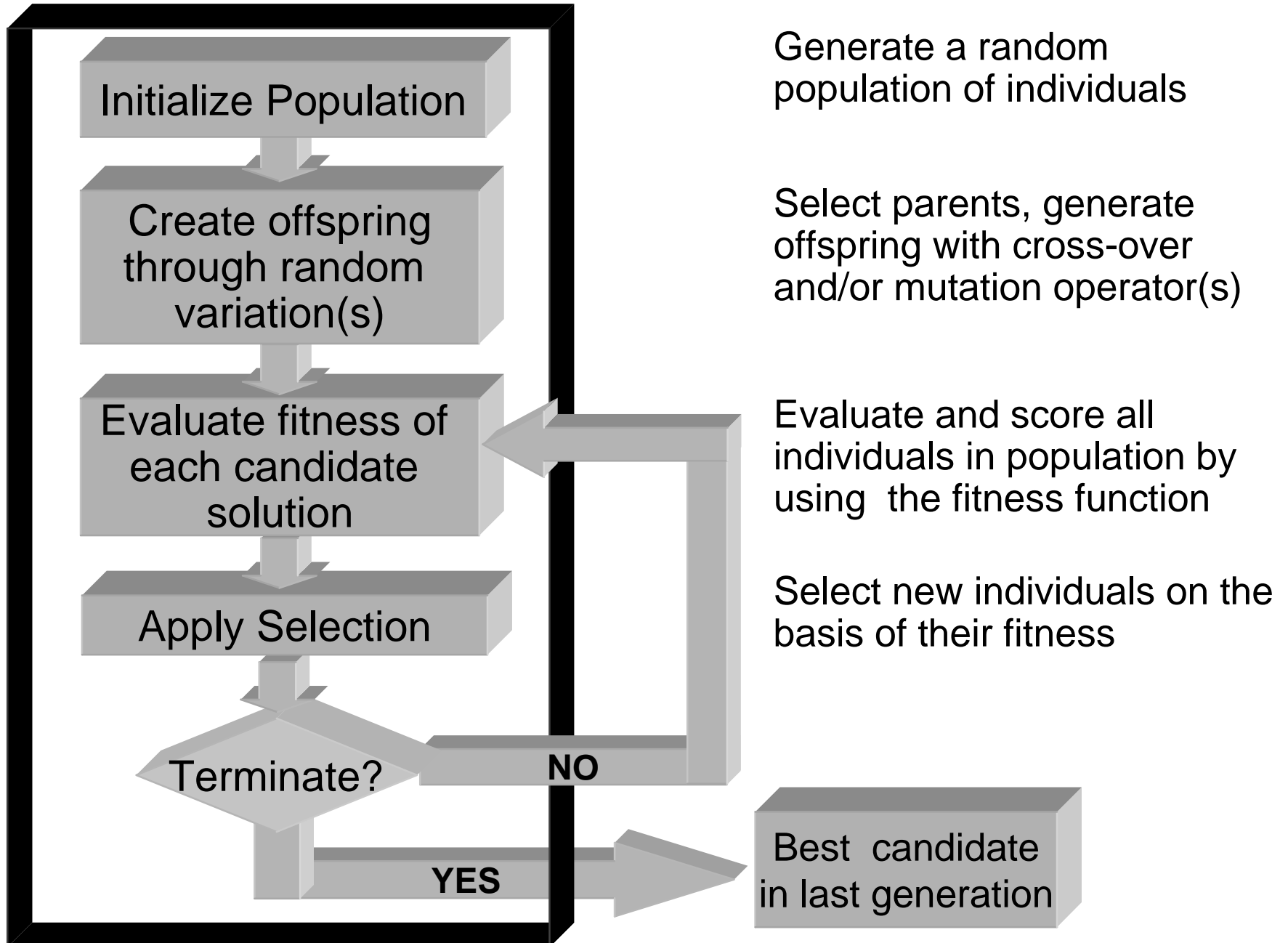
# Evolutionary Algorithms (EA)

- **EA are part of the Derivative-Free Optimization and Search Methods:**
  - Evolutionary Algorithms
  - Simulated annealing (SA)
  - Random search
  - Downhill simplex search
  - Tabu search
- **EA consists of**
  - **Evolutionary Strategies (ES)**
  - **Evolutionary Programming (EP)**
  - **Genetic Algorithms (GA)**
  - **Genetic Programming (GP)**

# Soft Computing: EA Systems



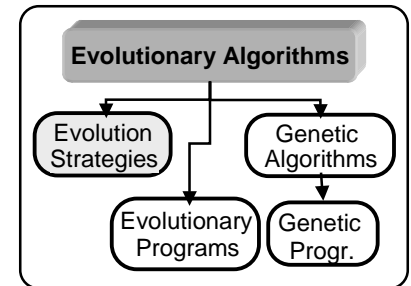
# EA Process Cycle



# Evolutionary Algorithms (EA) Characteristics

- EA exhibit an *adaptive behavior* that allows them to handle non-linear, high dimensional problems without requiring differentiability or explicit knowledge of the problem structure.
- Optimization is a side effect of their adaptation
- EA are very robust to time-varying behavior, even though they may exhibit low speed of convergence.
- Domain knowledge should be incorporated in the EA to improve their efficiency (NFL Theorem)

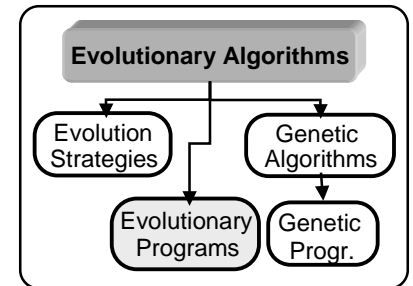
# Evolutionary Algorithms: ES



## Evolution Strategies (ES)

- Originally proposed for the optimization of continuous functions. Use *self-adjusted* parameters
- $(\mu, \lambda)$ -ES and  $(\mu + \lambda)$ -ES
  - A population of  $\mu$  parents generate  $\lambda$  offspring
  - Best  $\mu$  offspring are selected in the next generation
  - $(\mu, \lambda)$ -ES: parents are **excluded** from selection
  - $(\mu + \lambda)$ -ES: parents are **included** in selection
- Started as **(1+1)-ES** (*Reschenberg*) and evolved to  **$(\mu + \lambda)$ -ES** (*Schwefel*)
- Started with Mutation only (with individual mutation operator) and later added a recombination operator
- Focus on behavior of individuals

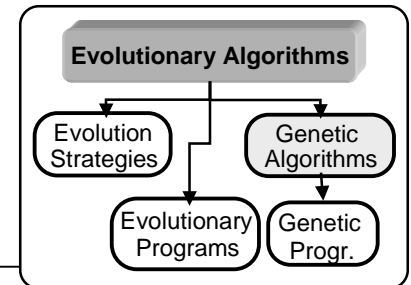
# Evolutionary Algorithms: EP



## Evolutionary Programming (EP)

- Originally proposed for sequence prediction and optimal gaming strategies
- Currently focused on continuous parameter optimization and training of NNs
- Could be considered a special case of  $(\mu + \mu)$ -ES without recombination operator
- Focus on behavior of species (hence no crossover)
- Proposed by *Larry Fogel (1963)*

# Evolutionary Algorithms: GA

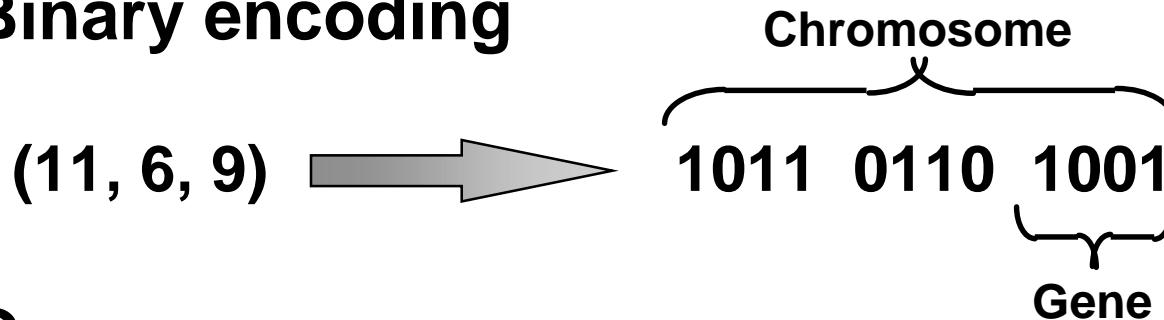


## Genetic Algorithms (GA)

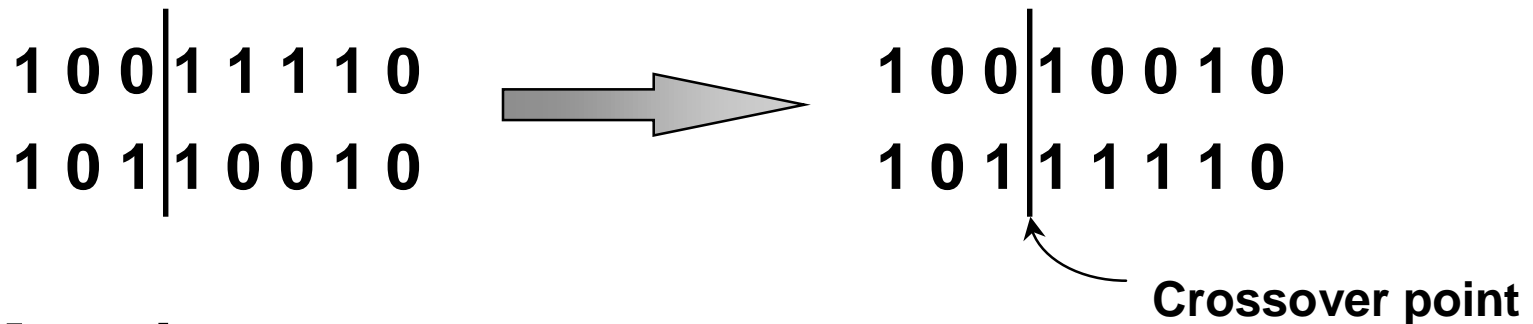
- Perform a randomized search in solution space using a genotypic rather than a phenotypic
- Each solution is encoded as a chromosome in a population (a binary, integer, or real-valued string)
  - Each string's element represents a particular feature of the solution
- The string is evaluated by a fitness function to determine the solution's quality
  - Better-fit solutions survive and produce offspring
  - Less-fit solutions are culled from the population
- Strings are evolved using mutation & recombination operators.
- New individuals created by these operators form next generation of solutions
- Started by *Holland (1962; 1975)*

# Genetic Algorithms: Example of Binary Encoding, Crossover, Mutation

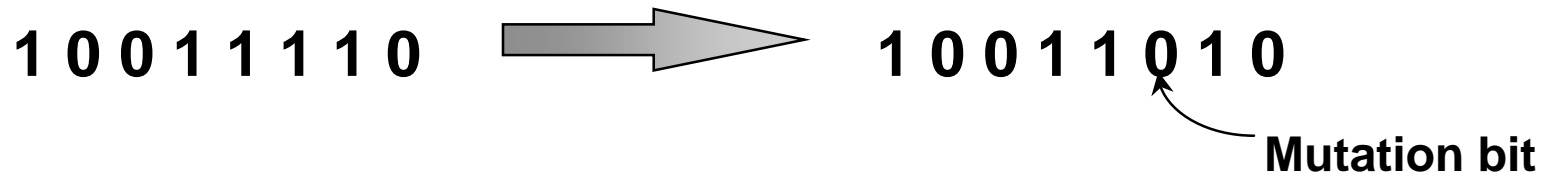
## Binary encoding



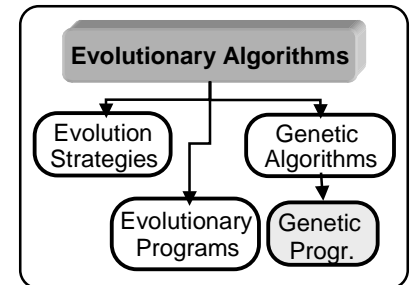
## Crossover



## Mutation



# Evolutionary Algorithms: GP

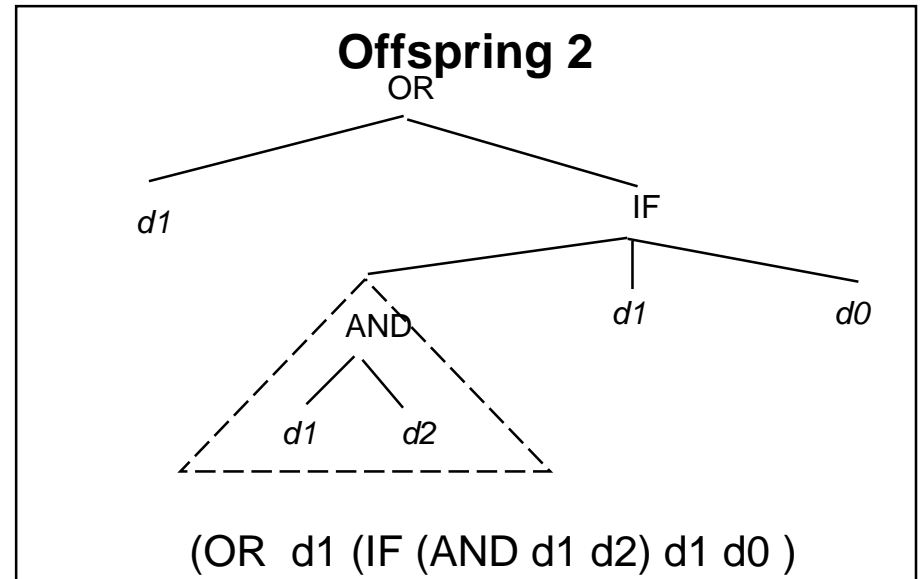
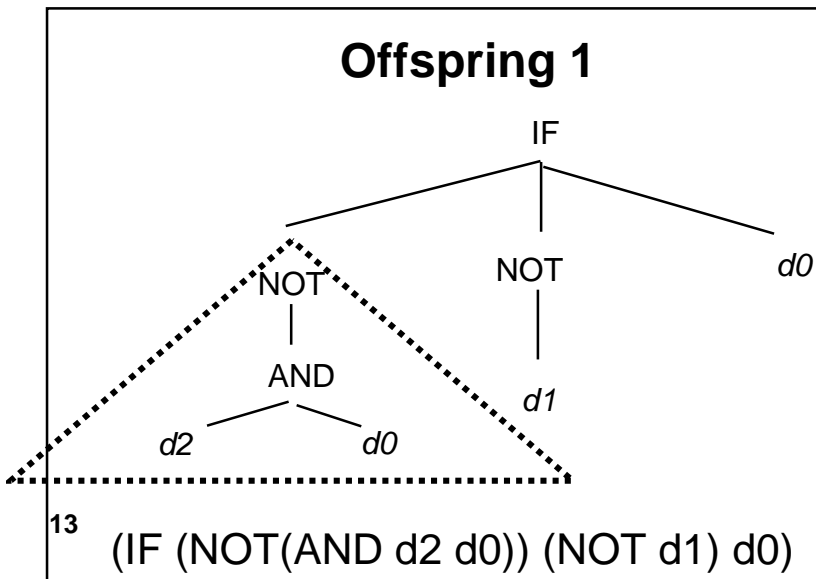
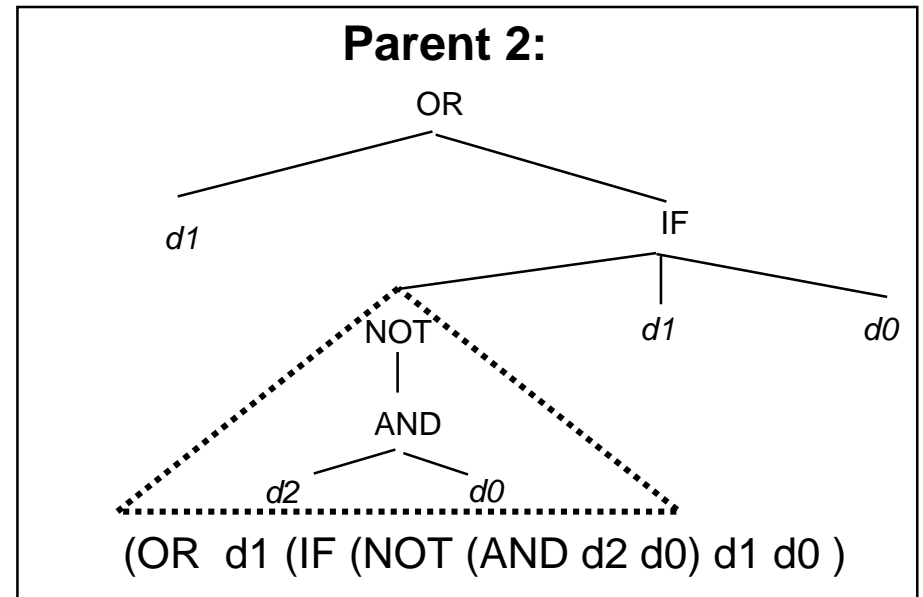
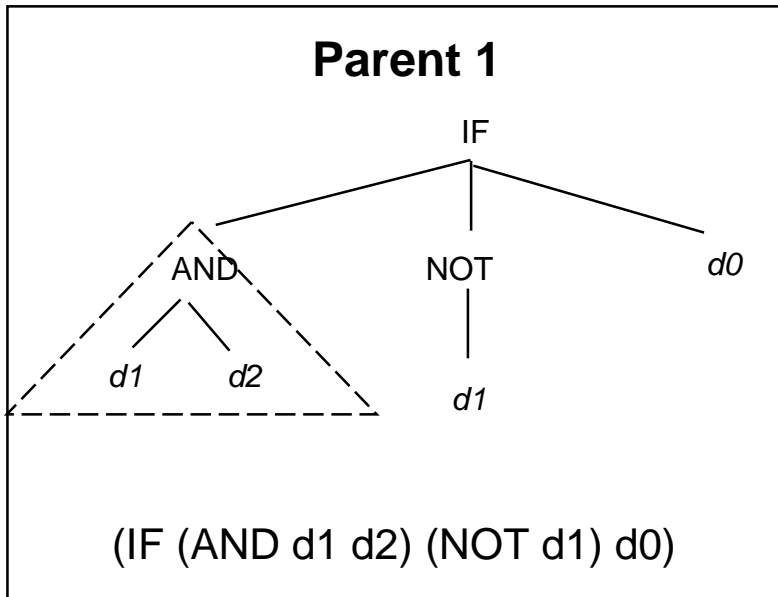


## Genetic Programming (GP)

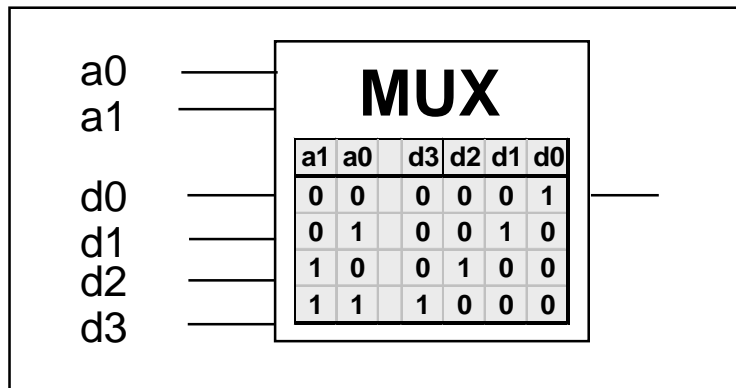
- A special case of Genetic Algorithms
  - Chromosomes have a *hierarchical* rather than *linear* structure
  - Their sizes are not predefined (dynamic length structures)
  - Individuals are usually tree-structured programs
  - Evaluate by execution
  - Syntax-preserving operators
  - Modified operators are applied to sub-trees or single nodes (see next slide)
- Proposed by *Koza (1992)*

# GP: Example of Sub-Tree Crossover

Modified operators are applied to sub-trees or single nodes.



# Example of GP: 6-Multiplexor Problem



## Functions:

- 3-arg **IF**
- 2-arg **AND, OR**
- 1-arg **NOT**

## Terminals

- Address Bits: **a0, a1**
- Data Bits: **d0, d1, d2, d3**

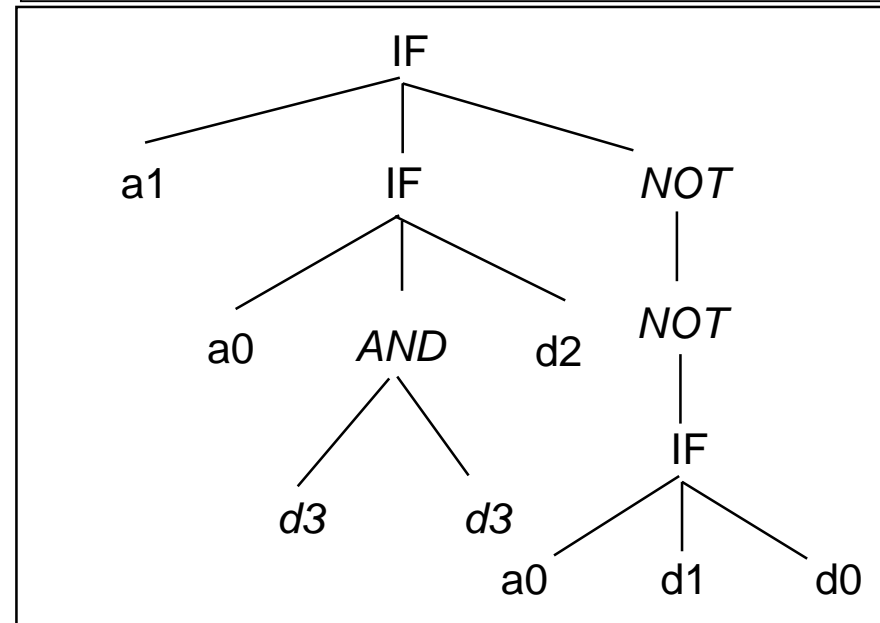
**Population** = 500

**Generation** = 201

**Fitness** = # of correct outputs (out of 64)

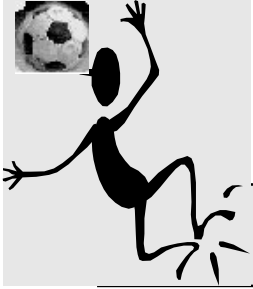
Solution after 39 generations:

- Correct but not optimal  
(see code in *yellow italics*)



# Outline

- Evolutionary Computation
  - Derivative Free Optimization and NFL Theorem
  - Process Cycle
  - Components (ES, EP, GA, GP)



## • Genetic Algorithms

- General Characteristics
- Representation
- Evaluation & Constraints
- Operators
- Components Summary
- Functional Optimization Example
- Evolution Stages

# **GAs General Characteristics**

- **No gradient information required**
- **No restrictions on structure of evaluation function (could be a black box)**
- **Resulting search is global**
- **Local Optima are avoided by hyperplane sampling in Hamming space (crossovers) plus random perturbations (mutations)**
- **Potential for massive parallelism**
- **They can be hybridized with conventional optimization methods**

# Genetic Algorithms: Description & Representation

- **What are They?**

GAs are a new programming paradigm used to solve NP-hard problems by performing a randomized search in the solution space.

- **Encoding:**

GAs *encode* the solution to a given problem in a binary (or real-valued) string. Each string's element represents a particular feature in the solution.

# Genetic Algorithms: Evaluation & Constraints

- **Evaluation:**

The string (solution) is *evaluated* by a fitness function to determine the solution's quality: good solutions survive and have off-springs, while bad solutions are discontinued.

- **Constraints:**

Solution's constraints can be modeled by *penalties* in the fitness function or encoded directly in the solution *data structures*.

# Genetic Algorithms: Operators

- **Operators:**

**To improve current solutions, the string is modified by two basic type of operators: Cross-over and Mutations.**

- **Cross-over are (sometime) deterministic operators that capture the best features of two parents and pass it to a new off-spring string.**
- **Mutations are probabilistic operators that try to introduce needed solutions features in populations of solutions that lack such feature.**

# Genetic Algorithms: Components Summary

## **1) Encoding Technique**

- Chromosome Structure

## **2) Evaluation or Fitness Function**

- The Environment

## **3) Initialization Procedure**

- Creation

## **4) Genetic Operators**

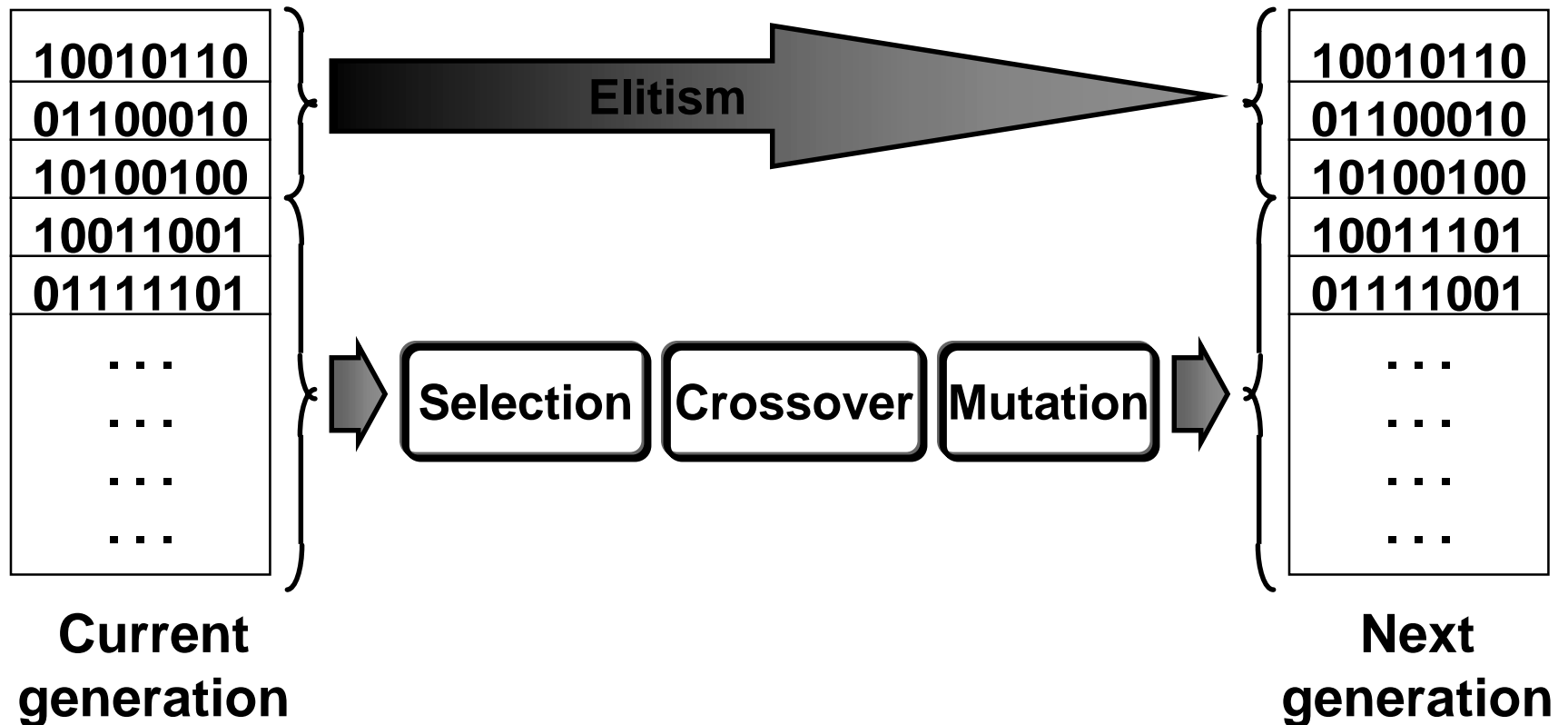
- Mutation, Recombination or Crossover

## **5) Parameter Setting**

- Practice and art

# Genetic Algorithms: Example of Process

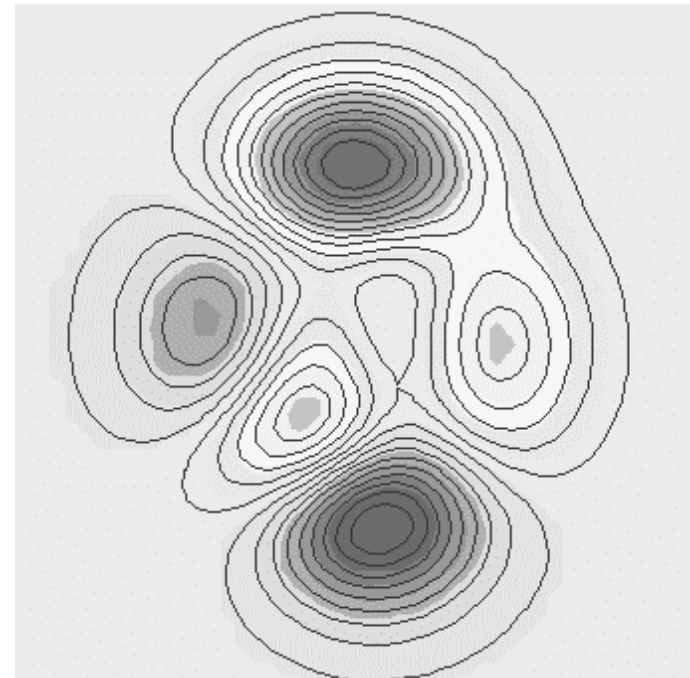
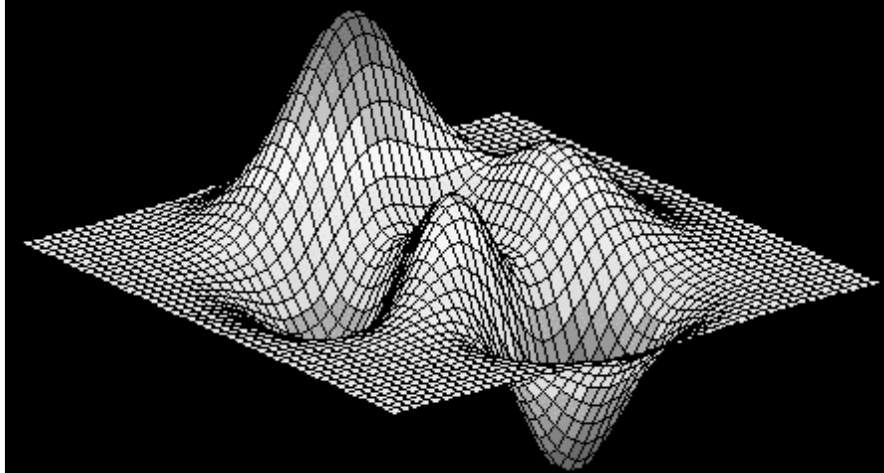
## Flowchart



# Genetic Algorithms: Functional Optimization Example

**Example: Find the max. of the “peaks” function**

$$z = f(x, y) = 3*(1-x)^2*\exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 - y^5)*\exp(-x^2-y^2) - 1/3*\exp(-(x+1)^2 - y^2).$$



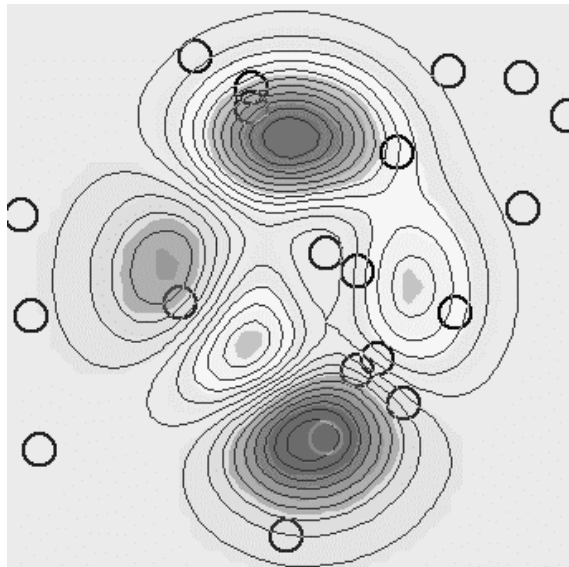
# Genetic Algorithms: Functional Optimization Example

## Derivatives of the “peaks” function

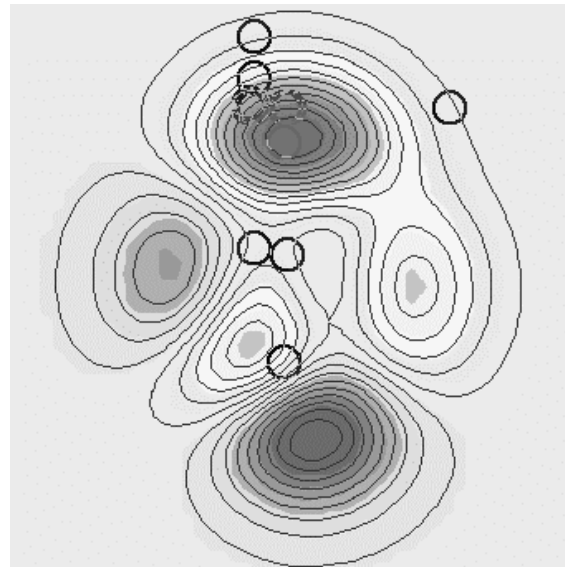
- $\frac{dz}{dx} = -6*(1-x)*\exp(-x^2-(y+1)^2) - 6*(1-x)^2*x*\exp(-x^2-(y+1)^2) - 10*(1/5-3*x^2)*\exp(-x^2-y^2) + 20*(1/5*x-x^3-y^5)*x*\exp(-x^2-y^2) - 1/3*(-2*x-2)*\exp(-(x+1)^2-y^2)$
- $\frac{dz}{dy} = 3*(1-x)^2*(-2*y-2)*\exp(-x^2-(y+1)^2) + 50*y^4*\exp(-x^2-y^2) + 20*(1/5*x-x^3-y^5)*y*\exp(-x^2-y^2) + 2/3*y*\exp(-(x+1)^2-y^2)$
- $\frac{d(\frac{dz}{dx})}{dx} = 36*x*\exp(-x^2-(y+1)^2) - 18*x^2*\exp(-x^2-(y+1)^2) - 24*x^3*\exp(-x^2-(y+1)^2) + 12*x^4*\exp(-x^2-(y+1)^2) + 72*x*\exp(-x^2-y^2) - 148*x^3*\exp(-x^2-y^2) - 20*y^5*\exp(-x^2-y^2) + 40*x^5*\exp(-x^2-y^2) + 40*x^2*\exp(-x^2-y^2)*y^5 - 2/3*\exp(-(x+1)^2-y^2) - 4/3*\exp(-(x+1)^2-y^2)*x^2 - 8/3*\exp(-(x+1)^2-y^2)*x$
- $\frac{d(\frac{dz}{dy})}{dy} = -6*(1-x)^2*\exp(-x^2-(y+1)^2) + 3*(1-x)^2*(-2*y-2)^2*\exp(-x^2-(y+1)^2) + 200*y^3*\exp(-x^2-y^2) - 200*y^5*\exp(-x^2-y^2) + 20*(1/5*x-x^3-y^5)*\exp(-x^2-y^2) - 40*(1/5*x-x^3-y^5)*y^2*\exp(-x^2-y^2) + 2/3*\exp(-(x+1)^2-y^2) - 4/3*y^2*\exp(-(x+1)^2-y^2)$

# Genetic Algorithms: Functional Optimization Example

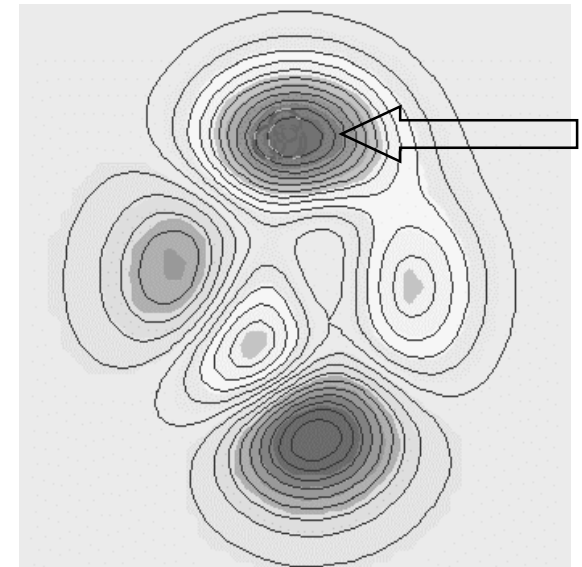
**GA process:**



**Initial population**



**5th generation**

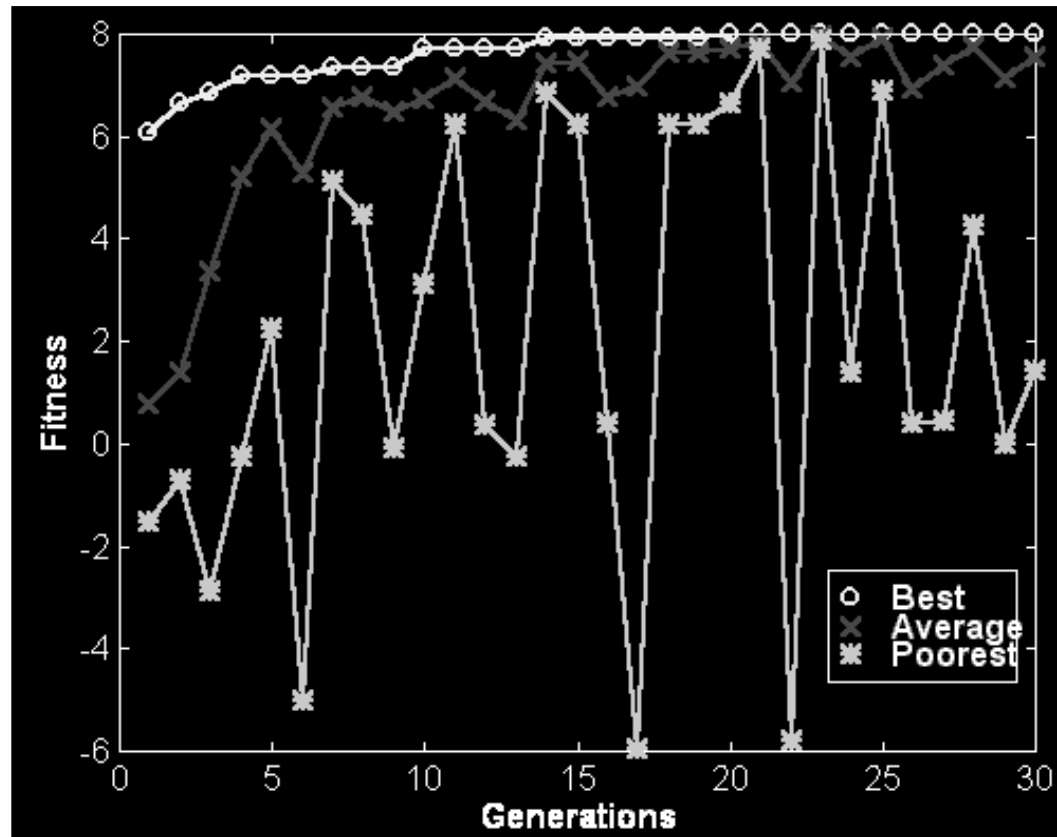


**10th generation**

**Not necessarily the Global Optimum, but region(s) of *good* solutions**

# GAs Performance Measurements

## Performance profile



Best: Monotonically non-decreasing (with elitist)  
Average: Improves over time  
Ratio: Best/Average is a measure of convergence

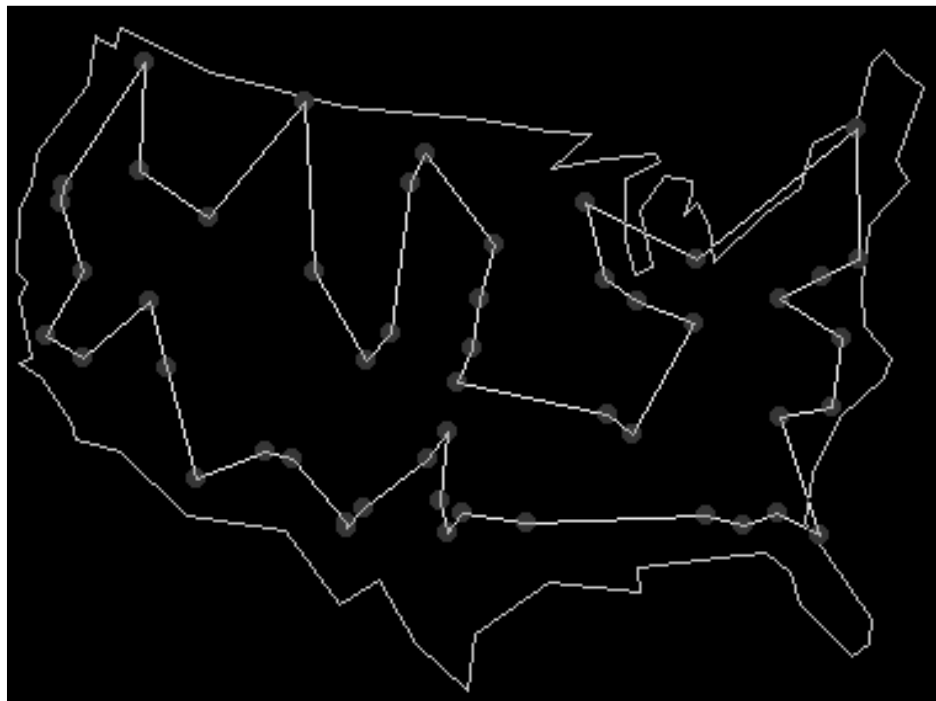
# GAs Evolution Stages

- **Exploration** : GAs generate enough candidates to have enough diversity (building blocks) for good solution
  - Increased Population Size
  - More forgiving parents selection
  - Stronger mutation
- **Transition** : GAs refine their performance by performing a down-selection & managing their resources during exploitation
- **Exploitation** : GAs refine their solutions by focusing on good individuals and enforcing more stringent solution quality requirements
  - Decreased Population Size
  - Stricter parents selection
  - Weaker mutation

# Typical EA Applications: Scheduling

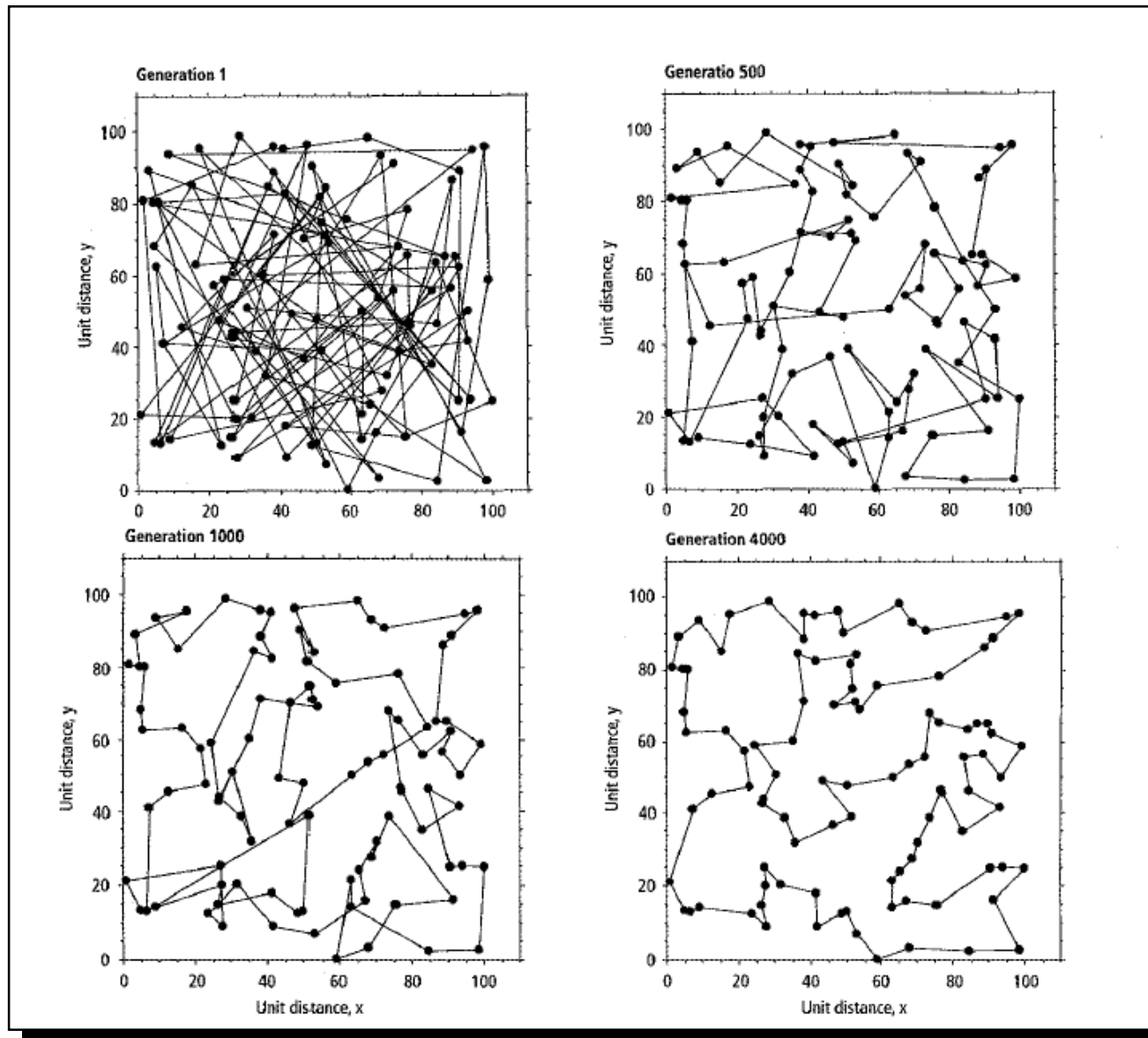
## Example: Travel Salesperson Problem (TSP)

How to transverse  $n$  cities once and only once with a minimal total distance?



# TSP Solutions

GA Using inversion, switching, and crossover with repair



# Additional Resources

- **ES**

- <http://lautaro.fb10.tu-berlin.de/intseit2/xs2evost.html>
- <http://citeseer.nj.nec.com/back91survey.html>

- **EP**

- <http://www.natural-selection.com/eps/>
- [http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/Q1\\_2.htm](http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/www/Q1_2.htm)

- **GA**

- <http://www.aic.nrl.navy.mil/galist/>
- <http://www-illigal.ge.uiuc.edu:8080/>
- <http://www.scs.carleton.ca/~csgs/resources/gaal.html>

- **GP**

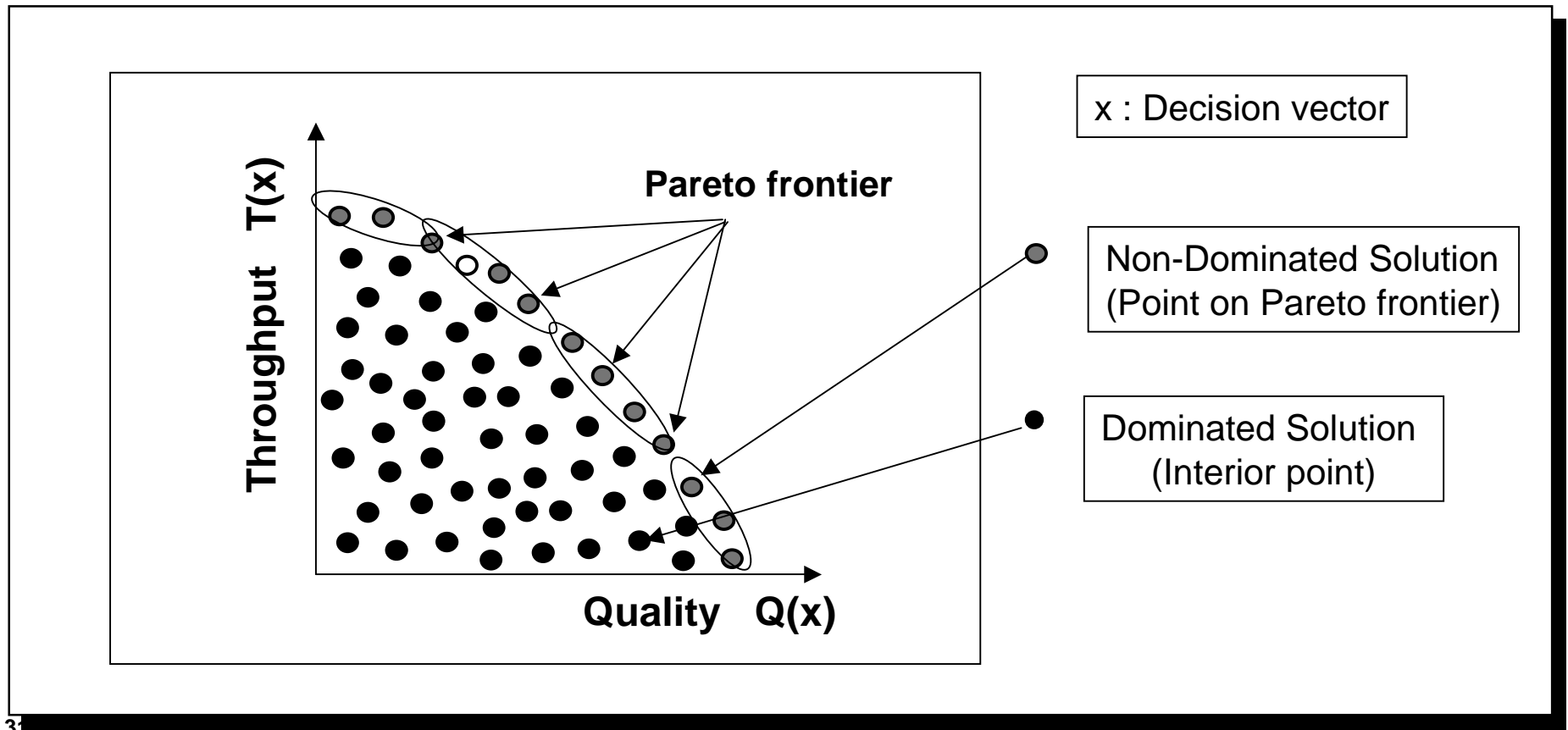
- <http://www.genetic-programming.org/>
- <http://www.geneticprogramming.com/>
- <http://www.cs.ucl.ac.uk/research/genprog/>

# Evolutionary Multi-Objective Optimization (MOO)

- **Real-world problems are characterized by multiple measures of performance, which need to be optimized, or at least satisfied simultaneously.**
- **Objectives and constraints are expressed by equalities and inequalities**
- **EMOO used in problems that have more than one variable to account for. For instance:**
  - Design of turbine engines:
    - Maximize fuel efficiency,
    - Minimize cost,
    - Minimize manufacturing time
    - Maximize reliability, etc.
  - Troop Employment Plan:
    - Minimize friendly casualties,
    - Minimize deployment time,
    - Maximize controlled area,
    - Maximize enemy's losses, etc.

# EMOO: Vector-Valued Fitness Function

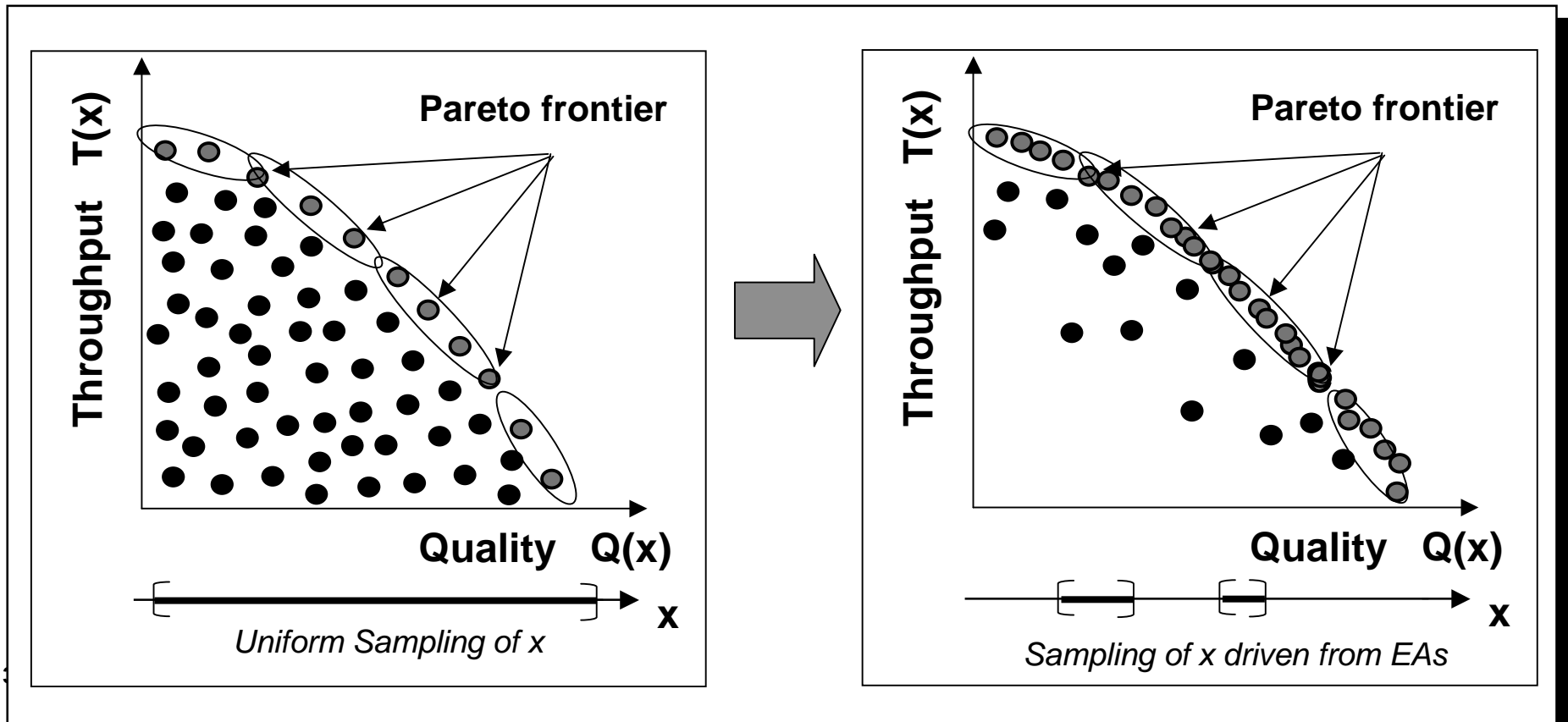
- There is no longer one optimal point, but an infinite number of **Non-Dominated points that form the Pareto Frontier**
  - Example: Identify the Pareto frontier for the problem [Max  $Q(x)$ , Max  $T(x)$ ]



# EMOO: A Possible Quality Measure

One way to evaluate the quality of the search method is to determine how well the Pareto Surface was sampled during the evolution, compared with the total number of points evaluated.

We could use the following ratio as a quality metric for EMOO:



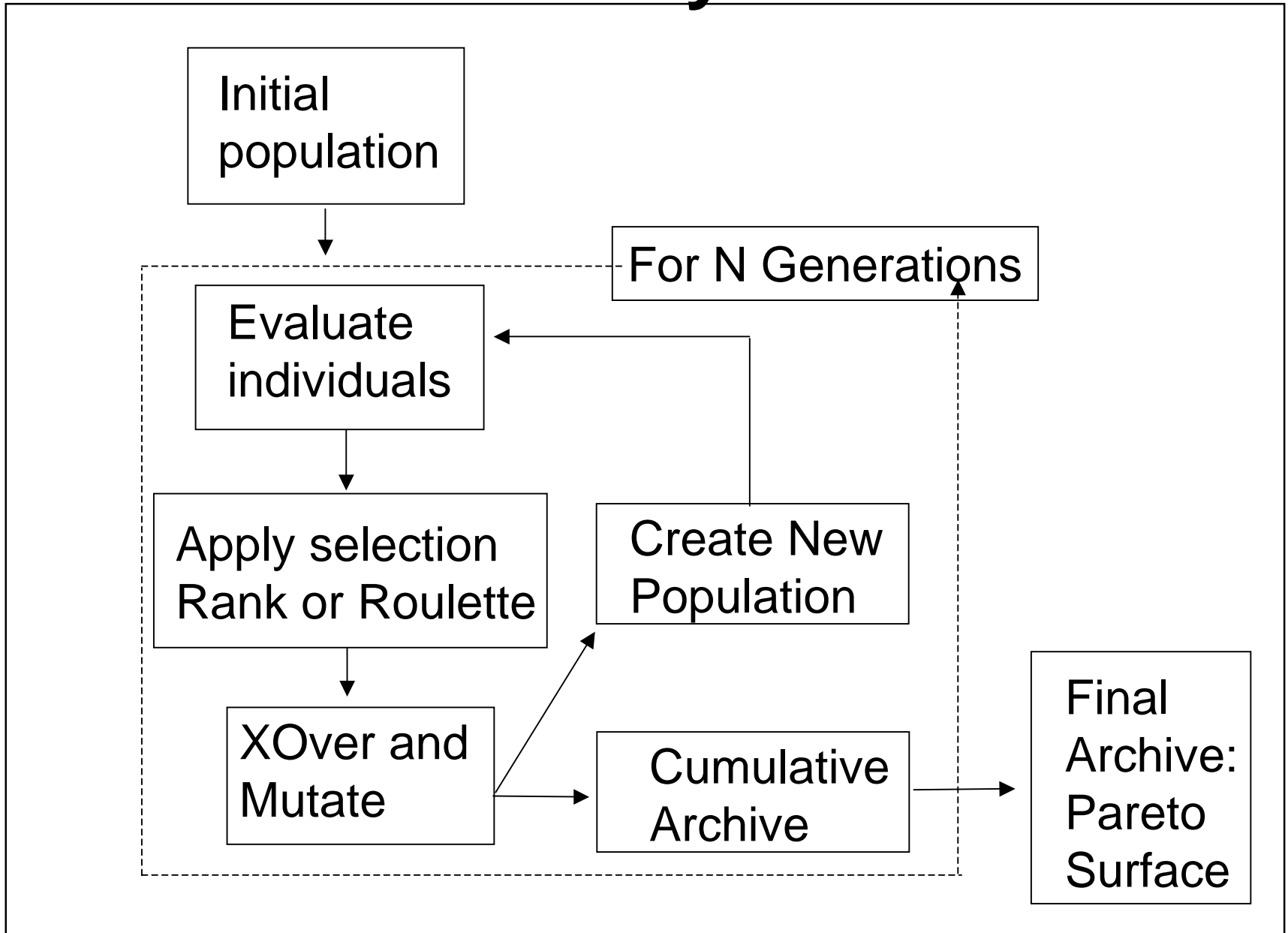
# EMOO: A Review

- **Scalar-Valued Fitness Functions**
  - Aggregated fitness function focuses on one tradeoff point in frontier
    - Use of *variable weights* in multi-criteria aggregating function
    - Dynamic Control of weights over time
- **Multi-Valued Fitness Function: Generation of Pareto Frontier**
  - **VE-GA**: Vector Evaluated GA (*Shaffer & Grefenstette 1985*)
  - **VOW-GA**: Variable Objective Weighting GA (*Hajela & Lin 1992*)
  - **MO-GA**: Multiobjective Optimization GA (*Fonseca & Fleming 1994*)
  - **NP-GA**: Niche Pareto GA (*Horn, Nafpíolitis, Goldberg, 1994*)
  - **NS-GA**: Non Dominated Sorting GA (*Srinivas & Deb, 1995*)
  - **RW-GA**: Random Weights GA (*Ishibushi & Murata, 1998*)
  - **Multi-Sexual GA** (*Lis & Elben 1997*)

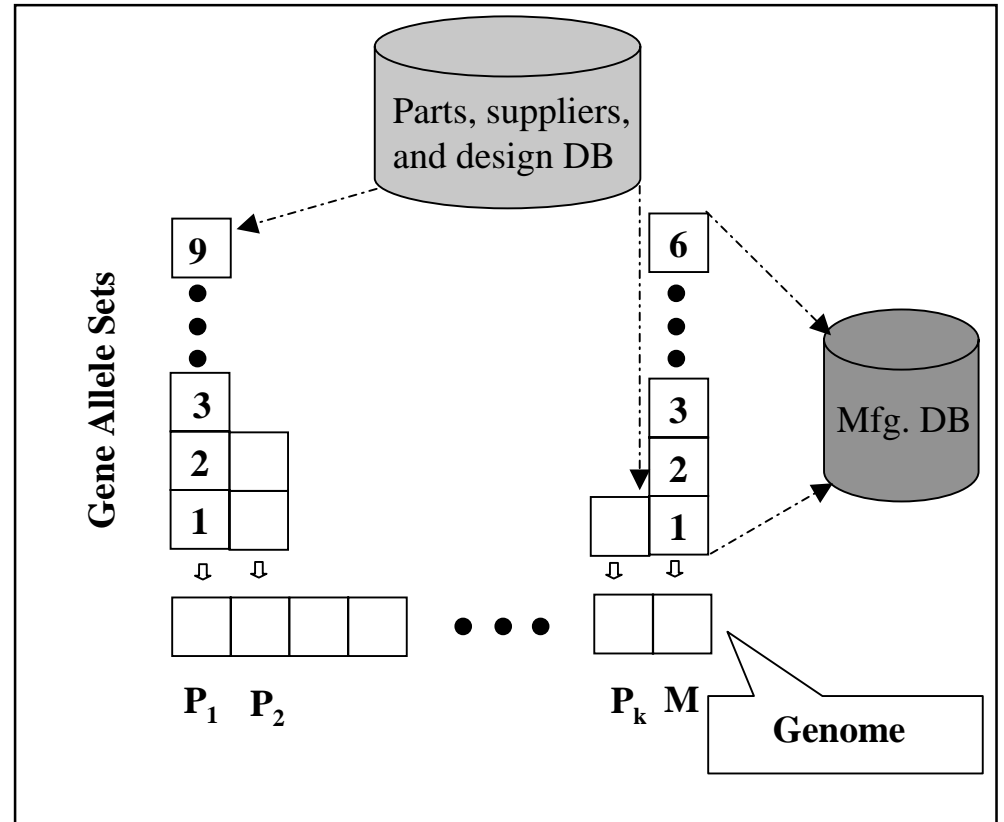
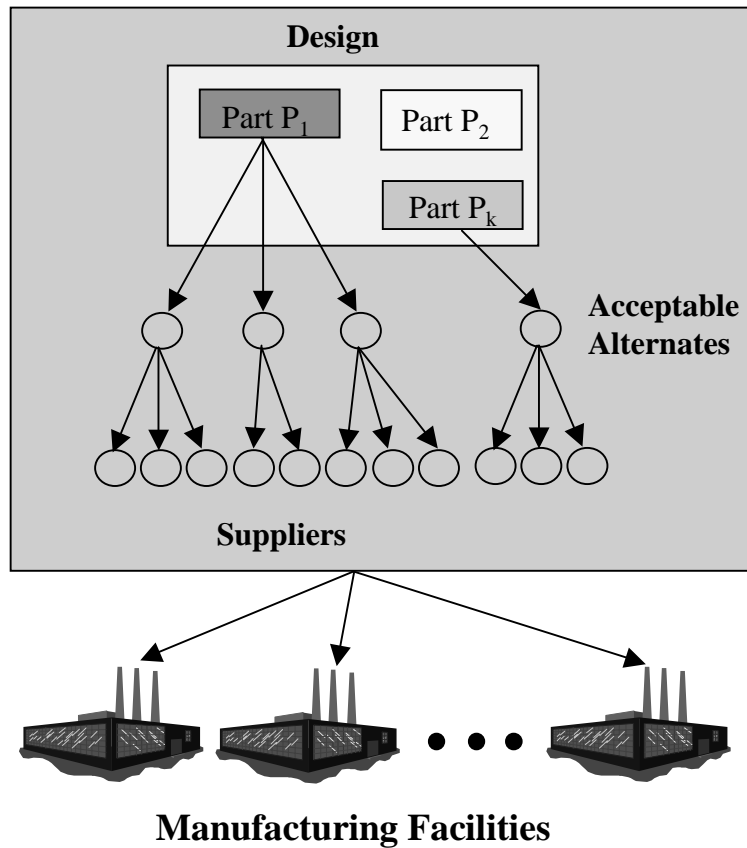
# Multi-Sexual Genetic Algorithms (MSGGA)

- **MSGGA's Three Main Ideas:**
  - 1) Provide each individual with an additional feature: a sex or gender tag
  - 2) Map optimization Criteria to sexes by a 1-to-1 mapping and evaluate individuals by the optimization criteria belonging to their sex
  - 3) Use *multi-parent crossover* for recombination (sexual reproduction). This requires one parent from each sex

# MSGGA Cycle



# Flexible Manufacturing Problem



$$\min_{i,j} \{C_{ij}^T, T_{ij}^T\}$$

**Object-level Optimization Problem, minimizing  
Total Cost and Total Time**

# Planning Problem Formulation

## Three assignment problems: $A_1$ , $A_2$ , $A_3$

- $A_1$  is the assignment of **parts** (from a parts library) to a design that satisfies a predetermined functional specification. Multiple designs that satisfy the functional specification are possible.
- $A_2$  is the assignment of **suppliers** (from a list of available suppliers) who will supply the parts in a design, and the assignment problem
- $A_3$  is the assignment of designs to available **manufacturing resources**.

# Total Cost of Design $D_i$ assigned to Manufacturing $M_j$

**Total Cost =**

$$C_{ij}^T = \frac{C_{Mij}}{f_{TR}^{M_j}(D_i)} + C_P$$

**is a function of Total Manufacturing Cost per hour, Throughput and Procurement Cost.**

where:

$$C_{Mij} = f_{MVC}^{M_j} \left( f_{MD}^{M_j} \left( f_{PC}(D_i), f_{ATY}(D_i) \right) \right) + f_{MEC}(M_j).$$

**Total manufacturing cost per hour** for a design  $D_i$  manufactured at manufacturing line  $M_j$  is a function of *variable* costs and *fixed* costs

- Variable costs depend on the degree of manufacturing difficulty of the design, which is determined by the design percentage of core and by the mounting type.
- Fixed costs are a function of the manufacturing site only.

$f_{TR}^{M_j}(D_i)$  **Throughput** on a particular manufacturing line  $M_j$  is a function of the type and number of auxiliary components in a design  $D_i$

<sup>38</sup>  $C_P$  **Procurement Cost**

## Total Delay Time of Design $D_i$ assigned to Manufacturing $M_j$

$$\text{Total Delay} = T_{ij}^T = f_{TS}^{M_j} \left( f_{MD}^{M_j} (f_{PC}(D_i), f_{ATY}(D_i)) \right) + f_{TD}(f_{PC}(D_i)) + T_P$$

is a function of the setup time, design time overhead, and maximum procurement time.

where:

$$f_{TS}^{M_j} \left( f_{MD}^{M_j} (f_{PC}(D_i), f_{ATY}(D_i)) \right)$$

**Design Time Overhead** is a function of the manufacturing difficulty of the design  $D_i$ , which is determined by the design percentage of core and by the mounting type.

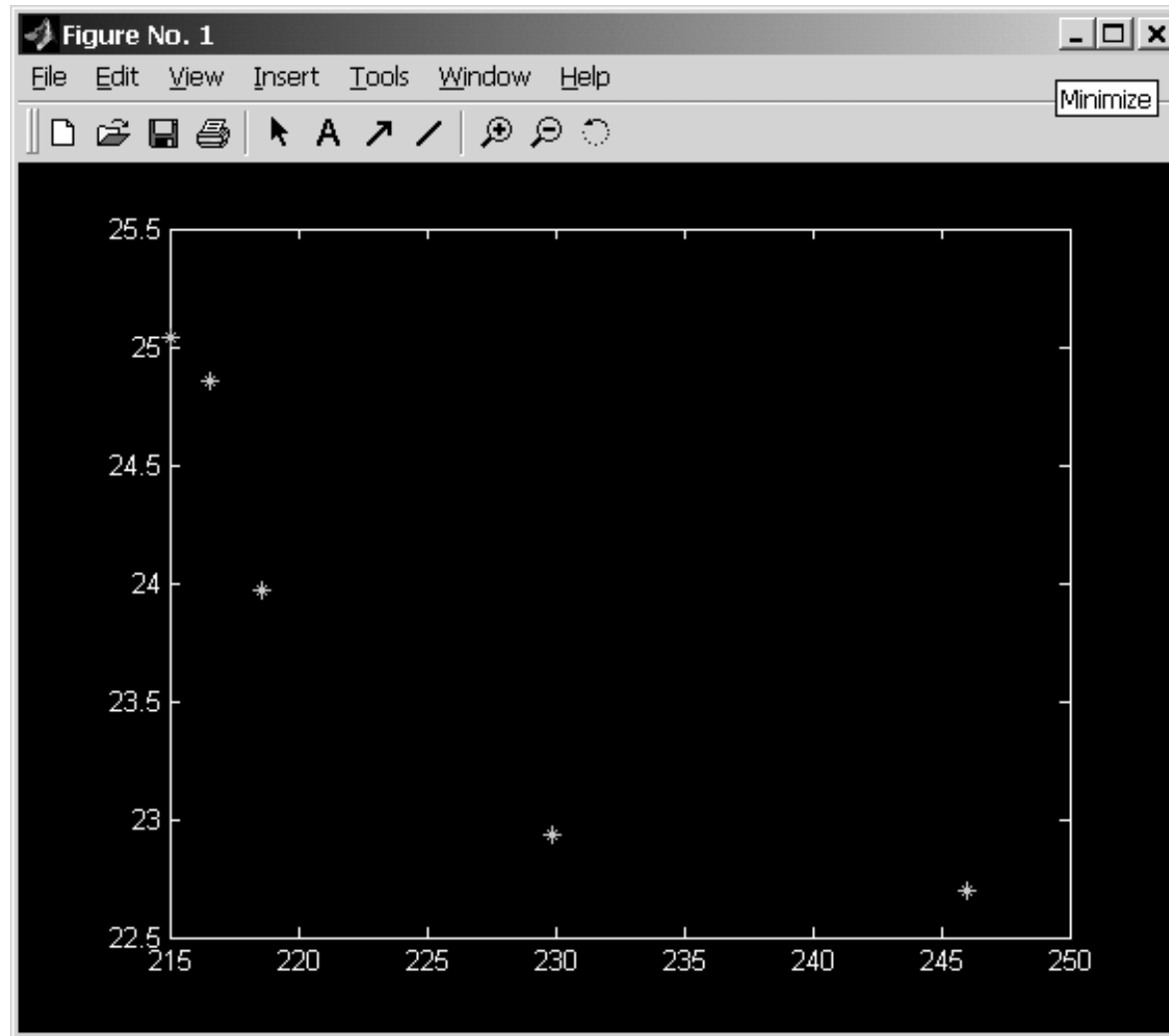
$$f_{TD}(f_{PC}(D_i))$$

**Design Time Overhead** is a function of its percentage of core

$$T_P$$

**Maximum Procurement Delay**

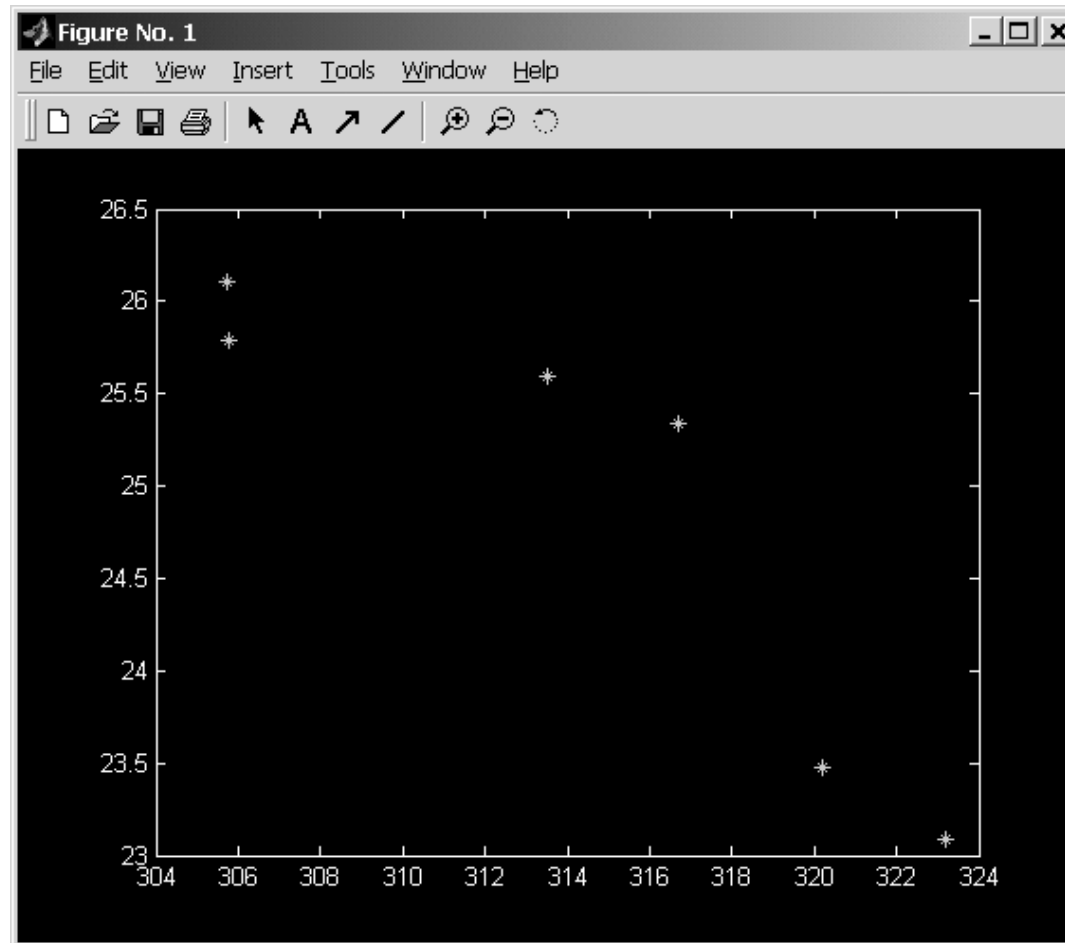
# Results of Flex. Manufacturing Problem $O(10^{11})$



Found 5 Non-dominated solutions (out of  $4 \times 10^{11}$ )

**18 different configurations** had their cost-time values mapped into one of these five different tradeoff values

# Results of Flex. Manufacturing Problem $O(10^{15})$



Found 6 Non-dominated solutions (out of  $2.7 \times 10^{15}$ )

An average of **150 different configurations** had their cost-time values mapped into one of these six different tradeoff values

# Conclusions

- We presented novel extensions to the work of Lis & Eiben.
- We found that, with a fixed population size **gender selection**, the parameter that was earlier expected to have a significant effect on MSEA performance **contributes very little** to the performance of the algorithm on the benchmark functions.
- We found that the most significant influencers of MSEA performance are the **selection** and **crossover** operators.
- The algorithm versions that yield the best performance on the test functions were used in experiments on the design and manufacturing planning problem.

The MSEA identified only a small number of non-dominated points even when the decision space was very large -  **$O(10^{15})$**

- This is due to the nature of the planning problem whereby many different alternative design and manufacturing plans yield the same costs and cycle times.
- The MSEA was able to identify several **equally good non-dominated** planning alternatives
- This demonstrates the ability of the algorithm to adequately probe solutions on the iso-contours of the cost and time fitness spaces

# Future Work

- **Possible extensions to enhance the MSEA:**

- Control population size of each gender. This would allow the MSEA to select from more individuals of a certain gender.
  - By adding these extra individuals, we will be increasing the probe capability for the corresponding objective and increasing the likelihood of finding stronger individuals along that dimension.
- The crossover operation could also be based on including more than one individual of the same gender.
  - This approach would not increase the number of individuals in the population, and would achieve the effect of enhanced probe capability.

- **Possibilities extensions of experiments in flexible manufacturing**

- Extending the problem complexity even further to explore number of points in Pareto surface
- Experiment with the parameters of the MSEA to find out the optimal settings for this problem.
  - Crossover type and gender selection type would be good candidates for variables that may have an impact on the performance of the algorithm.