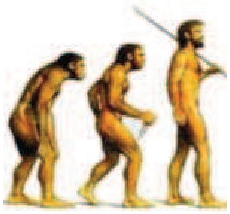
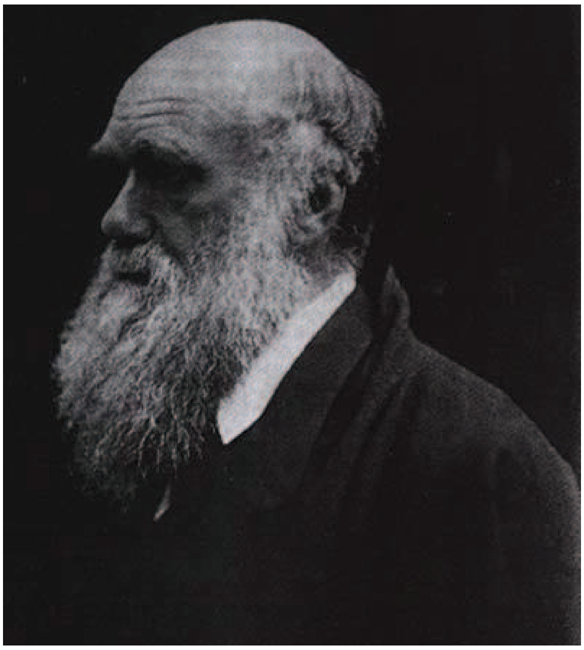
Biologically Inspired Computation

- Evolutionary computation (e.g., genetic algorithms)
- Immune-system-inspired computer/network security
- Ant-colony optimization
- Swarm intelligence
- Neural networks
- Molecular (DNA) computation



Genetic Algorithms: A Tutorial



Charles Darwin
1809–1882

> *"Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime."*
>
> - David E. Goldberg
> *Computer Design*, May 1995

The Genetic Algorithm

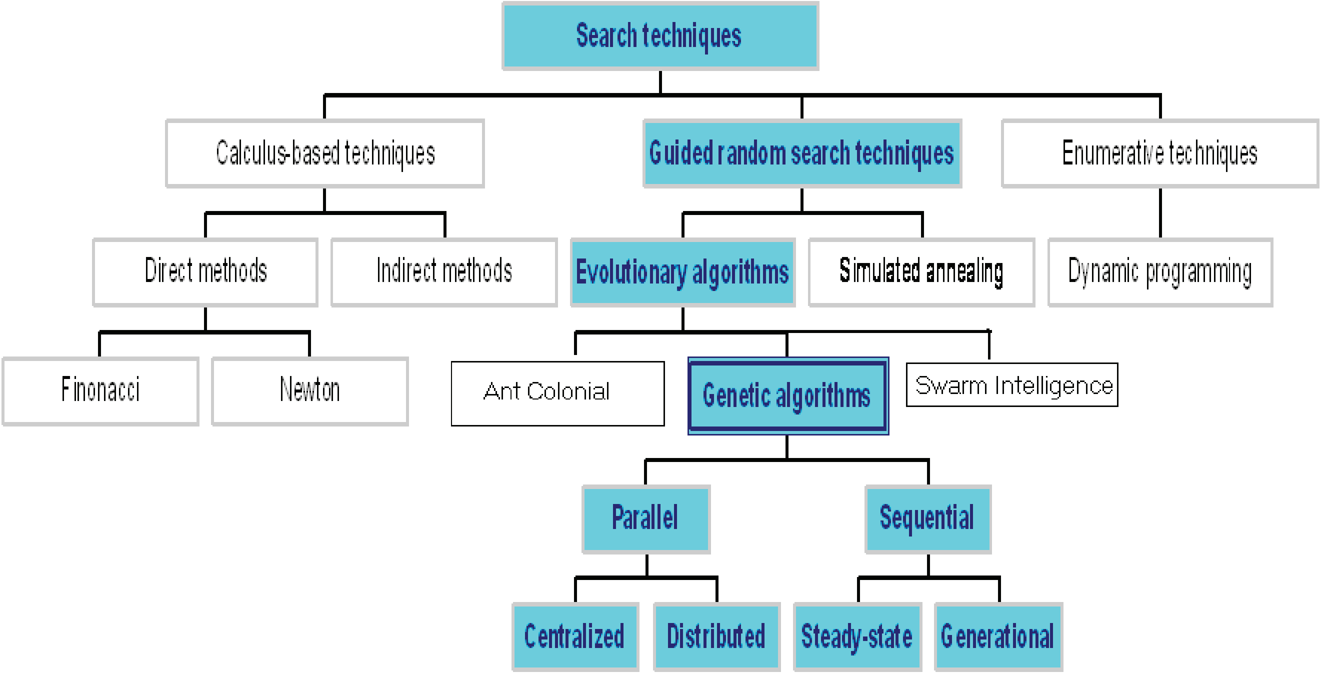
- Directed search algorithms based on the “mechanics” of biological evolution
- Developed by **John Holland**, University of Michigan (1970’s)
  - To understand the adaptive processes of natural systems
  - To design artificial systems software that retains the robustness of natural systems
- Provide efficient, effective techniques for optimization and machine learning applications
- Widely-used today in business, scientific and engineering processes.

Evolutionary Computation

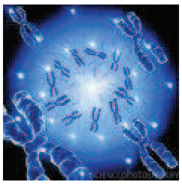
A collection of computational methods inspired by biological evolution:

- A population of candidate solutions evolves over time, with the fittest at each generation contributing the most offspring to the next generation
- Offspring are produced via crossover between parents, along with random mutations and other “genetic” operations.

Classes of Search Techniques



Components of a GA

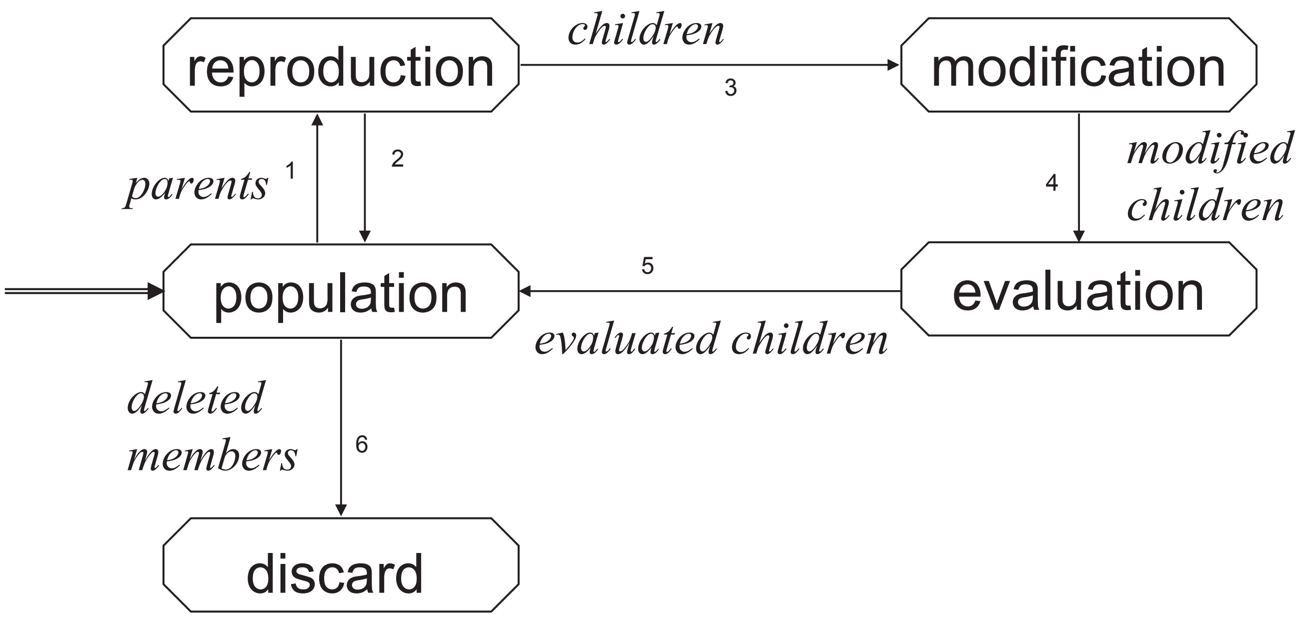


- Basic genetic population *(gene,chromosome)*
- Initialization procedure *(creation)*
- Selection of parents *(reproduction)*
- Evaluation function *(environment)*
- Genetic operators *(mutation, recombination)*
- Parameter settings *(practice and art)*

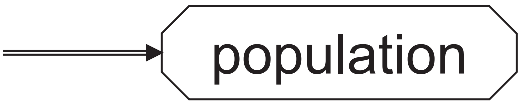
Simple Genetic Algorithm

```
{
    initialize node population;
    evaluate node population;
    while (TerminationCriteriaNotSatisfied)
    {
        select parent nodes for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

The GA Cycle of Reproduction



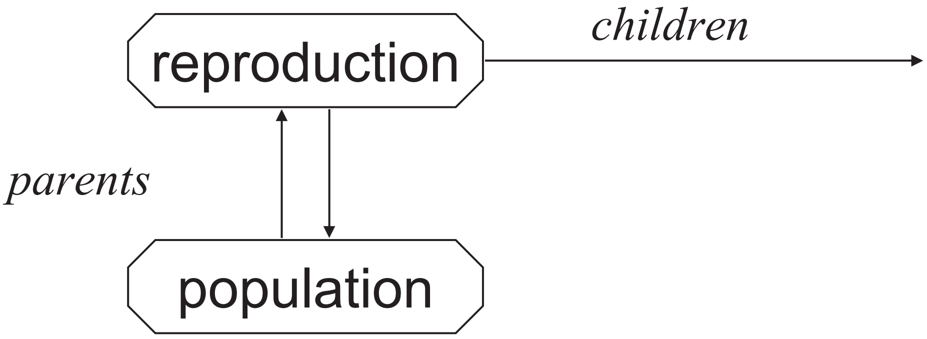
Population



Chromosomes could be:

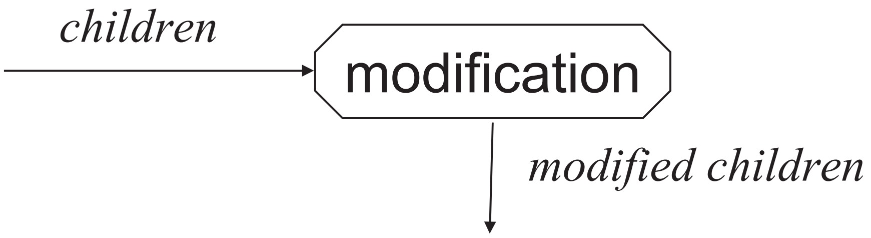
♦ Bit strings	(0101 ... 1100)
♦ Real numbers	(43.2 -33.1 ... 0.0 89.2)
♦ Any element	(E11 E3 E7 ... E1 E15)
♦ Lists of rules	(R1 R2 R3 ... R22 R23)
♦ Program elements	(genetic programming)
♦ ... any data structure ...	

Reproduction



Parents are selected at random with selection chances "biased" in relation to evaluations.

Chromosome Modification



- Modifications are stochastically triggered
- Operator types are:
  - Mutation (alter, change)
  - Crossover (recombination)

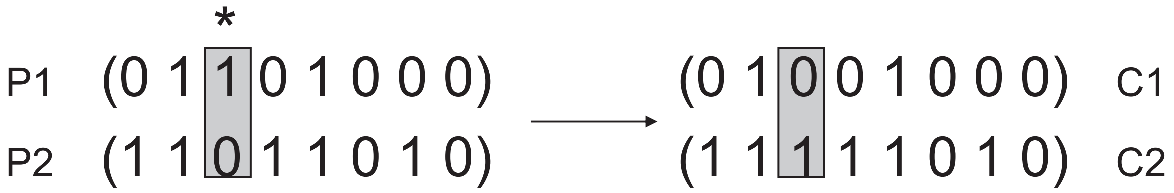
Mutation: Local Modification

Before:	(1	0	1	1	0	1	1	0)
After:	(0	1	1	0	0	1	1	0)

Before:	(1.38	-69.4	326.44	0.1)
After:	(1.38	-67.5	326.44	0.1)

- Causes movement in the search space (local or global)
- Restores lost information to the population

Crossover: Recombination



*

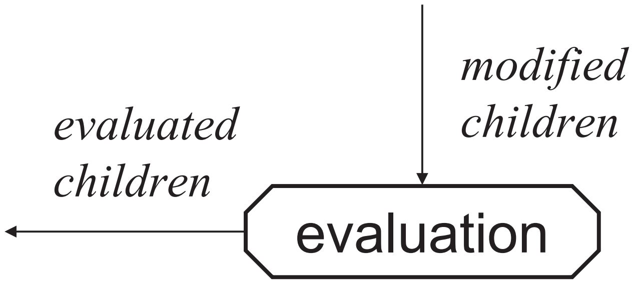
P1 (0 1 1 0 1 0 0 0) → (0 1 0 0 1 0 0 0) C1

P2 (1 1 0 1 1 0 1 0) (1 1 1 1 1 0 1 0) C2

Crossover is a critical feature of genetic algorithms:

- It greatly accelerates search early in evolution of a population
- It leads to effective combination of schemata (subsolutions on different chromosomes)

Evaluation



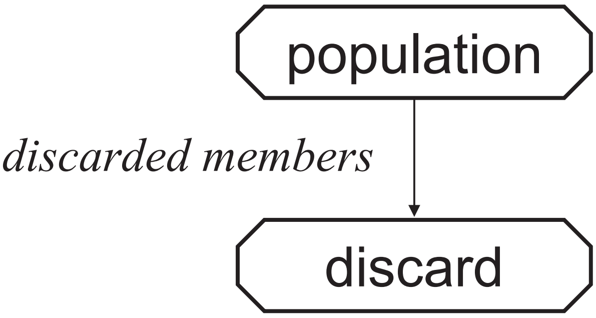
modified children

evaluated children

evaluation

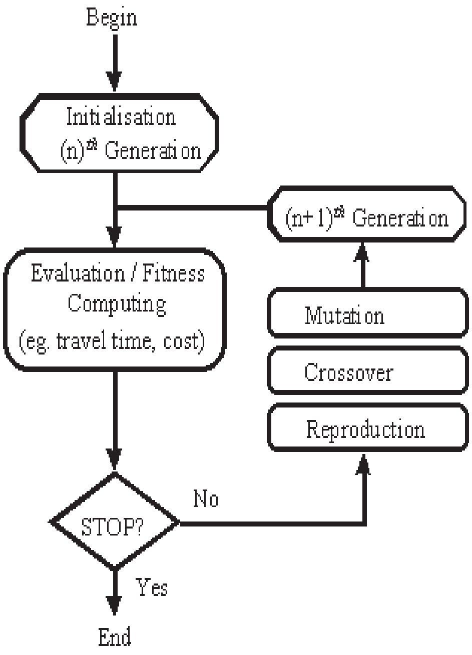
- The evaluator decodes a chromosome and assigns it a fitness measure
- The evaluator is the only link between a classical GA and the problem it is solving

Deletion

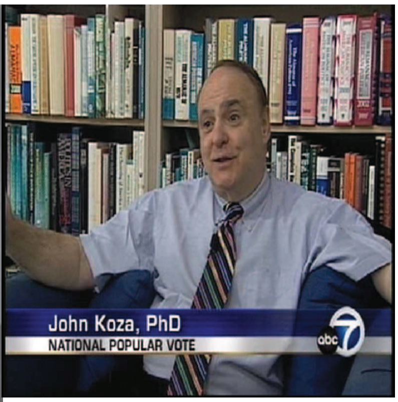


- *Generational* GA:
  entire populations replaced with each iteration
- *Steady-state* GA:
  a few members replaced each generation

Genetic Programming

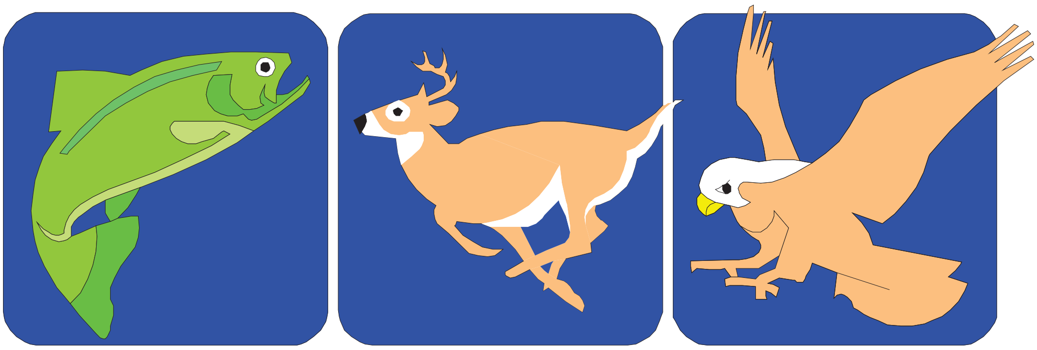


David E. Goldberg
Director, Illinois Genetic Algorithms Laboratory (IlliGAL)
University of Illinois at Urbana-Champaign



John R. Koza
Consulting Professor
Department of Electrical Engineering
School of Engineering, Stanford University

A Simple Example



"The Gene is by far the most sophisticated program around."

- Bill Gates, *Business Week*, June 27, 1994

A Simple Example

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

Representation

Representation is an ordered list of city numbers known as an *order-based* GA.

1) Chennai
2) Coimbatore
3) Vellore
4) Hyderabad
5) Mumbai
6) Pondicherry
7) Tiruchi
8) Vizag

CityList1	(3	5	7	2	1	6	4	8)
CityList2	(2	5	7	6	8	1	3	4)

Crossover

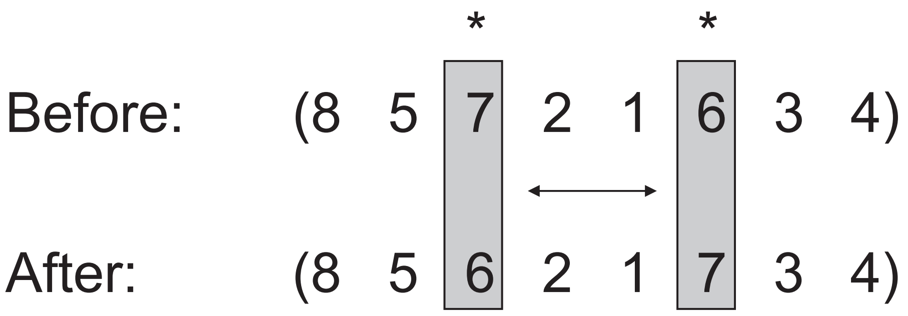
Crossover combines inversion and recombination:

			*			*		
Parent1	(3	5	7	2	1	6	4	8)
Parent2	(2	5	7	6	8	1	3	4)
Child	(8	5	7	2	1	6	3	4)

This operator is called the *Order1* crossover.

Mutation

Mutation involves reordering of the list:

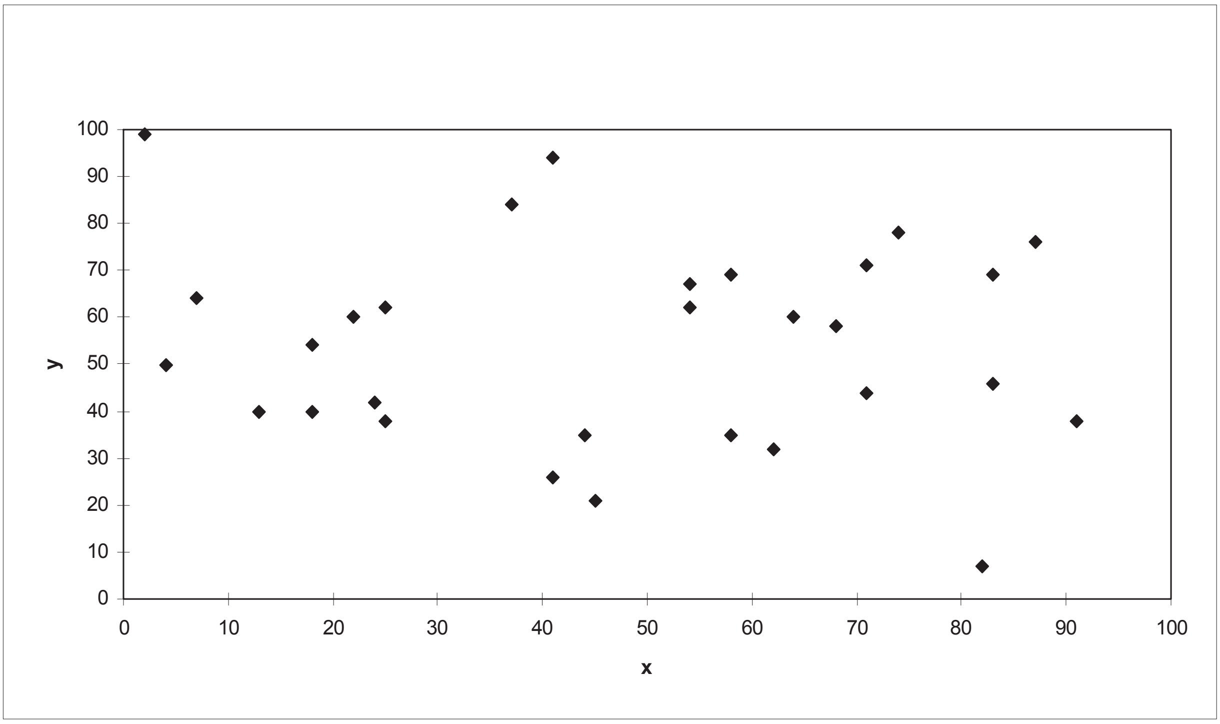


* *

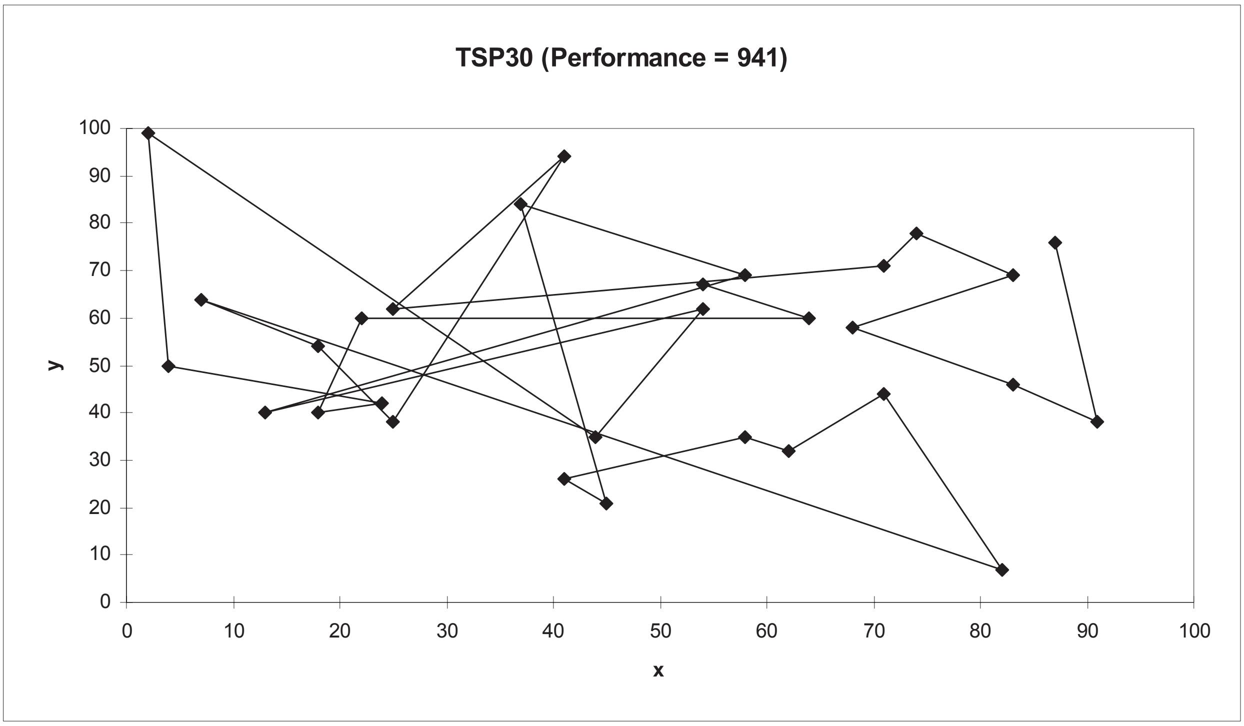
Before: (8 5 7 2 1 6 3 4)

After: (8 5 6 2 1 7 3 4)

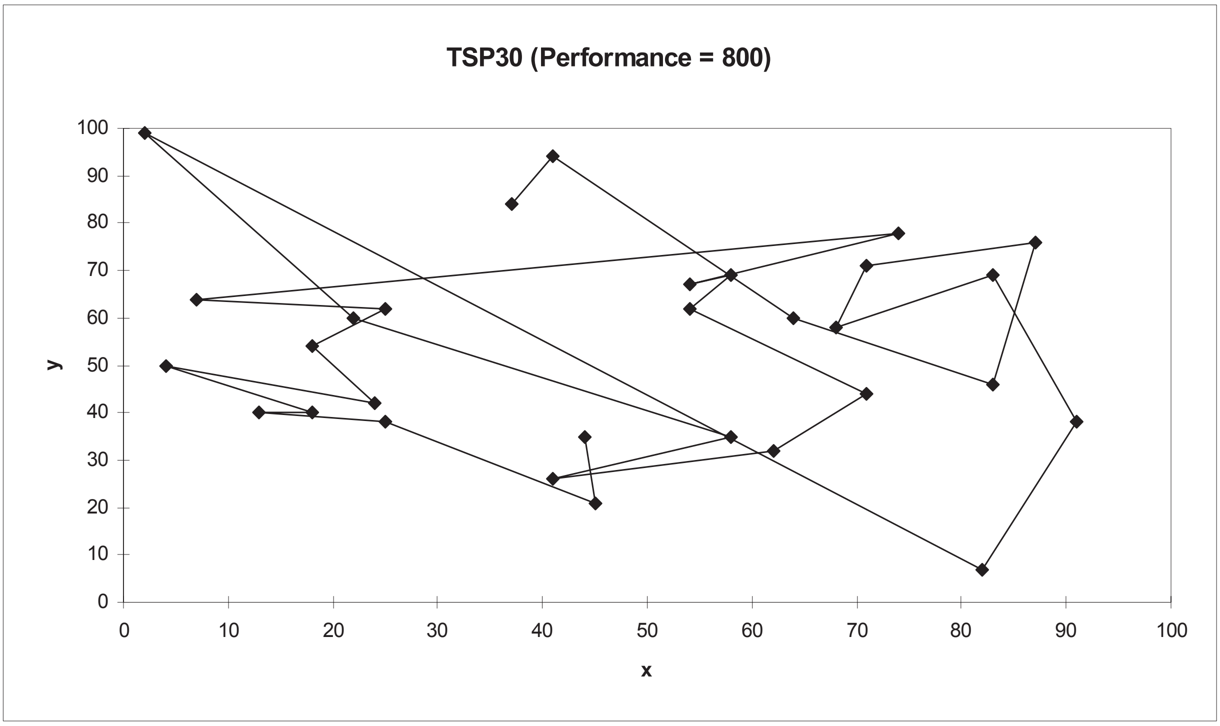
TSP Example: 30 Cities



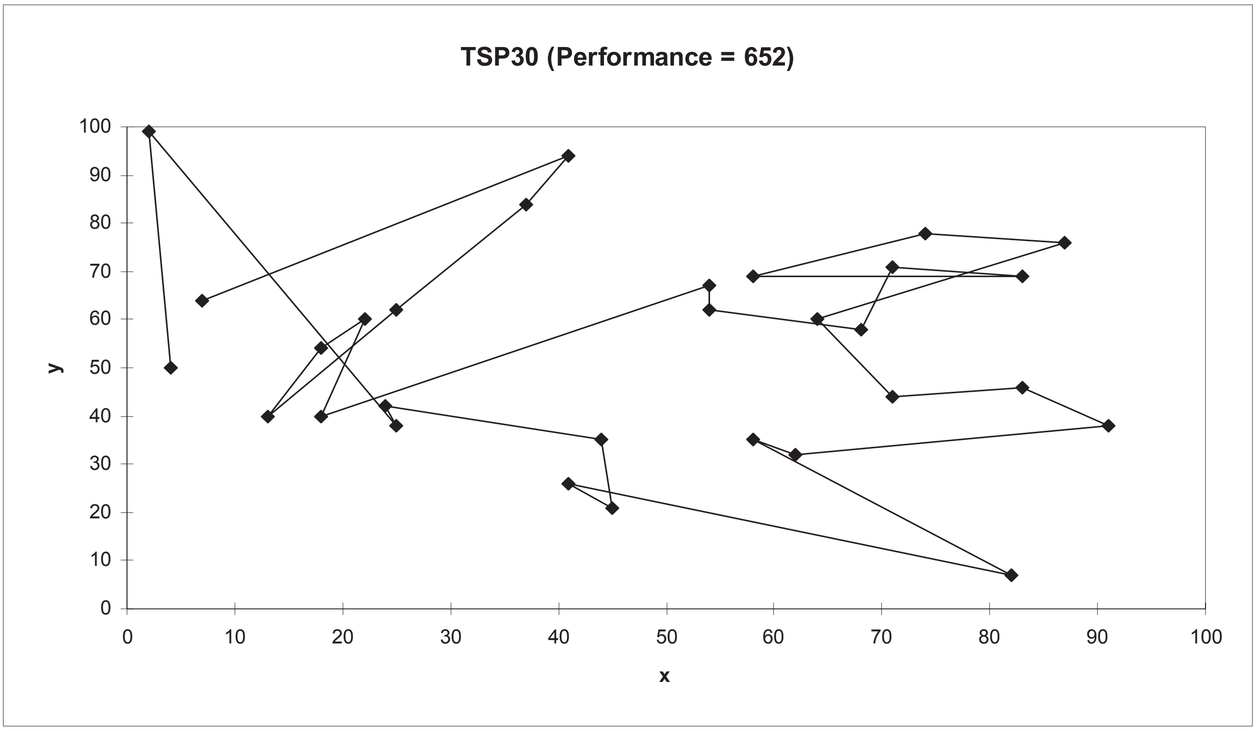
Solution $_i$ (Distance = 941)



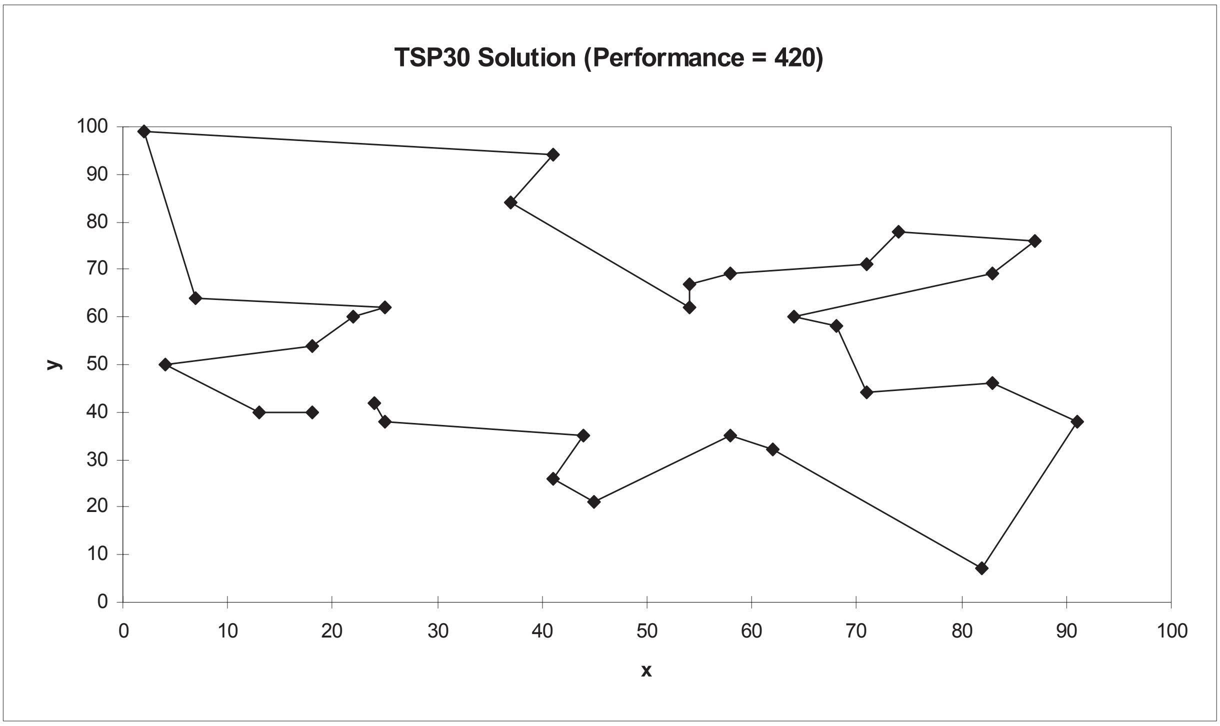
Solution $_j$(Distance = 800)



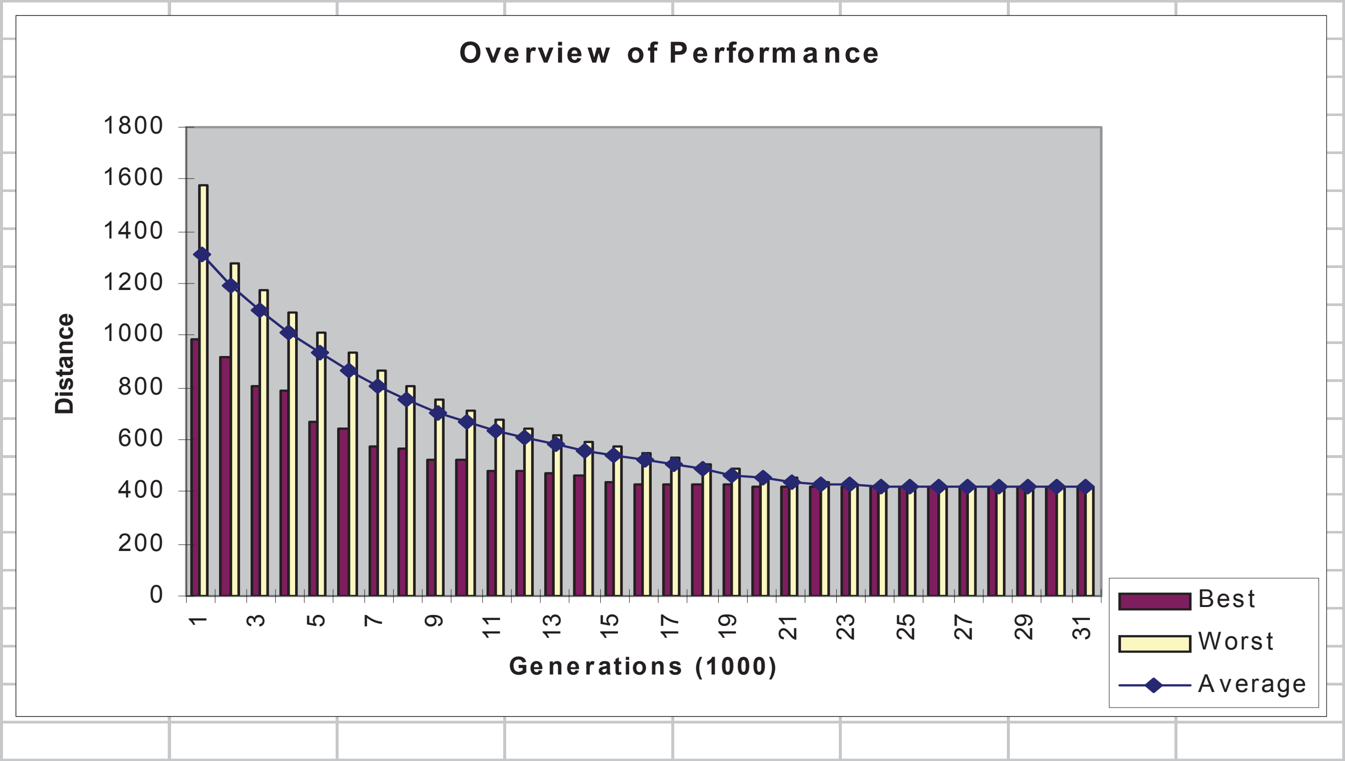
Solution $_k$(Distance = 652)



Best Solution (Distance = 420)



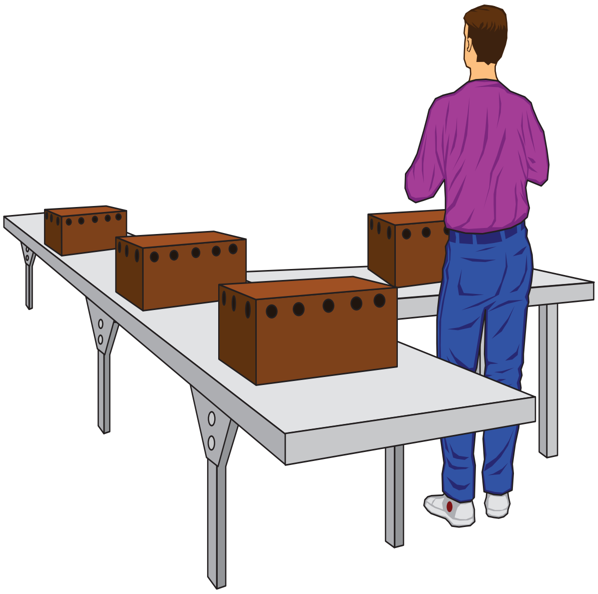
Overview of Performance



Considering the GA Technology

"Almost eight years ago ... people at Microsoft wrote a program which uses some genetic function for finding short code sequences. Windows 3.0 and XP, NT, and almost all Microsoft applications products have shipped with pieces of code created by that system."

- Nathan Myhrvold, Microsoft Advanced Technology Group, *Wired*, September 1995



Issues for GA Practitioners

- Choosing basic implementation issues:
  - representation
  - population size, mutation rate, ...
  - selection, deletion policies
  - crossover, mutation operators
- Termination Criteria
- Performance, scalability
- Solution is only as good as the evaluation function (often hardest part)

Benefits of Genetic Algorithms

- Concept is easy to understand
- Modular, separate from application
- Supports multi-objective optimization
- Good for "noisy" environments
- Always gives answer; answer gets better with time
- Inherently parallel; easily distributed

Benefits of Genetic Algorithms (cont.)

- Multiple ways to speed up and improve a GA-based application as knowledge about problem domain is gained
- Easy to exploit previous or alternate solutions
- Flexible building blocks for hybrid applications
- Substantial history and range of use

When to Use a GA

- Alternate solutions are too slow or much complicated
- Need an exploratory tool to examine new approaches
- Problem is similar to one that has already been successfully solved.
- Want to hybridize with an existing solution
- Benefits of the GA technology meet key problem requirements

Some GA Application Types

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

Some Applications of Genetic Algorithms

- Optimization and design
  - numerical optimization, circuit design, airplane design, factory scheduling, drug design, network optimization
- Automatic programming
  - evolving computer programs (e.g., for image processing), evolving cellular automata
- Machine learning and adaptive control
  - robot navigation, evolution of rules for solving "expert" problems, evolution of neural networks, adaptive computer security, adaptive user interfaces

Some Applications of Genetic Algorithms

- Complex data analysis and time-series prediction
  - prediction of chaotic systems, financial-market prediction, protein-structure prediction

- Scientific models of complex systems
  - economics, immunology, ecology, population genetics, evolution, cancer

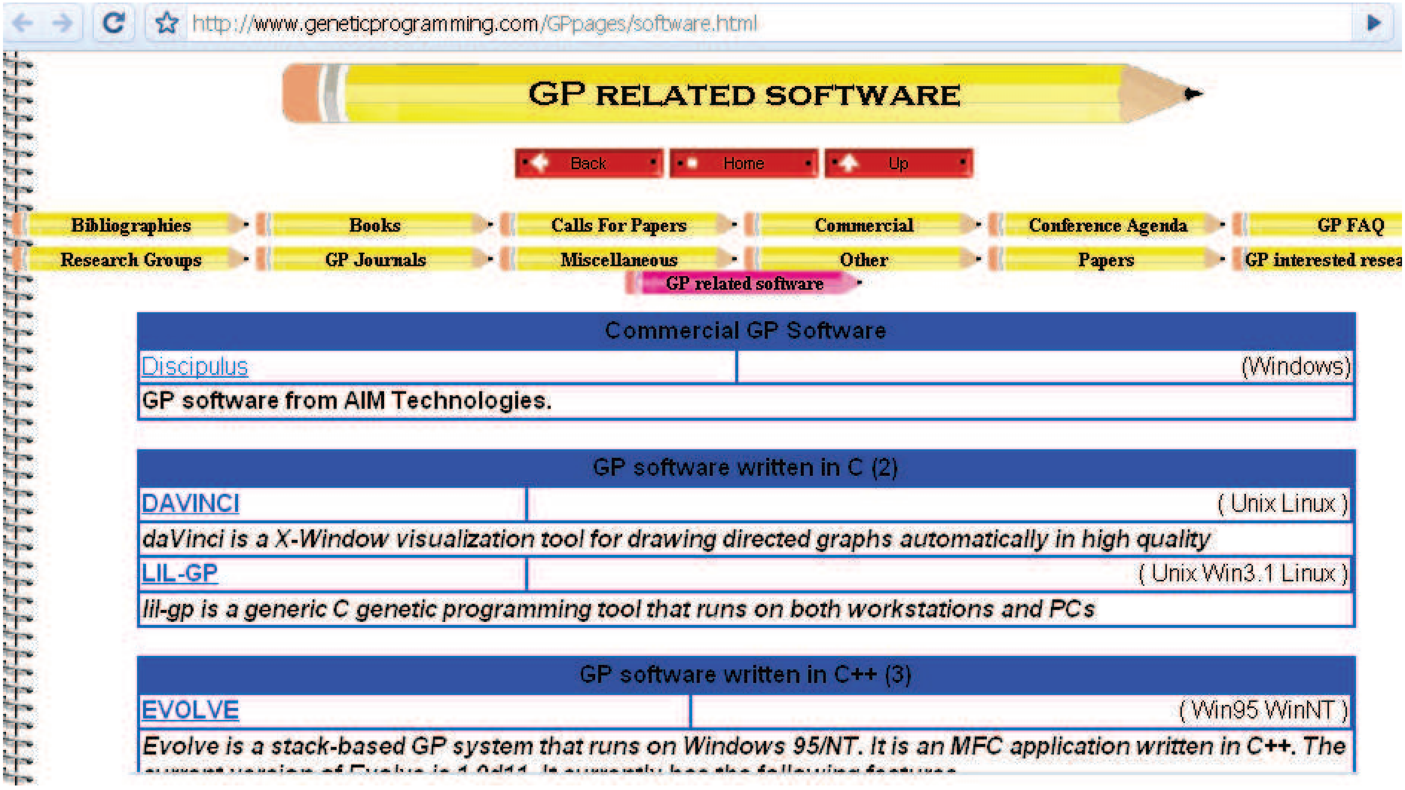
Evolutionary process

Essentials of Darwinian evolution:

- Organisms reproduce in proportion to their *fitness* in the environment
- Offspring inherit all traits from parents
- Traits are inherited with some variation, via mutation and sexual recombination

Essentials of evolutionary algorithms:

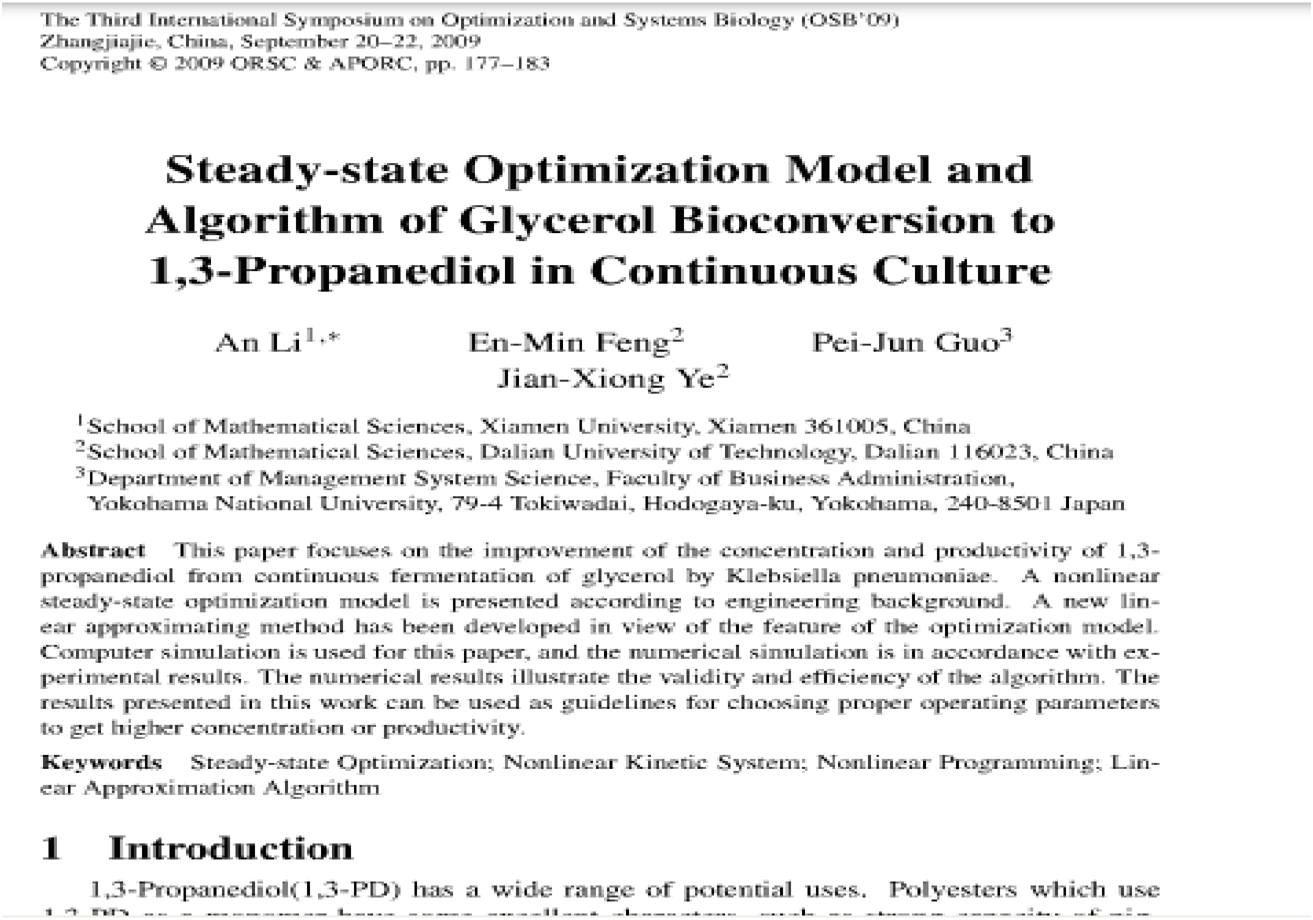
- Computer "organisms" (e.g., programs) reproduce in proportion to their *fitness* of problem environment (e.g., how well they perform a desired task)
- Offspring (outcome) inherit only strong traits (fitness)from their parent.
- Traits are inherited, with some variation, via mutation and cross-over methods



Commercial GP Software	
Discipulus	(Windows)
GP software from AIM Technologies.	

GP software written in C (2)	
DAVINCI	(Unix Linux)
daVinci is a X-Window visualization tool for drawing directed graphs automatically in high quality	
LIL-GP	(Unix Win3.1 Linux)
lil-gp is a generic C genetic programming tool that runs on both workstations and PCs	

GP software written in C++ (3)	
EVOLVE	(Win95 WinNT)
Evolve is a stack-based GP system that runs on Windows 95/NT. It is an MFC application written in C++. The current version of Evolve is 1.0d11. It currently has the following features	



The Third International Symposium on Optimization and Systems Biology (OSB'09)
Zhangjiajie, China, September 20–22, 2009
Copyright © 2009 ORSC & APORC, pp. 177–183

Steady-state Optimization Model and Algorithm of Glycerol Bioconversion to 1,3-Propanediol in Continuous Culture

An Li[1,*] En-Min Feng[2] Pei-Jun Guo[3]
Jian-Xiong Ye[2]

[1]School of Mathematical Sciences, Xiamen University, Xiamen 361005, China
[2]School of Mathematical Sciences, Dalian University of Technology, Dalian 116023, China
[3]Department of Management System Science, Faculty of Business Administration,
Yokohama National University, 79-4 Tokiwadai, Hodogaya-ku, Yokohama, 240-8501 Japan

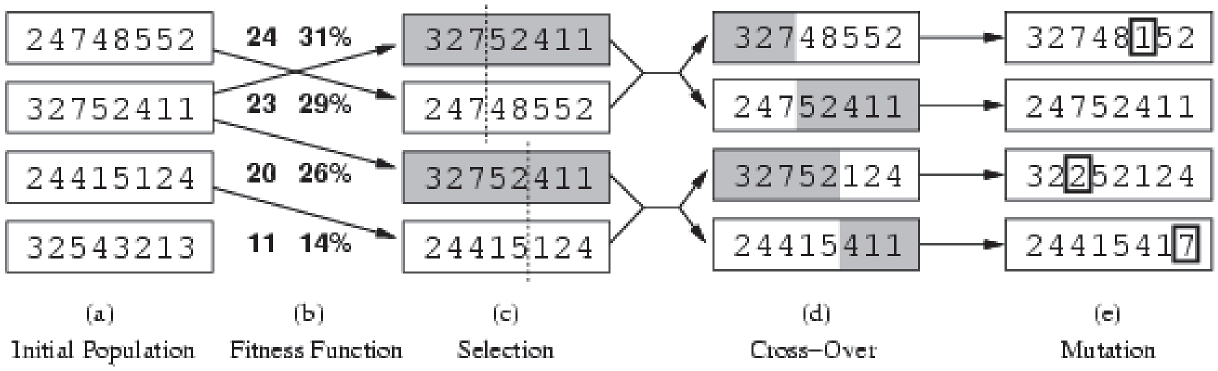
Abstract This paper focuses on the improvement of the concentration and productivity of 1,3-propanediol from continuous fermentation of glycerol by Klebsiella pneumoniae. A nonlinear steady-state optimization model is presented according to engineering background. A new linear approximating method has been developed in view of the feature of the optimization model. Computer simulation is used for this paper, and the numerical simulation is in accordance with experimental results. The numerical results illustrate the validity and efficiency of the algorithm. The results presented in this work can be used as guidelines for choosing proper operating parameters to get higher concentration or productivity.

Keywords Steady-state Optimization; Nonlinear Kinetic System; Nonlinear Programming; Linear Approximation Algorithm

1 Introduction

1,3-Propanediol(1,3-PD) has a wide range of potential uses. Polyesters which use 1,3-PD as a monomer have some excellent characters, such as strong capacity of pig-

Genetic algorithms



- Fitness function: number of non-attacking pairs of queens (min = 0, max = 8 × 7/2 = 28)   24/(24+23+20+11) = 31%   23/(24+23+20+11) = 29% etc

DEFINITION OF THE GENETIC ALGORITHM (GA)

The *genetic algorithm* is a probabilistic search algorithm that

(a) iteratively transforms a set (called a *population*) of mathematical objects (typically fixed-length binary character strings),

(b) each with an associated fitness value, into a new population of offspring objects using the Darwinian principle of natural selection,

(c) using operations that are patterned after naturally occurring genetic operations, such as crossover and mutation.

BENTLEY

BENTLEY GEOSPATIAL | Haestad Methods Solutions

Genetic Algorithm Optimization for Accurate Hydraulic and Water Quality Analysis of Water Systems

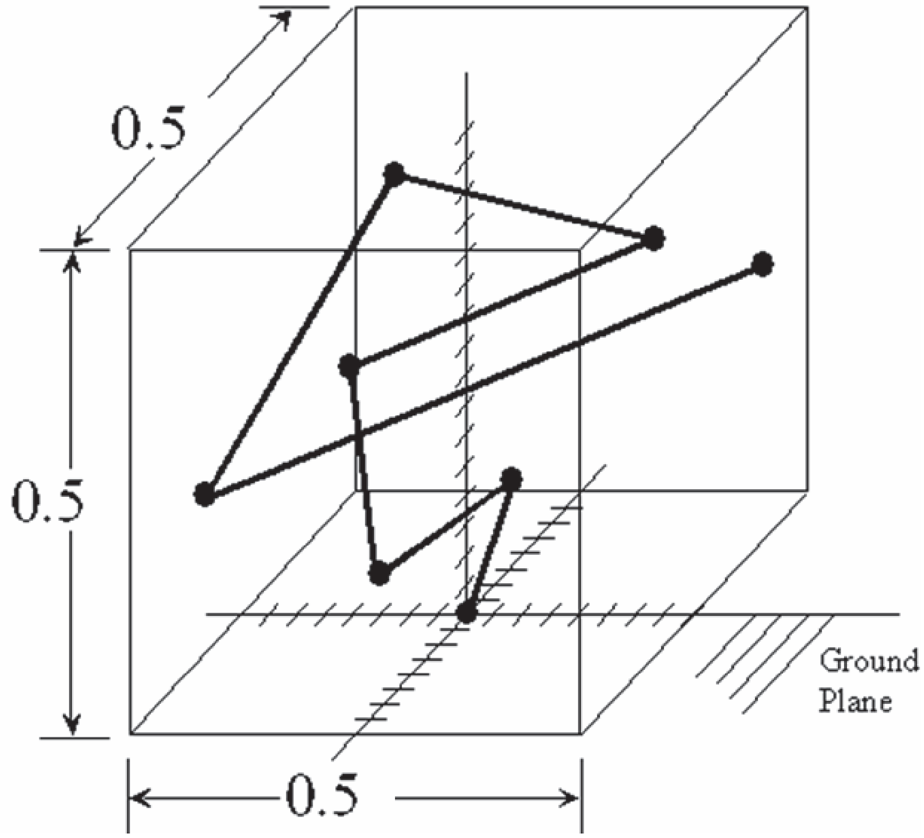
Z. Y. Wu, T. Walski, R. Mankowski, G. Herrin and R. Gurrieri
Bentley Systems, Incorporated, USA

E. F. Arniella, E. Gianellaand, Envirosoft Eng. & Sci., Inc., USA

C. Clark, City of Sidney, Ohio, USA

P. Sage, United Utilities PLC, UK

ANTENNA DESIGN



ANTENNA DESIGN

- The problem (Altshuler and Linden 1998) is to determine the *x-y-z* coordinates of the 3-dimensional position of the ends (*X1, Y1, Z1, X2, Y2, Z2,… , X7, Y7, Z7*) of 7 straight wires so that the resulting 7-wire antenna satisfies certain performance requirements
- The first wire starts at feed point (0, 0, 0) in the middle of the ground plane
- The antenna must fit inside the 0.5 cube

ANTENNA GENOME

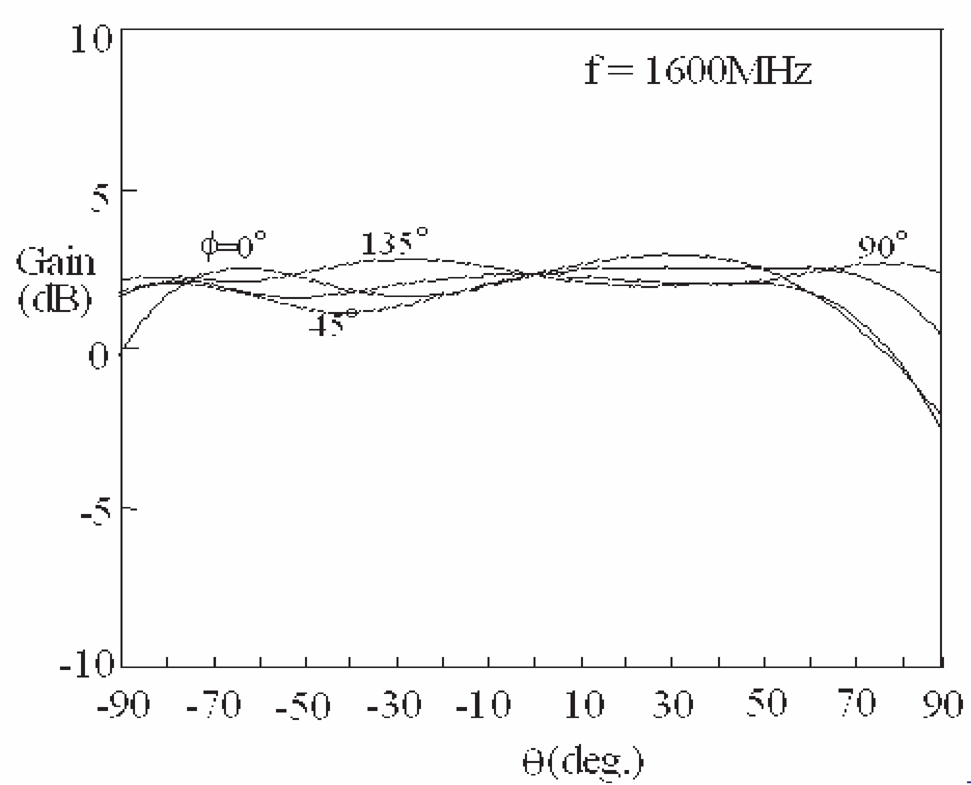
X_1	Y_1	Z_1	X_2	Y_2	Z_2	…
+0010	-1110	+0001	+0011	-1011	+0011	…

- 105-bit chromosome (genome)
- Each *x-y-z* coordinate is represented by 5 bits (4-bit granularity for data plus a sign bit)
- Total chromosome is 3 × 7 × 5 = 105 bits

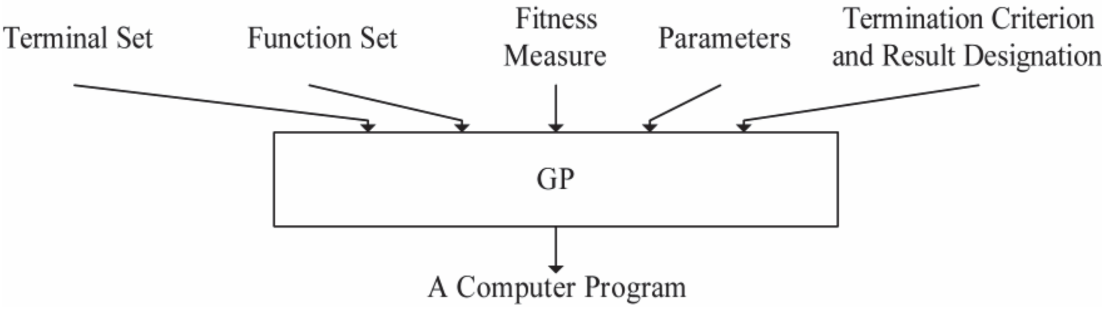
ANTENNA FITNESS

- Antenna is for ground-to-satellite communications for cars and handsets
- We desire near-uniform gain pattern 10° above the horizon
- Fitness is measured based on the “antenna's radiation pattern”. The radiation pattern is simulated by National Electro-magnetics Code (NEC)
- Fitness is “sum of the squares of the difference between the average gain and the antenna's gain”
- Radiation can be considered for angles Θ between -90° and +90° and all azimuth angles Φ from 0° to 180°
- The smaller the value of fitness, the better

GRAPH OF ANTENNA FITNESS



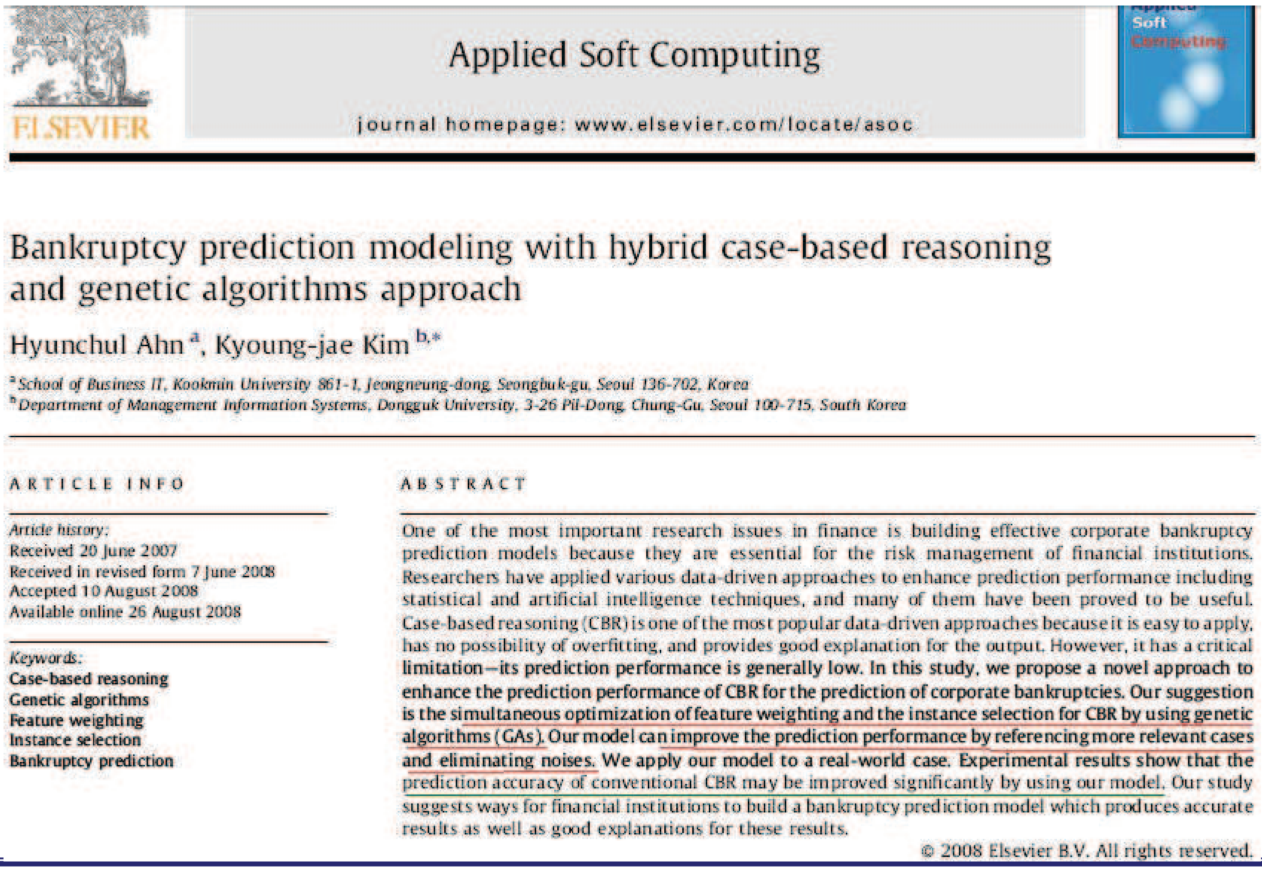
FIVE MAJOR PREPARATORY STEPS FOR GP



- Determining the set of terminal inputs
- Determining the set of functions
- Determining the fitness measure
- Determining the parameters for the run
- Determining the method for designating a result and the criterion for terminating a run

PREPARATORY STEPS

	Objective:	Find a computer program with one input (independent variable `X`) whose output equals the given data
1	Terminal set:	`T = {X, Random-Constants}`
2	Function set:	`F = {+, -, *, %}`
3	Fitness:	The sum of the absolute value of the differences between the candidate program's output and the given data (computed over numerous values of the independent variable *x* from –1.0 to +1.0)
4	Parameters:	Population size *M* = 4
5	Termination:	An individual emerges whose sum of absolute errors is less than 0.1



Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Bankruptcy prediction modeling with hybrid case-based reasoning and genetic algorithms approach

Hyunchul Ahn[a], Kyoung-jae Kim[b,*]

[a] School of Business IT, Kookmin University 861-1, Jeongneung-dong, Seongbuk-gu, Seoul 136-702, Korea
[b] Department of Management Information Systems, Dongguk University, 3-26 Pil-Dong, Chung-Gu, Seoul 100-715, South Korea

ARTICLE INFO

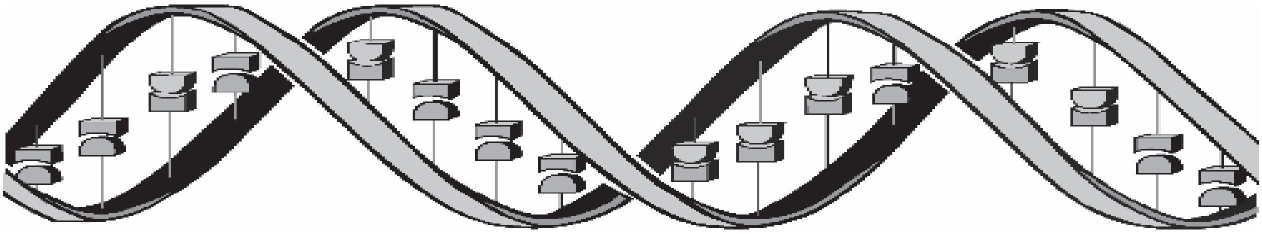
Article history:
Received 20 June 2007
Received in revised form 7 June 2008
Accepted 10 August 2008
Available online 26 August 2008

Keywords:
Case-based reasoning
Genetic algorithms
Feature weighting
Instance selection
Bankruptcy prediction

ABSTRACT

One of the most important research issues in finance is building effective corporate bankruptcy prediction models because they are essential for the risk management of financial institutions. Researchers have applied various data-driven approaches to enhance prediction performance including statistical and artificial intelligence techniques, and many of them have been proved to be useful. Case-based reasoning (CBR) is one of the most popular data-driven approaches because it is easy to apply, has no possibility of overfitting, and provides good explanation for the output. However, it has a critical limitation—its prediction performance is generally low. In this study, we propose a novel approach to enhance the prediction performance of CBR for the prediction of corporate bankruptcies. Our suggestion is the simultaneous optimization of feature weighting and the instance selection for CBR by using genetic algorithms (GAs). Our model can improve the prediction performance by referencing more relevant cases and eliminating noises. We apply our model to a real-world case. Experimental results show that the prediction accuracy of conventional CBR may be improved significantly by using our model. Our study suggests ways for financial institutions to build a bankruptcy prediction model which produces accurate results as well as good explanations for these results.

Conclusions



Question: *'Why GAs are so smart, rich?'*

Answer: *'Genetic algorithms **are** rich - rich in application across a large and growing number of disciplines.'*

- David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*