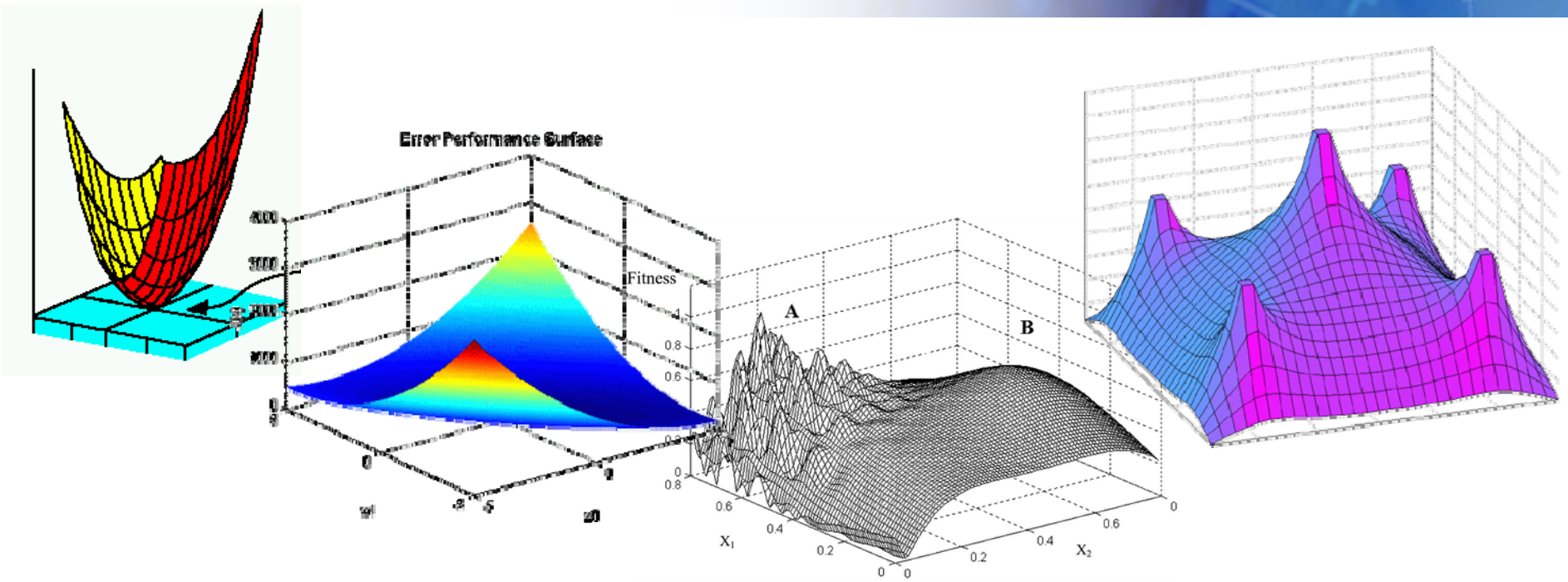
Engineering Optimization and Nature

Ender Ayanoglu
Center for Pervasive Communications and Computing
The Henry Samueli School of Engineering
University of California, Irvine

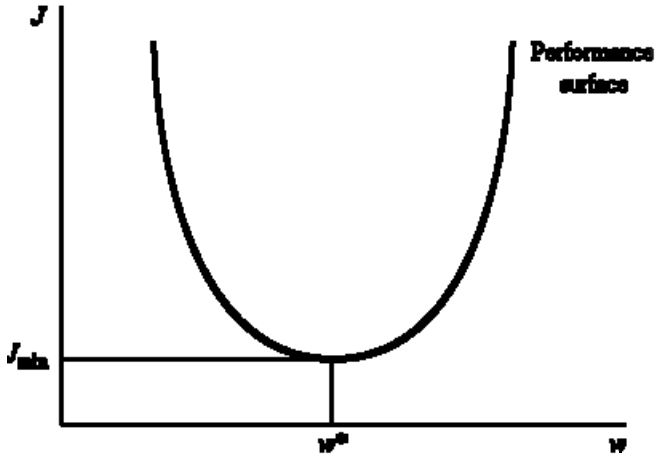
Calit2 UC Irvine Division
Workshop on Biological and Computing/Communication Systems
July 6, 2009

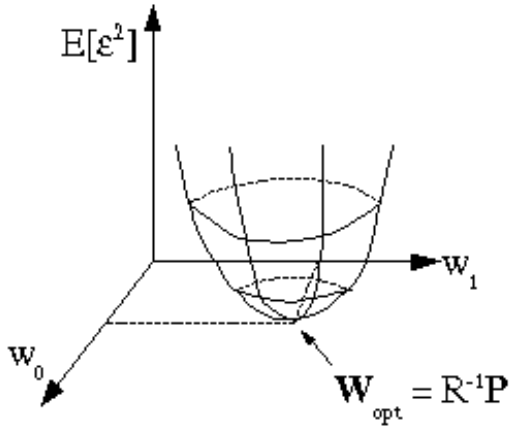
Optimization: Performance Surface



- Problem: Minimize or maximize a performance criterion in terms of the variables
- Can be formulated as a mathematical expression as a surface against the space of the variables

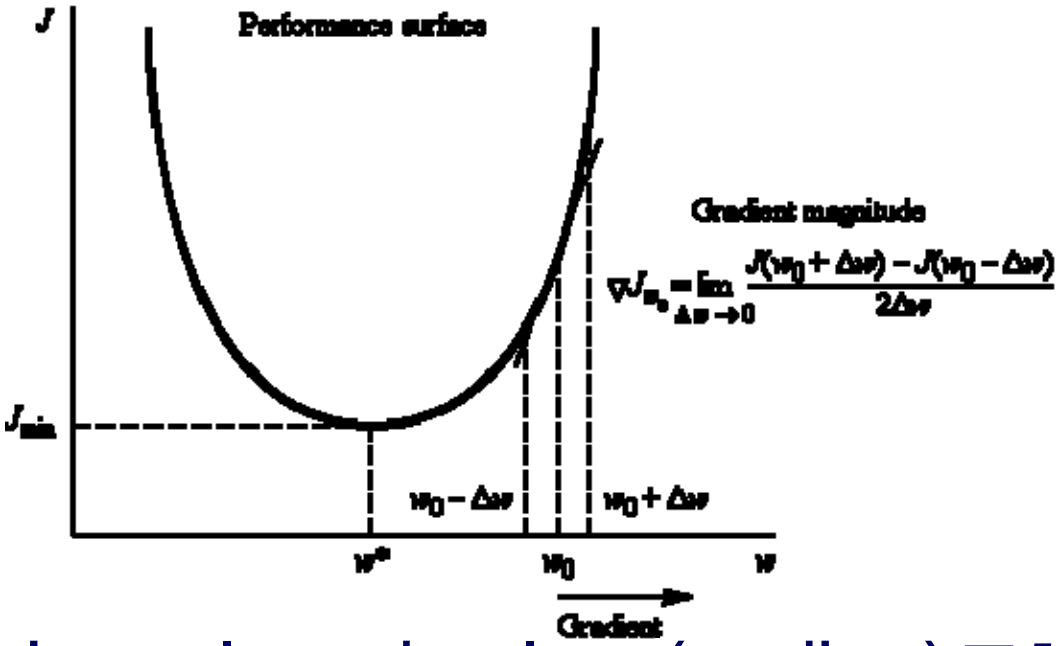
Special Case: Unimodal Performance Surface





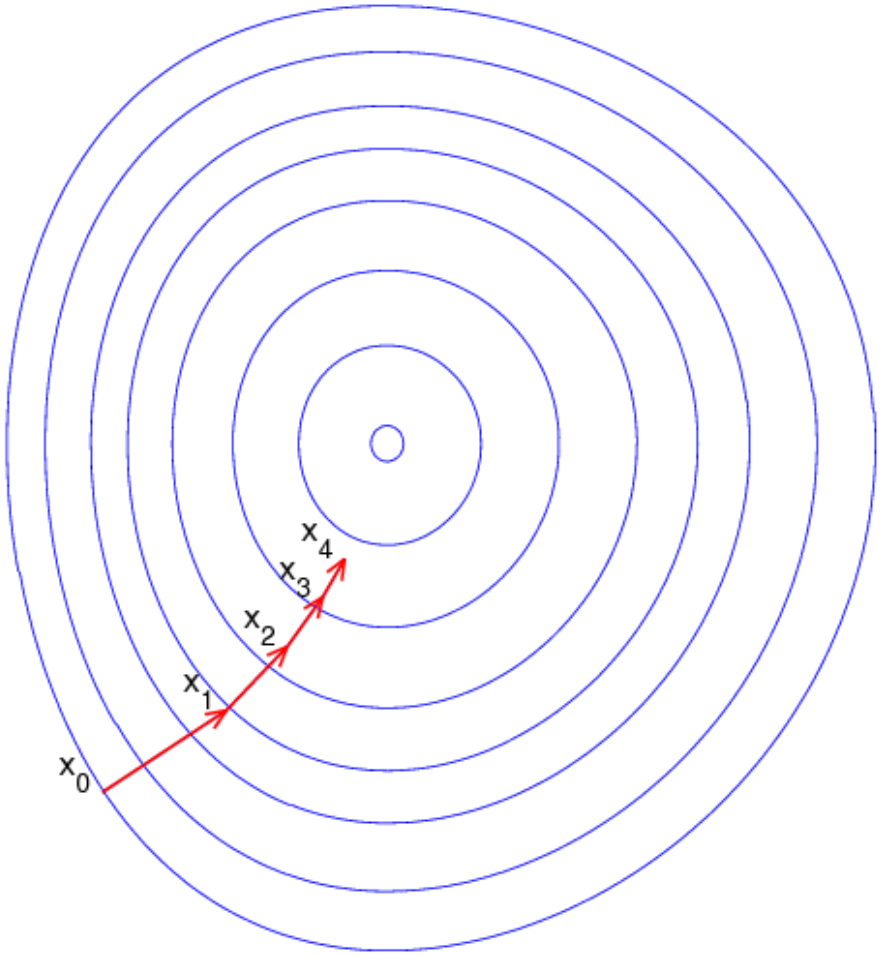
- There is a single (global) minimum or maximum, for example, a quadratic surface
- The goal is to find the minimum w^* that realizes minimum performance value J_{min}
- One may be able to evaluate $J(w)$, $\nabla J(w)$

Steepest Descent Optimization



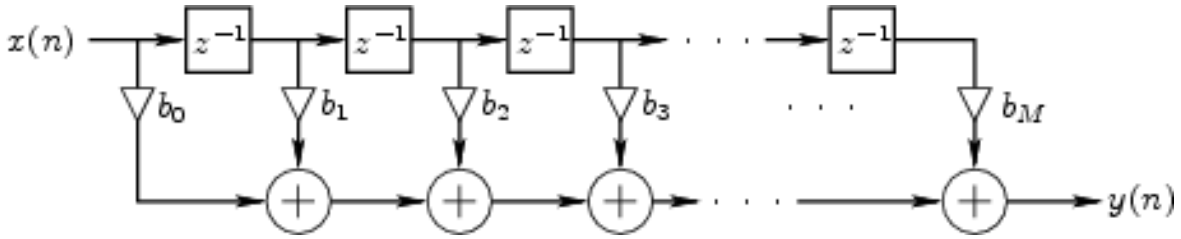
- At each point evaluate the slope (gradient) $\nabla J(w)$ of the performance surface $J(w)$ with respect to parameters w
- Move in the (negative) direction of the gradient towards the bottom of the performance curve
- $w_{k+1} = w_k - \mu \nabla J(w_k)$

Unimodal Performance Surface
Steepest Descent Trajectory



- At each step the descent is in the maximum possible rate
- Depending on the value μ, one can approach the minimum fast
- Depending on the value μ, there may be oscillations around J_{min}
- An estimate for $\nabla J(w_k)$ can be substituted
- Well-known example: LMS algorithm

LMS Algorithm

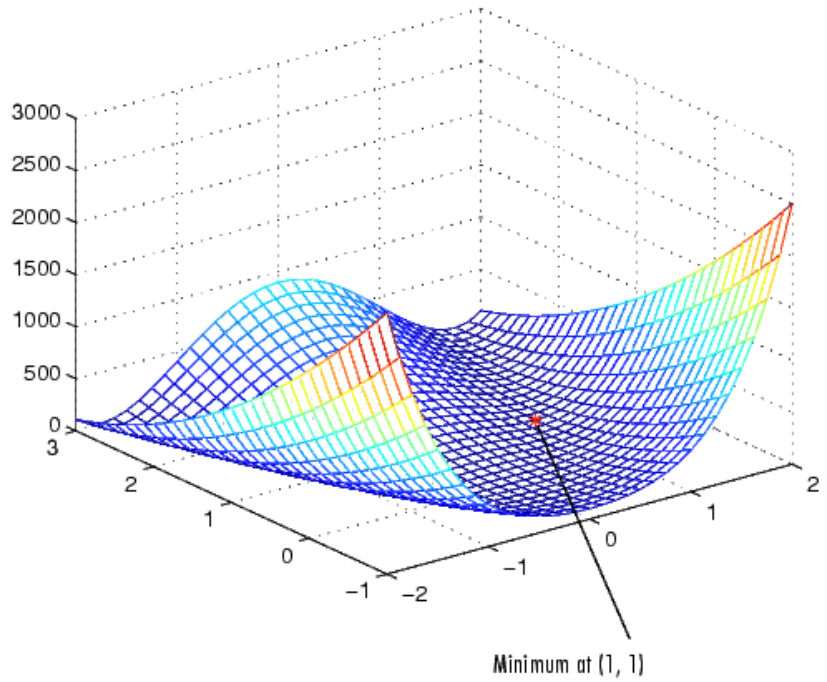


- Employs a digital filter with a series of delay elements (z^{-1}) and variable weights b_i, $i = 0, 1, \ldots, M$
- There is a desired response $d(n) = d_k$
- The error is $e(n) = d(n) - y(n) = e_k$
- The goal is to minimize mean squared error $E[e^2(n)]$
- Approximating the true gradient with its stochastic value

$$w_{k+1} = w_k + \mu e_k x_k$$

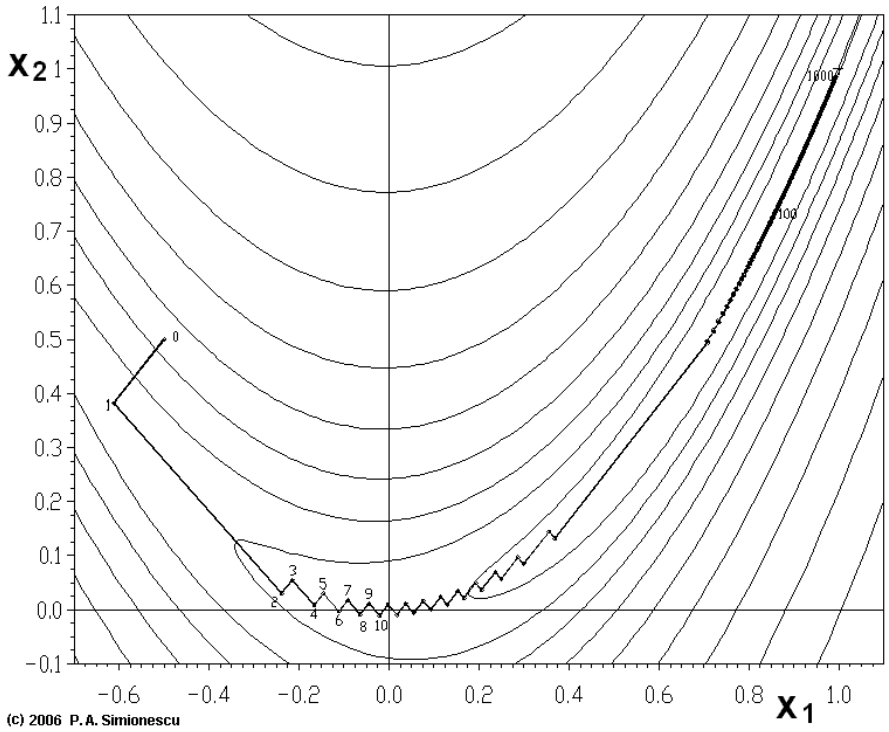
- Works well. Used in voiceband modems, neural networks.

Rosenbrock Function



- Nonconvex function used as a test problem for optimization algorithms
- $f(x, y) = (1 - x)^2 + 100\,(y - x^2)^2$
- Global minimum at $(x, y) = (1, 1)$ where $f(1, 1) = 0$

Steepest Descent with Rosenbrock Function



- The Rosenbrock function has a narrow curved valley which contains the minimum
- The bottom of the valley is very flat
- Because of the curved flat valley the optimization is zig-zagging slowly with small step sizes towards the minimum

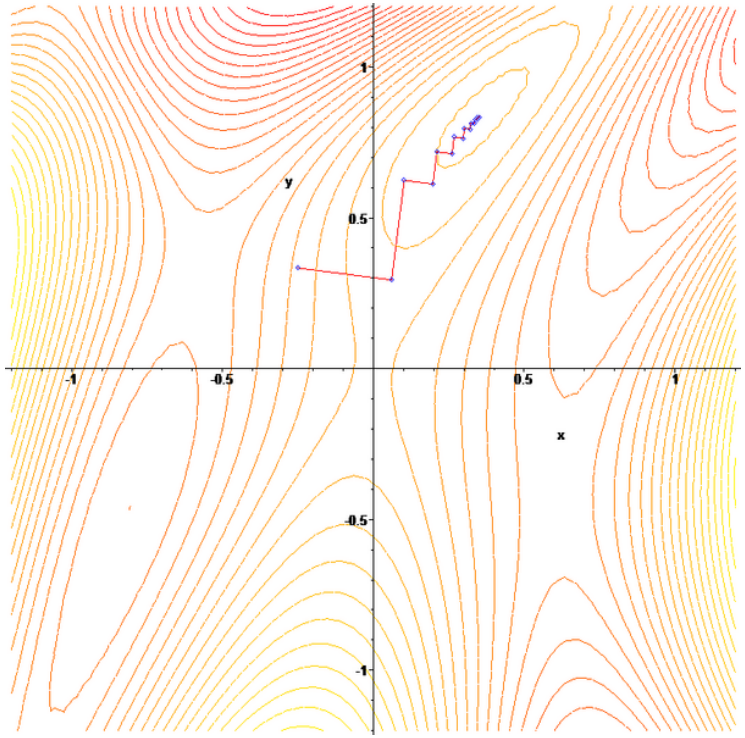
Making Steepest Descent Converge Faster

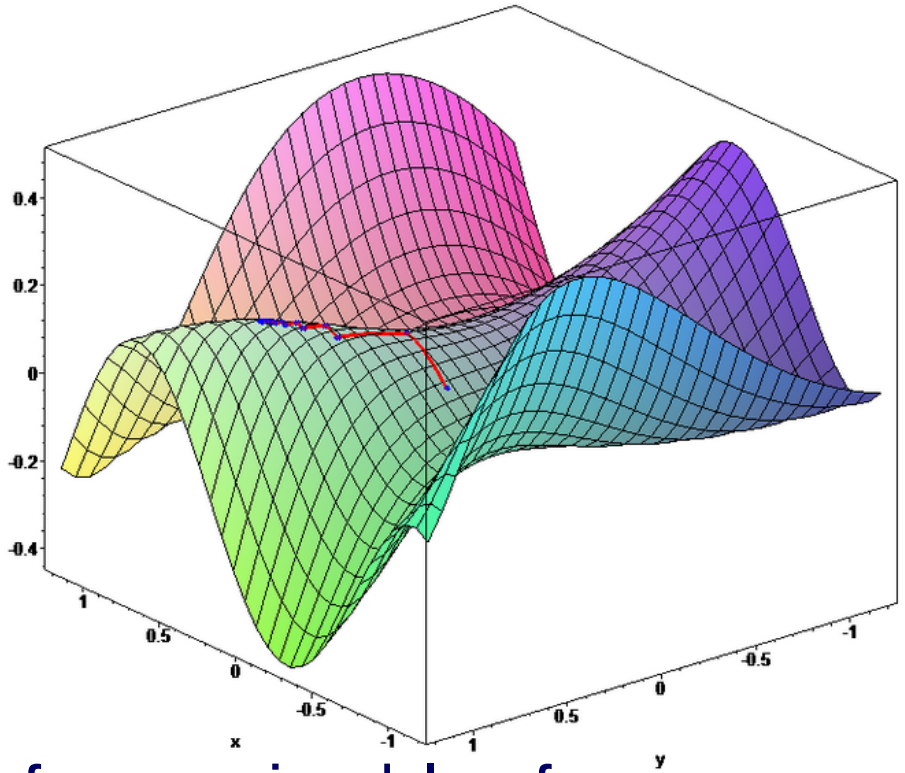
Steepest descent type algorithms that take the second derivative into account

- Newton-Raphson
- Polack-Ribiere
- Davidson-Fletcher-Powel
- Broydon-Fletcher-Goldfarb-Shanno
- Others

Nonunimodal Performance Surface Gradient Ascent

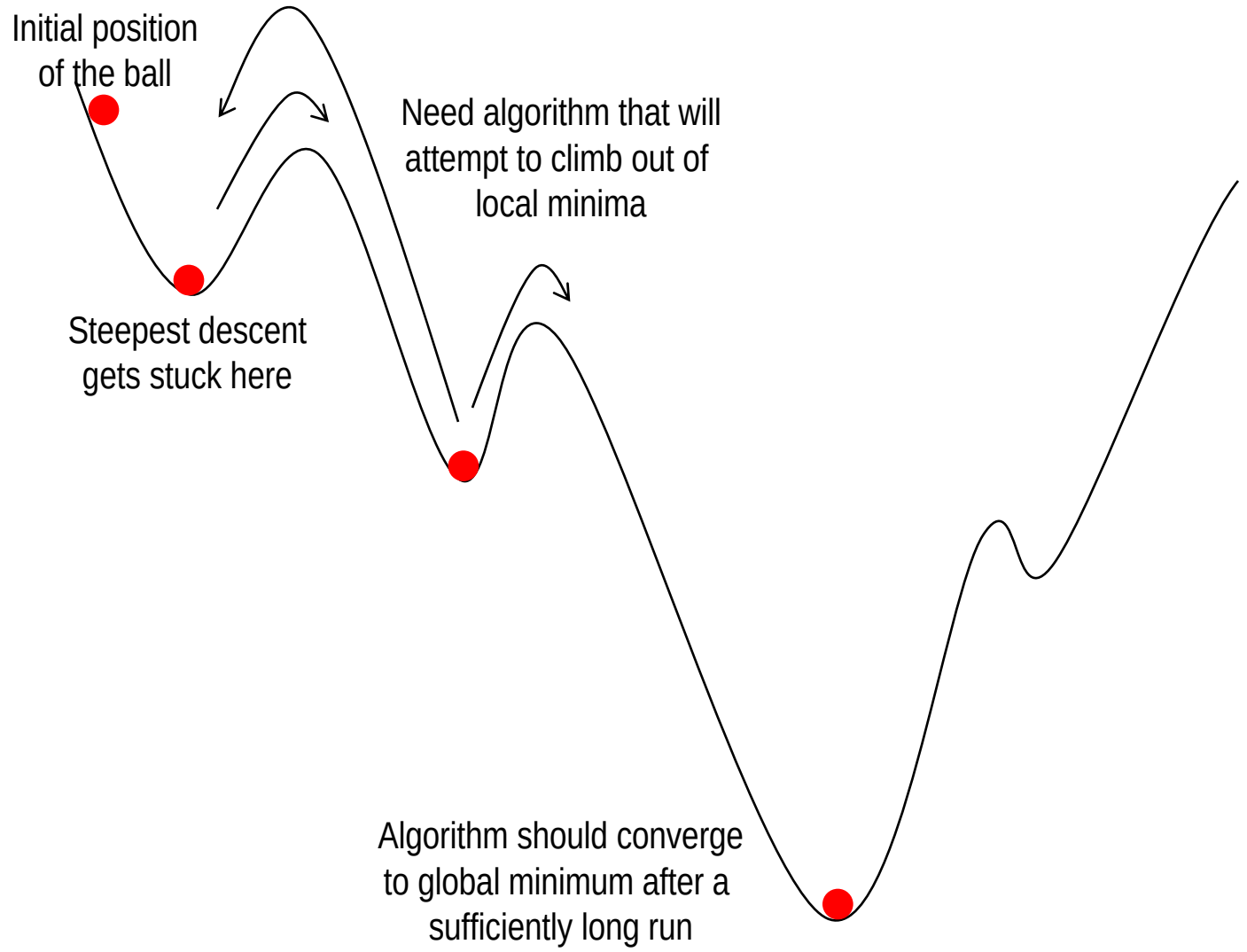
$f(x, y) = (x^2/2 - y^2/4 + 3) \cos(2x + 1 - e^y)$





- Steepest descent is not suitable for nonunimodal performance surfaces
- It gets stuck at a local minimum/maximum

How to Escape Local Minima?



Simulated Annealing: Motivation

- Annealing in metals
- Heat the solid state metal to a high temperature
- Cool it down very slowly according to a specific schedule.
- *If the heating temperature is sufficiently high to ensure random state and the cooling process is slow enough to ensure thermal equilibrium, then the atoms will place themselves in a pattern that corresponds to the global energy minimum of a perfect crystal (Metropolis et al., 1953)*

Simulated Annealing

Step 1: Initialize – Start with a random initial placement. Initialize a very high "temperature."

Step 2: Move – Perturb the placement through a defined move.

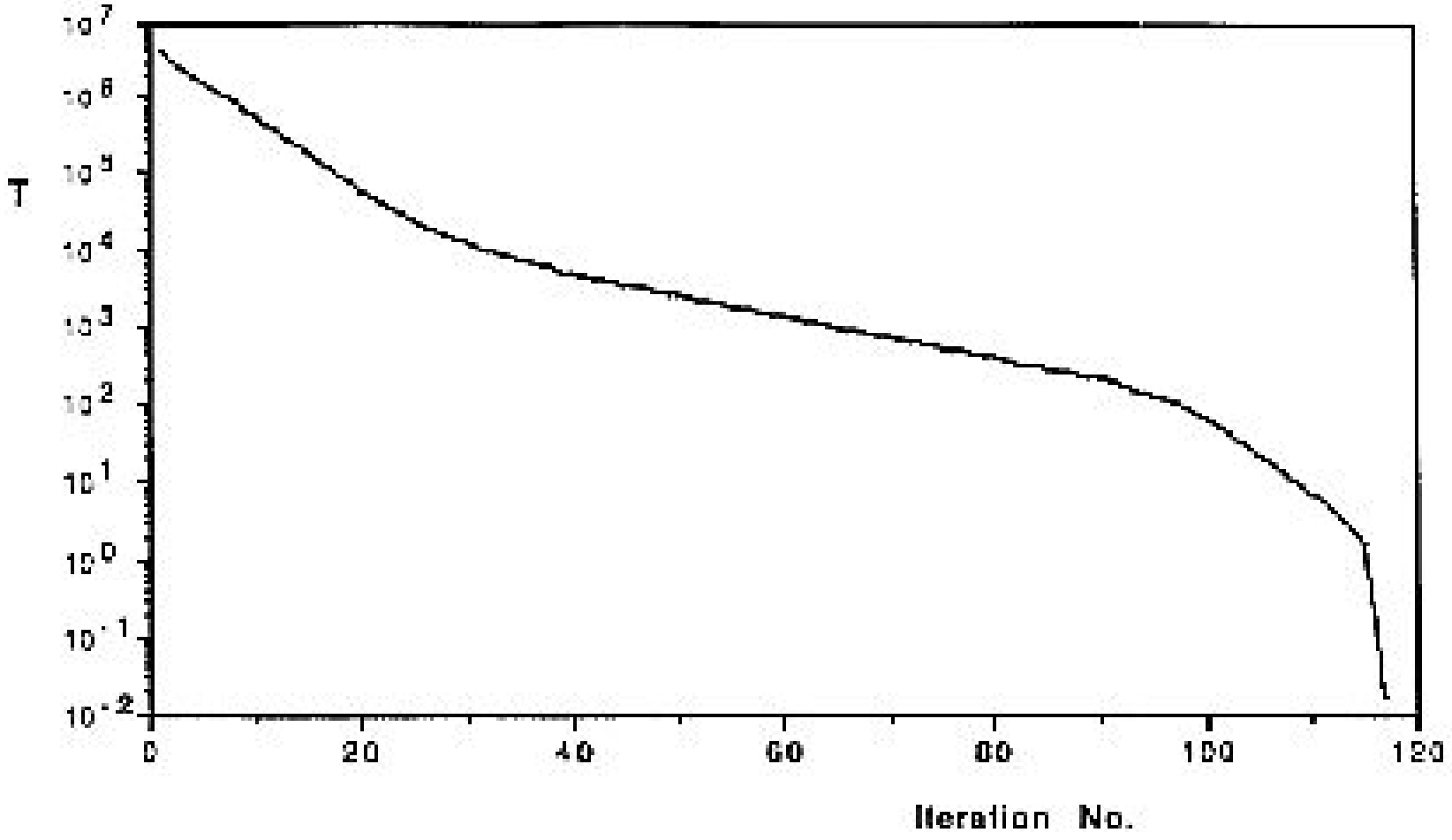
Step 3: Calculate score – calculate the change in the score due to the move made.

Step 4: Choose – Depending on the change in score, accept or reject the move. The probability of acceptance depends on the current "temperature."

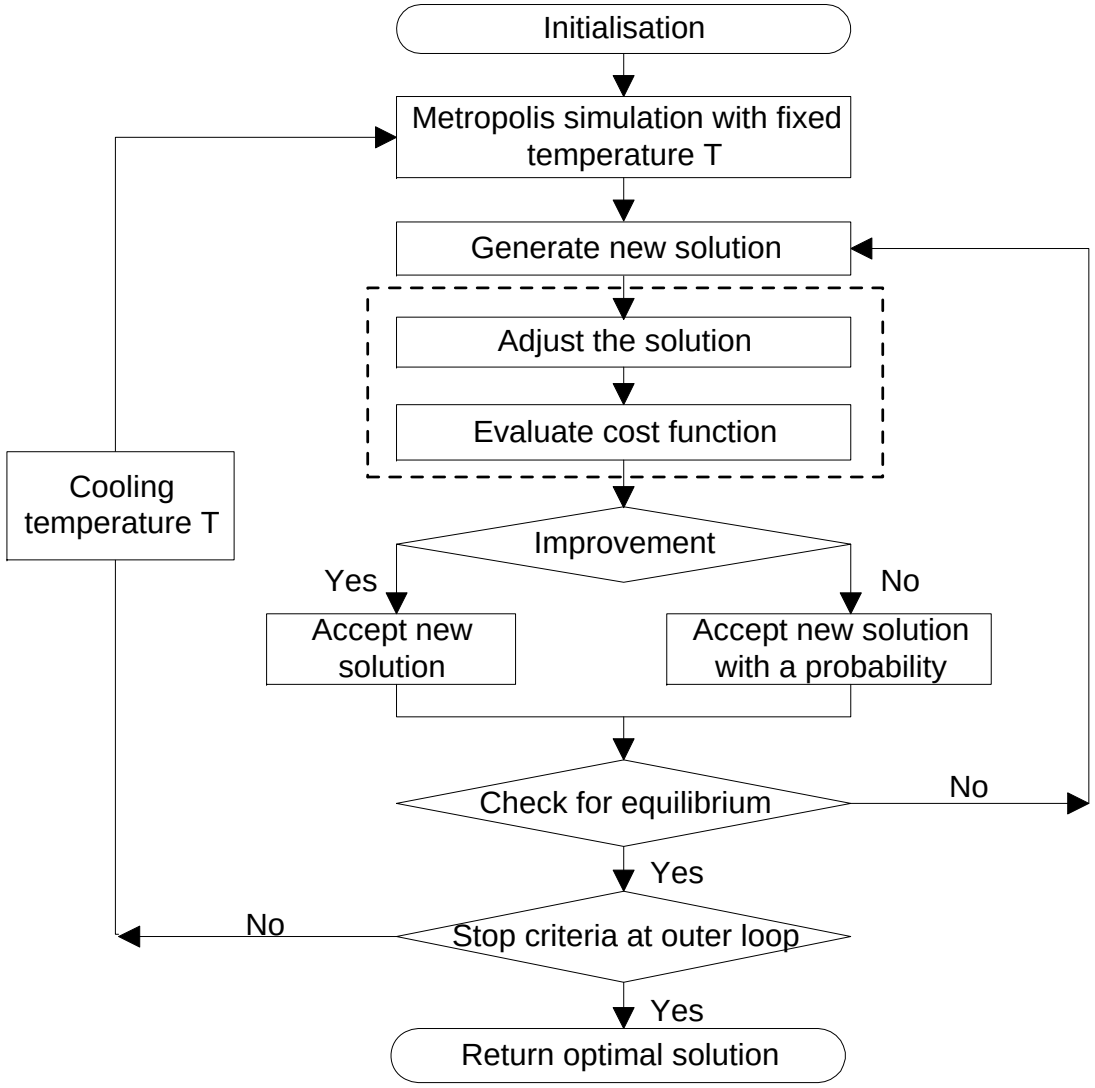
Step 5: Update and repeat– Update the temperature value by lowering the temperature. Go back to Step 2.

The process is carried out until "Freezing Point" is reached.

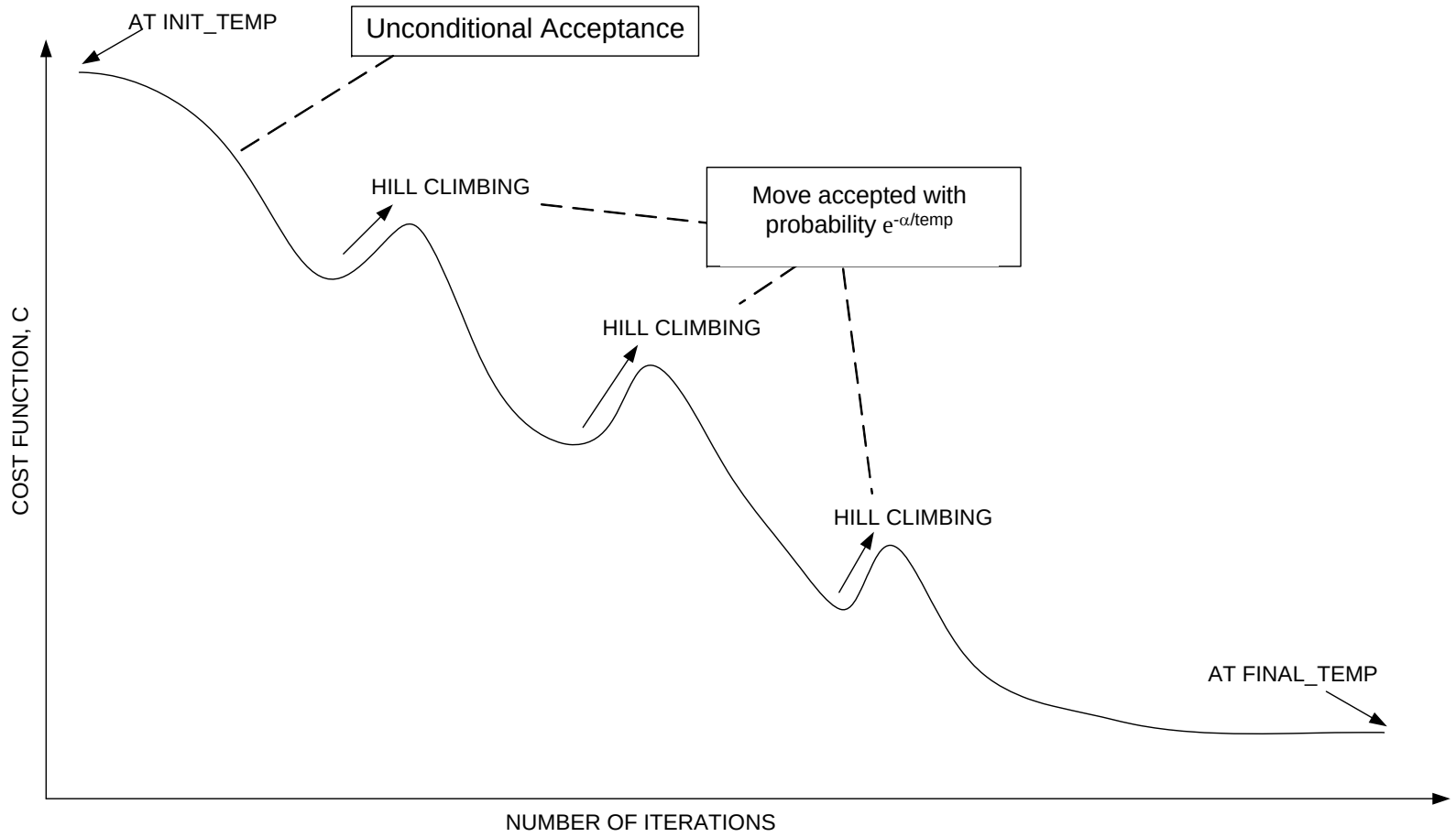
Cooling Schedule



SA Flowchart



Convergence of SA



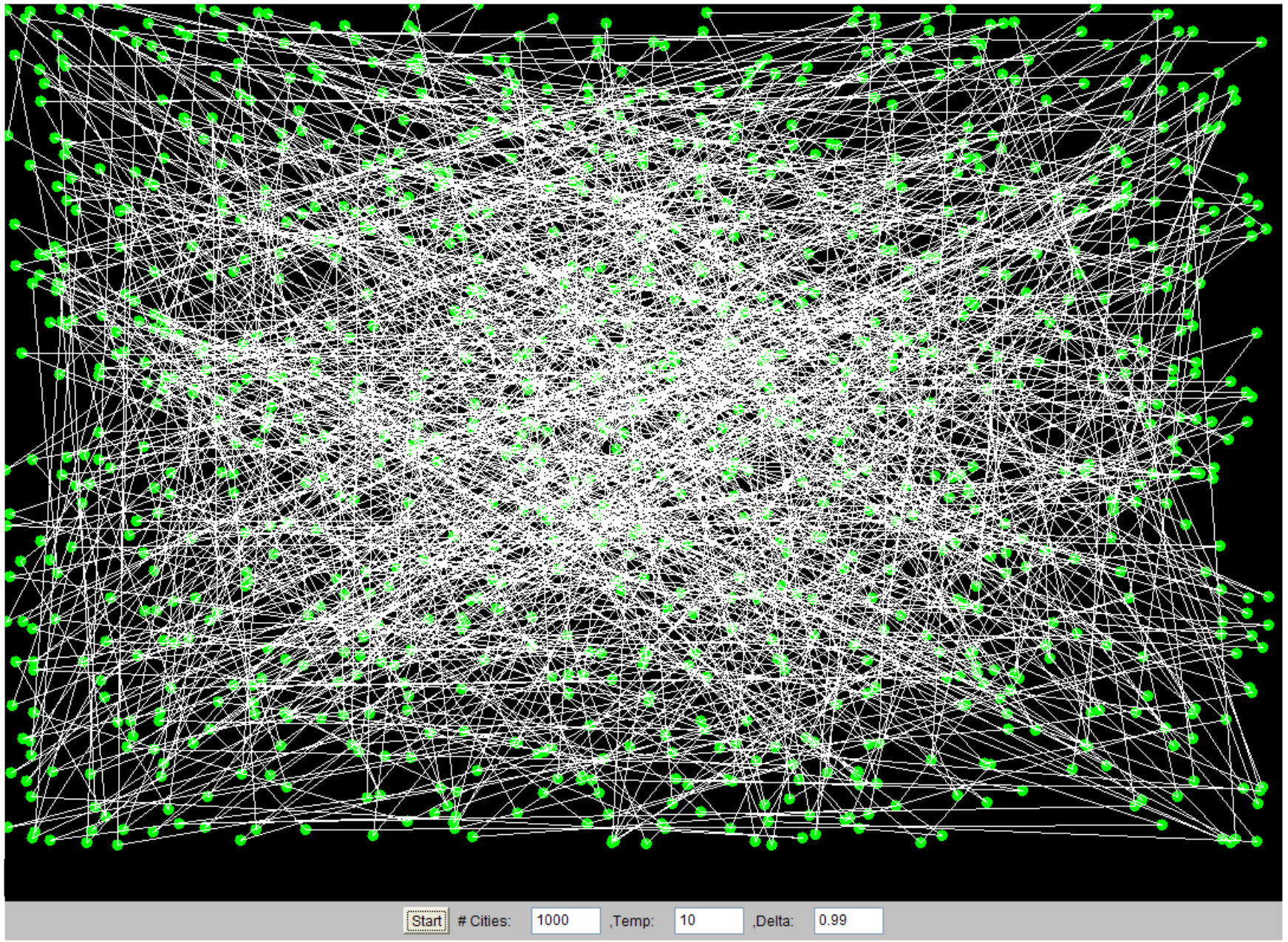
Simple Example

The Traveling Salesman Problem

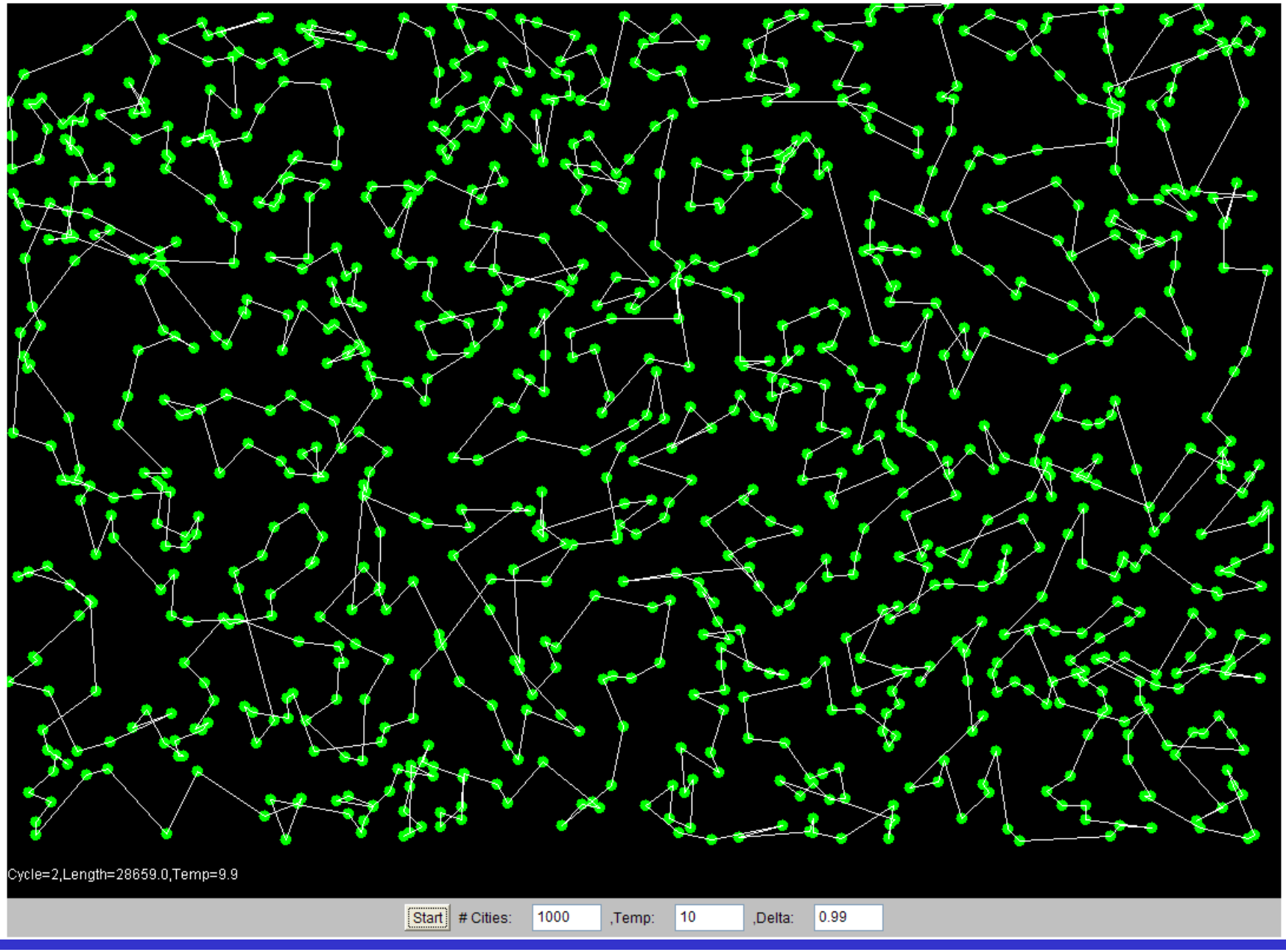
Find a tour of a given set of cities so that

- Each city is visited only once
- The total distance traveled is minimized

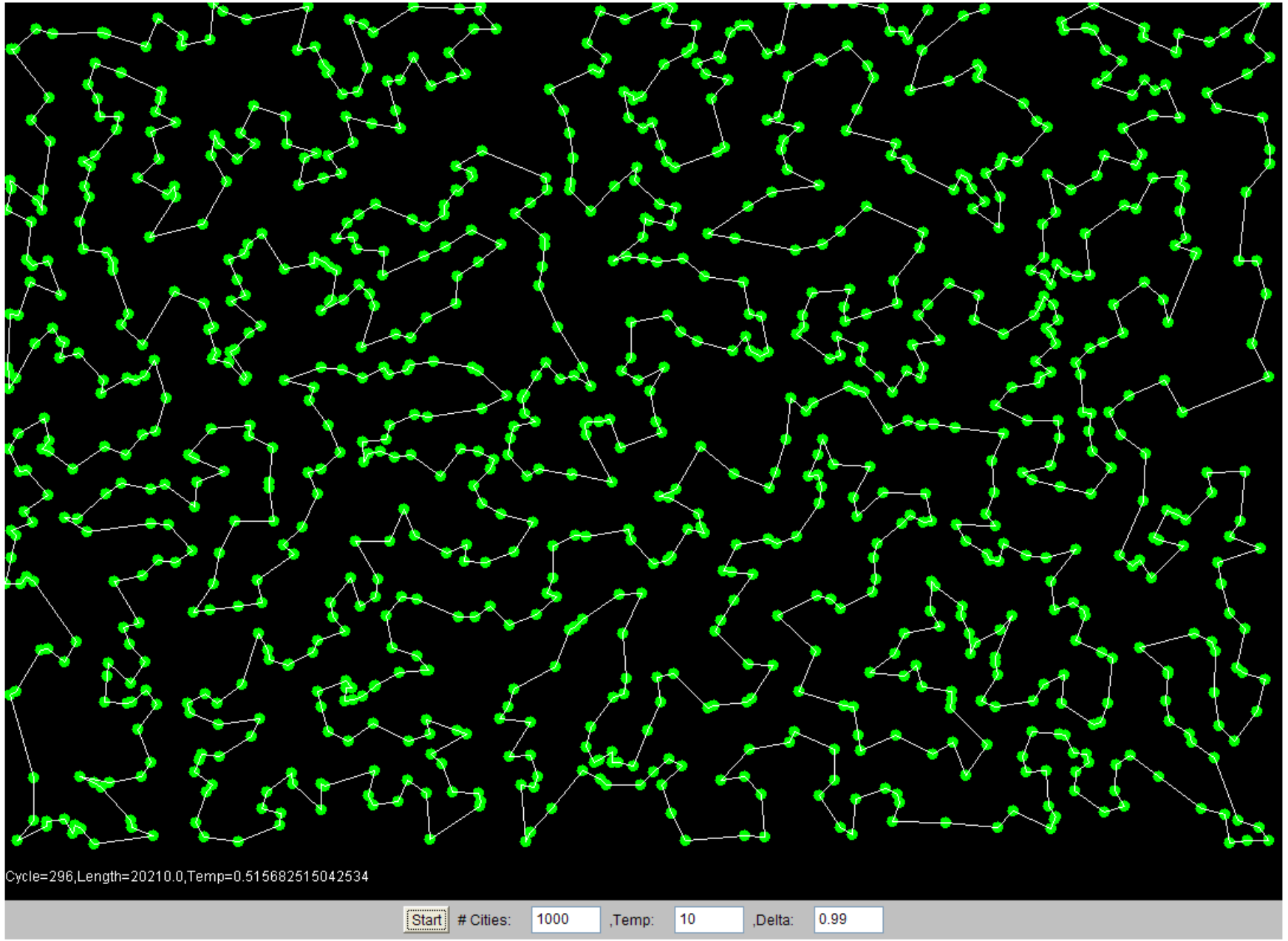
Simulated Annealing Solution of the Traveling Salesman Problem



Simulated Annealing Solution of the Traveling Salesman Problem



Simulated Annealing Solution of the Traveling Salesman Problem



SA Properties

- SA is a general solution method that is easily applicable to a large number of problems
- "Tuning" of the parameters is relatively easy
- Generally the quality of the results of SA is good, although it can take a lot of time
- Results are generally not reproducible: another run can give a different result
- SA can leave an optimal solution and not find it again (so remember the best solution found so far)
- Proven to find the optimum under certain conditions; one of these conditions is that you must run forever

SA Applications

- Computer-aided circuit design (placement, routing, etc) [IBM, Berkeley]
- Signal processing
- Boltzmann machines (AI)
- Operations research
- Econometrics
- Biology: Protein structure calculations, Gene clustering, etc.

Genetic Algorithms

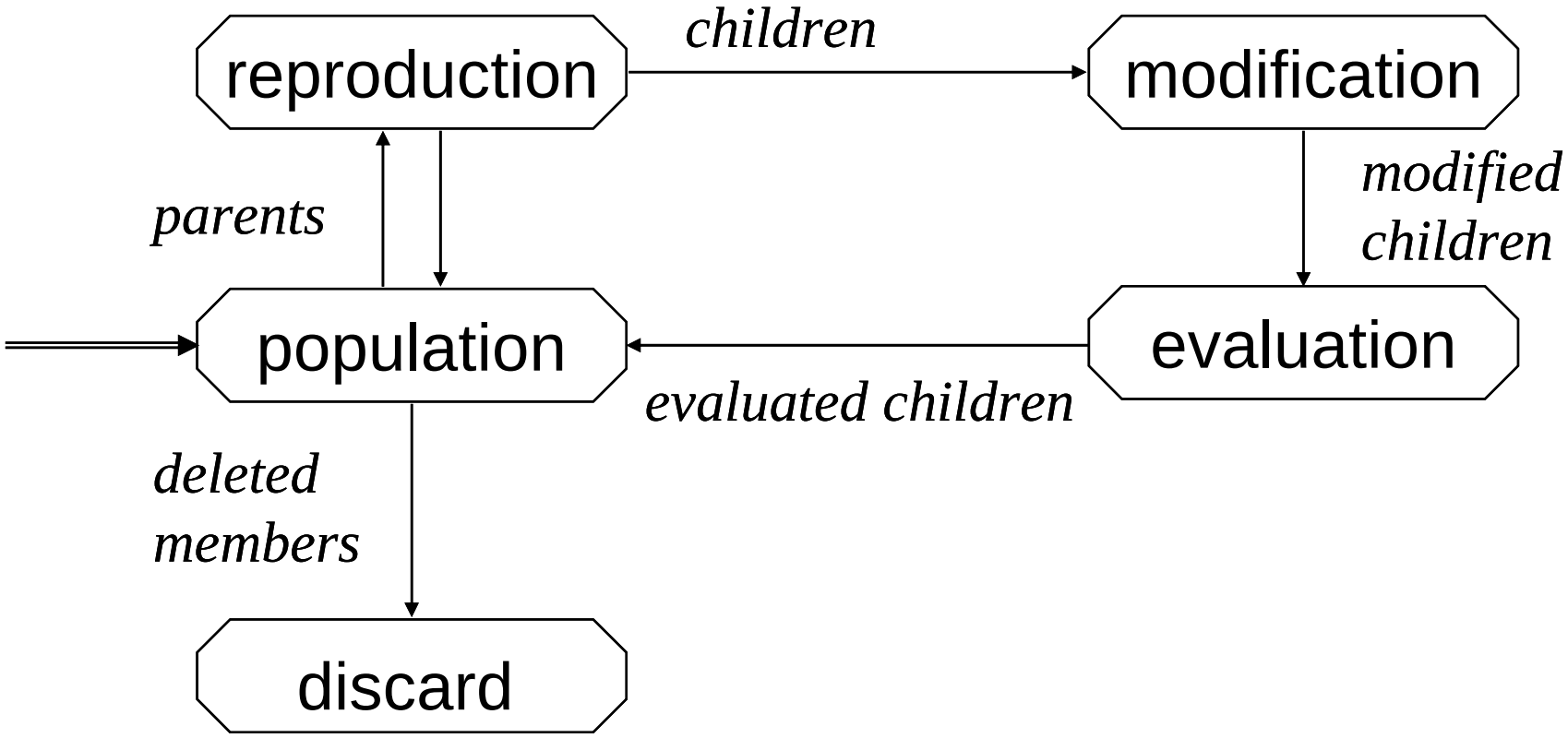
- Directed search algorithms based on the mechanics of biological evolution
- Developed by John Holland, University of Michigan (1970s)
  - To understand the adaptive processes of natural systems
  - To design artificial systems software that retains the robustness of natural systems
- Provide efficient,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

Components of a GA

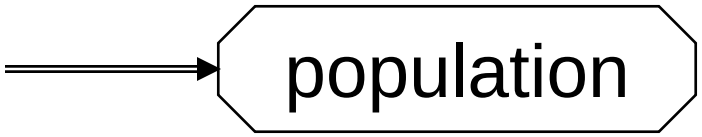
A problem to solve, and ...

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

GA Cycle of Reproduction



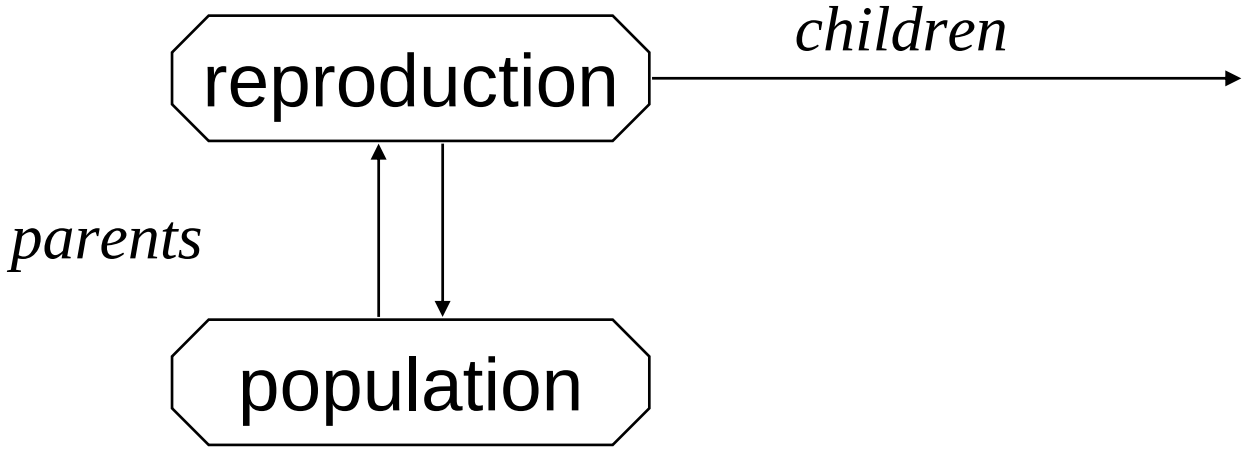
Population



Chromosomes could be

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

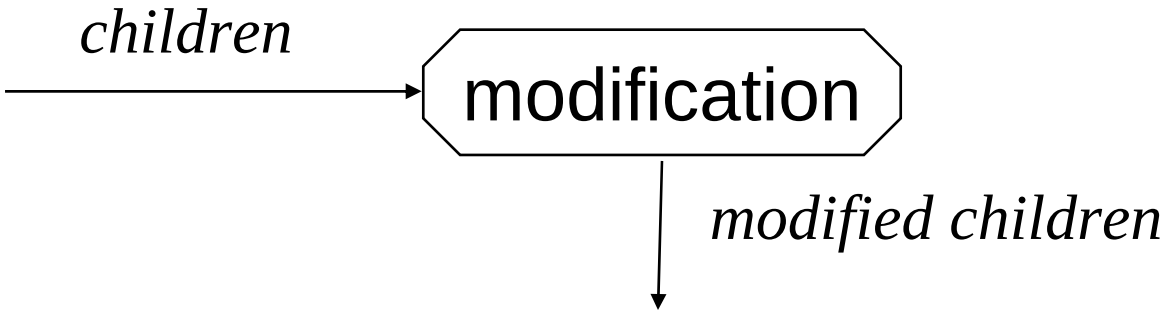
Reproduction



Parents are selected at random

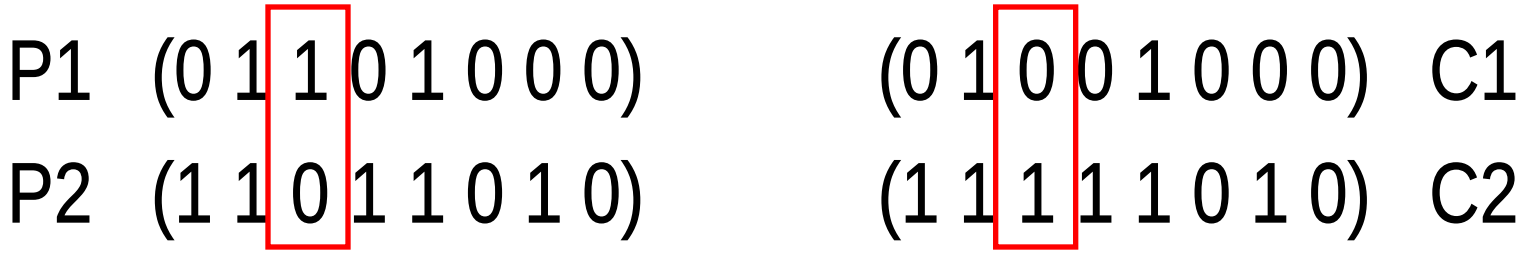
with selection chances biased in relation to chromosome evaluations

Chromosome Modification



- Modifications are stochastically triggered
- Operator types are
  - Crossover (recombination)
  - Mutation

Crossover: Recombination



- Crossover is a critical feature of genetic algorithms
  - Greatly accelerates search early in evolution of a population
  - Leads to effective combination of schemata (subsolutions on different chromosomes)

Mutation: Local Modification

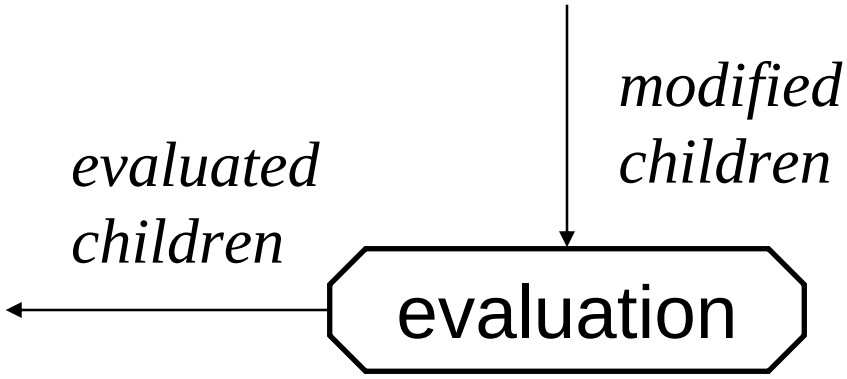
Before: (0 0 1 1 0 1 1 0)

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

Before: (1.38 -69.4 326.44 0.1)

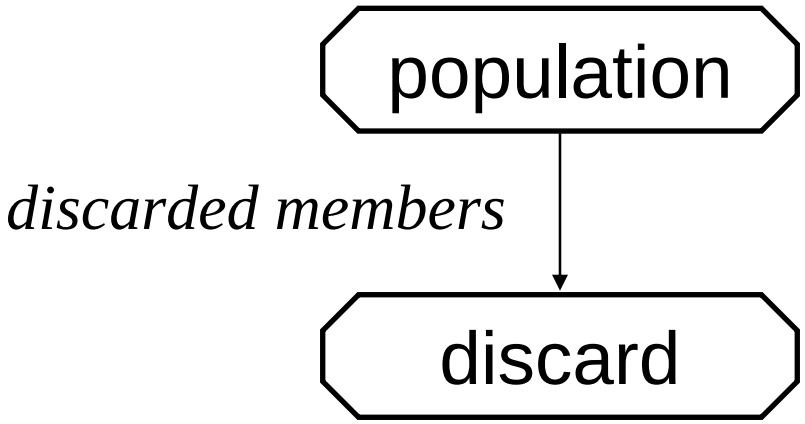
After: (1.38 -67.5 326.44 0.1)

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

Simple Example

The Traveling Salesman Problem

Representation is an ordered list of city numbers

1) London	3) Dunedin	5) Beijing	7) Tokyo
2) Venice	4) Singapore	6) Phoenix	8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Crossover Example

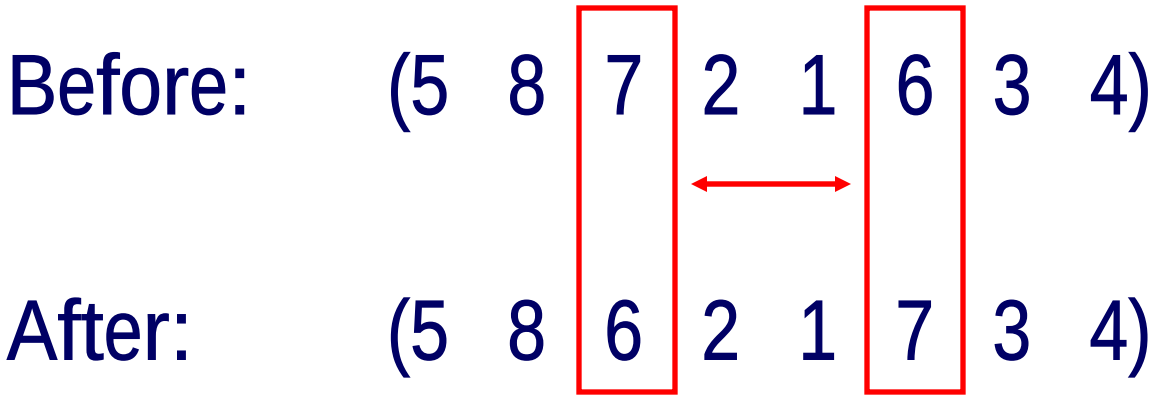
Crossover combines inversion and recombination:

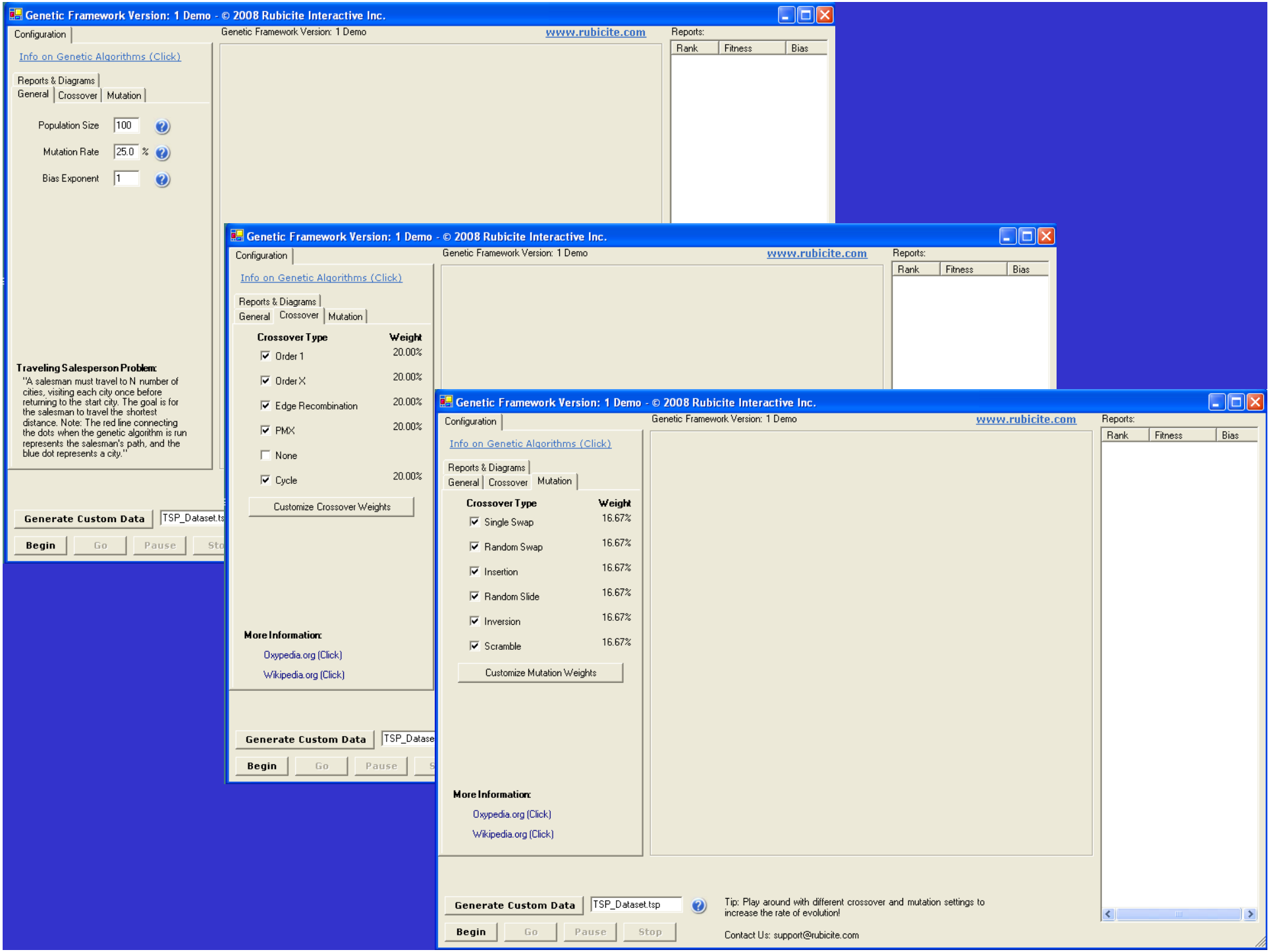
Parent1	(3	5	7	2	1	6	4	8)
Parent2	(2	5	7	6	8	1	3	4)
Child	(5	8	7	2	1	6	3	4)

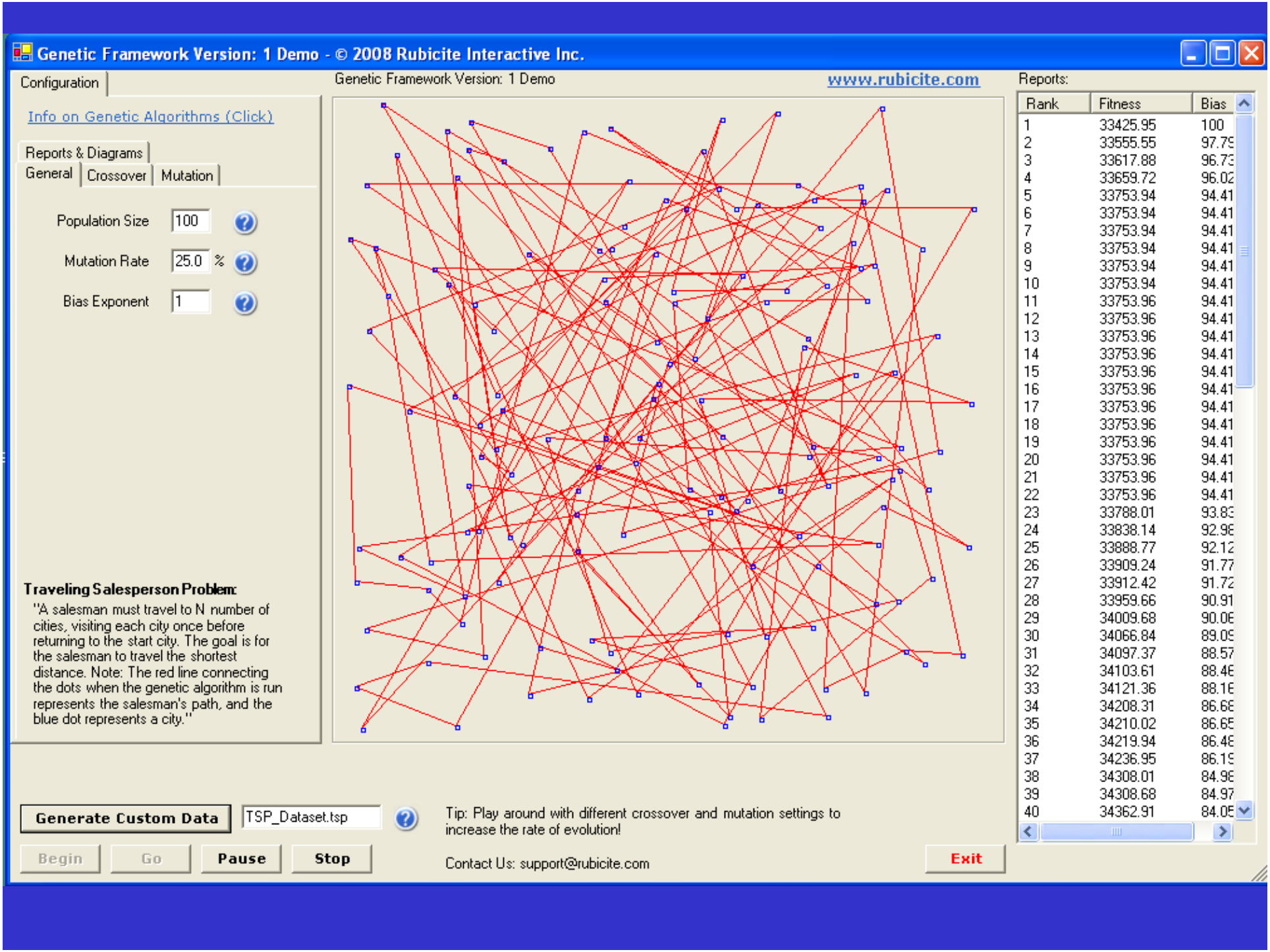
This operator is called the *Order1* crossover.

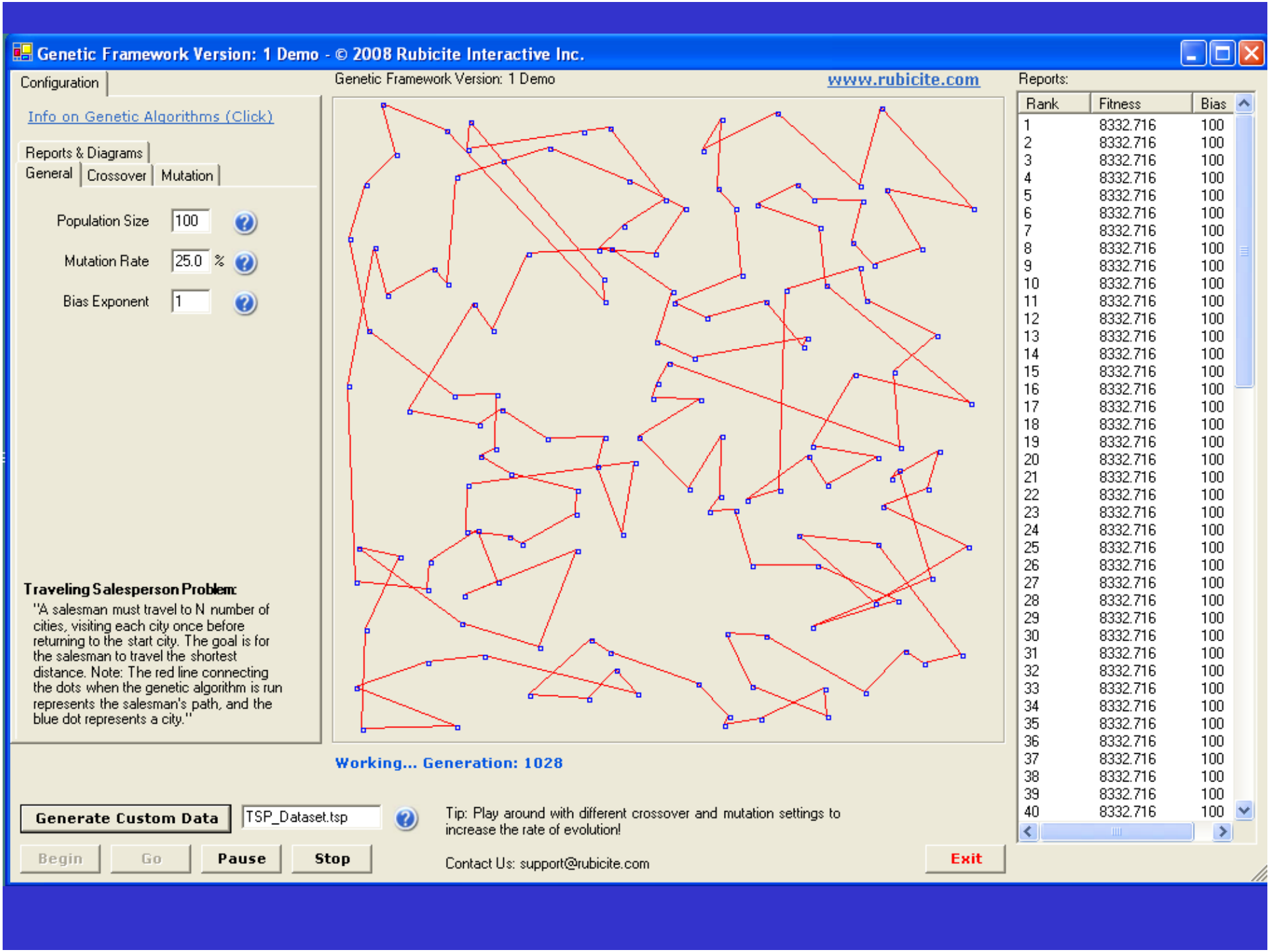
Mutation Example

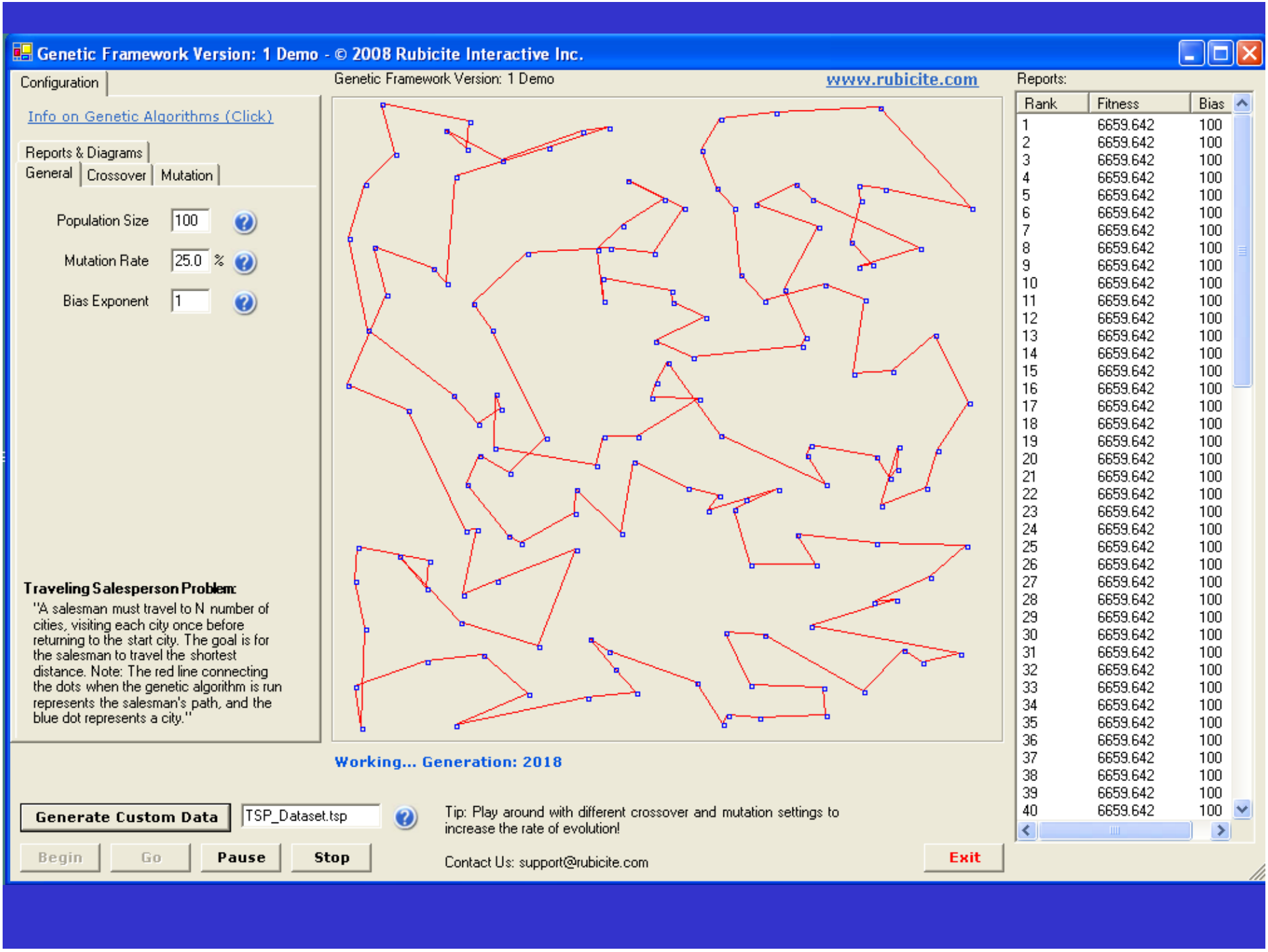
Mutation involves reordering of the list

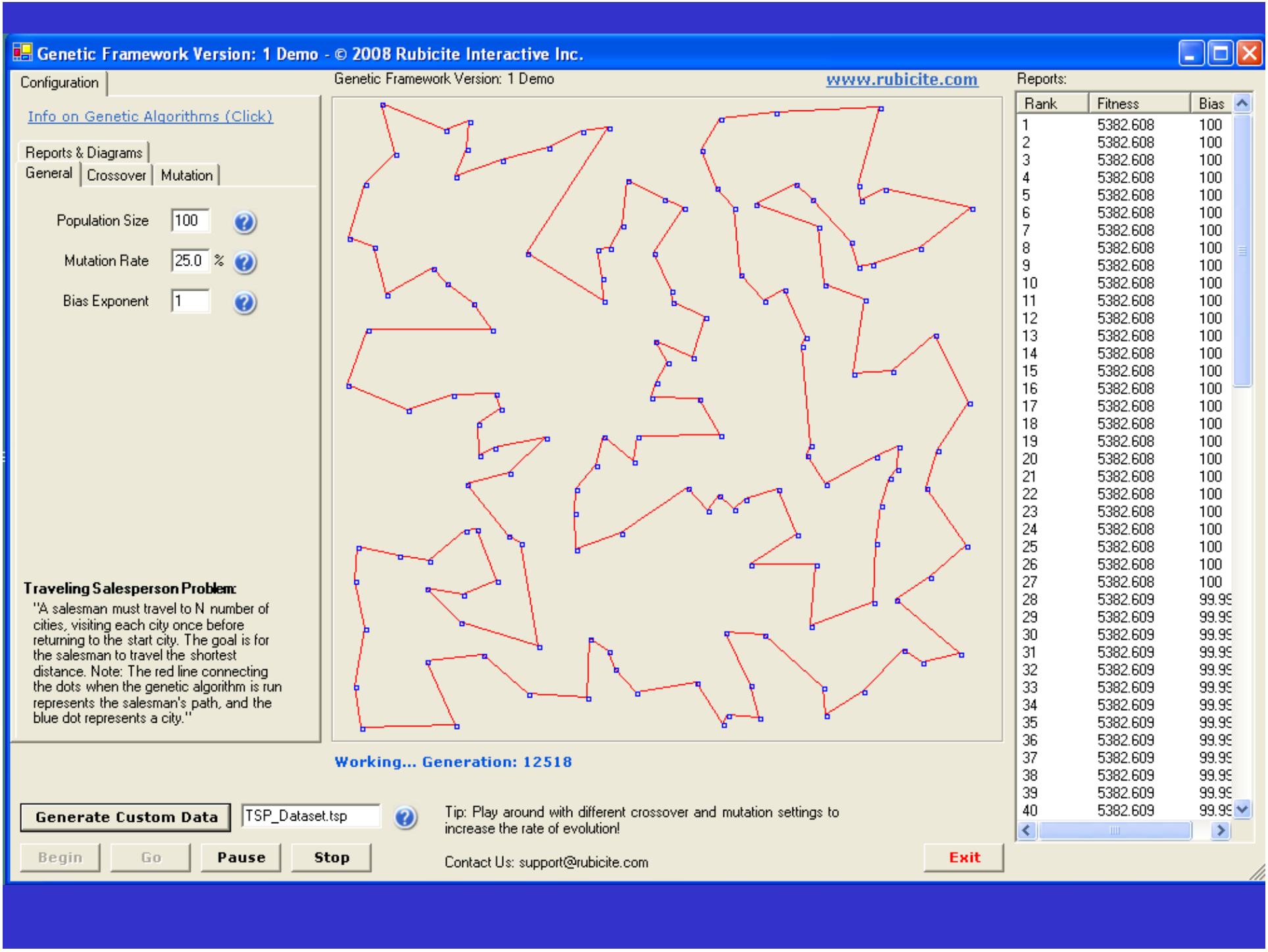


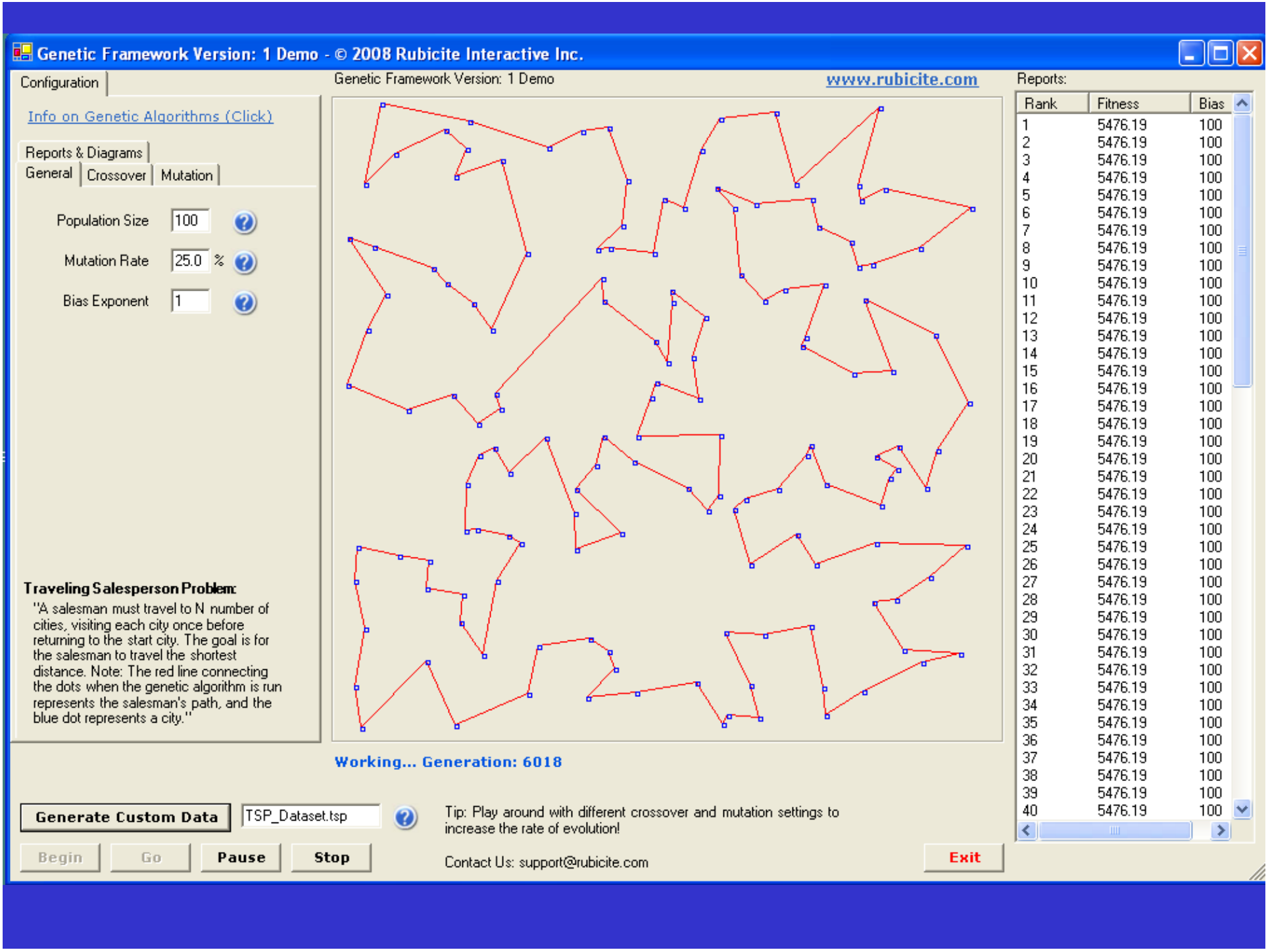

Genetic Framework Version: 1 Demo - © 2008 Rubicite Interactive Inc.
Configuration
Info on Genetic Algorithms (Click)
Reports & Diagrams
General
Crossover
Mutation
Population Size
100
Mutation Rate
25.0
%
Bias Exponent
1
Traveling Salesperson Problem:
"A salesman must travel to N number of cities, visiting each city once before returning to the start city. The goal is for the salesman to travel the shortest distance. Note: The red line connecting the dots when the genetic algorithm is run represents the salesman's path, and the blue dot represents a city."
Generate Custom Data
TSP_Dataset.ts
Begin
Go
Pause
Genetic Framework Version: 1 Demo
www.rubicite.com
Reports:
Rank
Fitness
Bias
Crossover Type
Weight
Order 1
20.00%
Order X
20.00%
Edge Recombination
20.00%
PMX
20.00%
None
Cycle
20.00%
Customize Crossover Weights
More Information:
Oxypedia.org (Click)
Wikipedia.org (Click)
TSP_Dataset.tsp
Stop
Single Swap
16.67%
Random Swap
16.67%
Insertion
16.67%
Random Slide
16.67%
Inversion
16.67%
Scramble
16.67%
Customize Mutation Weights
Tip: Play around with different crossover and mutation settings to increase the rate of evolution!
Contact Us: support@rubicite.com


Genetic Framework Version: 1 Demo - © 2008 Rubicite Interactive Inc.
Configuration
Genetic Framework Version: 1 Demo
www.rubicite.com
Reports:
Info on Genetic Algorithms (Click)
Reports & Diagrams
General
Crossover
Mutation
Population Size
100
Mutation Rate
25.0
%
Bias Exponent
1
Traveling Salesperson Problem:
"A salesman must travel to N number of cities, visiting each city once before returning to the start city. The goal is for the salesman to travel the shortest distance. Note: The red line connecting the dots when the genetic algorithm is run represents the salesman's path, and the blue dot represents a city."
Rank
Fitness
Bias
1 33425.95 100
2 33555.55 97.7
3 33617.88 96.7
4 33659.72 96.0
5 33753.94 94.41
6 33753.94 94.41
7 33753.94 94.41
8 33753.94 94.41
9 33753.94 94.41
10 33753.94 94.41
11 33753.96 94.41
12 33753.96 94.41
13 33753.96 94.41
14 33753.96 94.41
15 33753.96 94.41
16 33753.96 94.41
17 33753.96 94.41
18 33753.96 94.41
19 33753.96 94.41
20 33753.96 94.41
21 33753.96 94.41
22 33753.96 94.41
23 33788.01 93.8
24 33838.14 92.9
25 33888.77 92.1
26 33909.24 91.77
27 33912.42 91.72
28 33959.66 90.91
29 34009.68 90.0
30 34066.84 89.0
31 34097.37 88.57
32 34103.61 88.4
33 34121.36 88.1
34 34208.31 86.6
35 34210.02 86.6
36 34219.94 86.4
37 34236.95 86.1
38 34308.01 84.9
39 34308.68 84.97
40 34362.91 84.0
Generate Custom Data
TSP_Dataset.tsp
Tip: Play around with different crossover and mutation settings to increase the rate of evolution!
Begin
Go
Pause
Stop
Contact Us: support@rubicite.com
Exit


Genetic Framework Version: 1 Demo - © 2008 Rubicite Interactive Inc.
Configuration
Genetic Framework Version: 1 Demo
www.rubicite.com
Reports:
Info on Genetic Algorithms (Click)
Reports & Diagrams
General
Crossover
Mutation
Population Size
100
Mutation Rate
25.0
%
Bias Exponent
1
Traveling Salesperson Problem:
"A salesman must travel to N number of cities, visiting each city once before returning to the start city. The goal is for the salesman to travel the shortest distance. Note: The red line connecting the dots when the genetic algorithm is run represents the salesman's path, and the blue dot represents a city."
Rank
Fitness
Bias
1 8332.716 100
2 8332.716 100
3 8332.716 100
4 8332.716 100
5 8332.716 100
6 8332.716 100
7 8332.716 100
8 8332.716 100
9 8332.716 100
10 8332.716 100
11 8332.716 100
12 8332.716 100
13 8332.716 100
14 8332.716 100
15 8332.716 100
16 8332.716 100
17 8332.716 100
18 8332.716 100
19 8332.716 100
20 8332.716 100
21 8332.716 100
22 8332.716 100
23 8332.716 100
24 8332.716 100
25 8332.716 100
26 8332.716 100
27 8332.716 100
28 8332.716 100
29 8332.716 100
30 8332.716 100
31 8332.716 100
32 8332.716 100
33 8332.716 100
34 8332.716 100
35 8332.716 100
36 8332.716 100
37 8332.716 100
38 8332.716 100
39 8332.716 100
40 8332.716 100
Working... Generation: 1028
Generate Custom Data
TSP_Dataset.tsp
Tip: Play around with different crossover and mutation settings to increase the rate of evolution!
Begin
Go
Pause
Stop
Contact Us: support@rubicite.com
Exit


Genetic Framework Version: 1 Demo - © 2008 Rubicite Interactive Inc.
Configuration
Genetic Framework Version: 1 Demo
www.rubicite.com
Reports:
Info on Genetic Algorithms (Click)
Reports & Diagrams
General
Crossover
Mutation
Population Size
100
Mutation Rate
25.0
%
Bias Exponent
1
Traveling Salesperson Problem:
"A salesman must travel to N number of cities, visiting each city once before returning to the start city. The goal is for the salesman to travel the shortest distance. Note: The red line connecting the dots when the genetic algorithm is run represents the salesman's path, and the blue dot represents a city."
Rank
Fitness
Bias
1 6659.642 100
2 6659.642 100
3 6659.642 100
4 6659.642 100
5 6659.642 100
6 6659.642 100
7 6659.642 100
8 6659.642 100
9 6659.642 100
10 6659.642 100
11 6659.642 100
12 6659.642 100
13 6659.642 100
14 6659.642 100
15 6659.642 100
16 6659.642 100
17 6659.642 100
18 6659.642 100
19 6659.642 100
20 6659.642 100
21 6659.642 100
22 6659.642 100
23 6659.642 100
24 6659.642 100
25 6659.642 100
26 6659.642 100
27 6659.642 100
28 6659.642 100
29 6659.642 100
30 6659.642 100
31 6659.642 100
32 6659.642 100
33 6659.642 100
34 6659.642 100
35 6659.642 100
36 6659.642 100
37 6659.642 100
38 6659.642 100
39 6659.642 100
40 6659.642 100
Working... Generation: 2018
Generate Custom Data
TSP_Dataset.tsp
Tip: Play around with different crossover and mutation settings to increase the rate of evolution!
Begin
Go
Pause
Stop
Contact Us: support@rubicite.com
Exit


Genetic Framework Version: 1 Demo - © 2008 Rubicite Interactive Inc.
Configuration
Genetic Framework Version: 1 Demo
www.rubicite.com
Reports:
Info on Genetic Algorithms (Click)
Reports & Diagrams
General
Crossover
Mutation
Population Size
100
Mutation Rate
25.0
%
Bias Exponent
1
Traveling Salesperson Problem:
"A salesman must travel to N number of cities, visiting each city once before returning to the start city. The goal is for the salesman to travel the shortest distance. Note: The red line connecting the dots when the genetic algorithm is run represents the salesman's path, and the blue dot represents a city."
Rank
Fitness
Bias
1 5382.608 100
2 5382.608 100
3 5382.608 100
4 5382.608 100
5 5382.608 100
6 5382.608 100
7 5382.608 100
8 5382.608 100
9 5382.608 100
10 5382.608 100
11 5382.608 100
12 5382.608 100
13 5382.608 100
14 5382.608 100
15 5382.608 100
16 5382.608 100
17 5382.608 100
18 5382.608 100
19 5382.608 100
20 5382.608 100
21 5382.608 100
22 5382.608 100
23 5382.608 100
24 5382.608 100
25 5382.608 100
26 5382.608 100
27 5382.608 100
28 5382.609 99.9
29 5382.609 99.9
30 5382.609 99.9
31 5382.609 99.9
32 5382.609 99.9
33 5382.609 99.9
34 5382.609 99.9
35 5382.609 99.9
36 5382.609 99.9
37 5382.609 99.9
38 5382.609 99.9
39 5382.609 99.9
40 5382.609 99.9
Working... Generation: 12518
Generate Custom Data
TSP_Dataset.tsp
Tip: Play around with different crossover and mutation settings to increase the rate of evolution!
Begin
Go
Pause
Stop
Contact Us: support@rubicite.com
Exit


Genetic Framework Version: 1 Demo - © 2008 Rubicite Interactive Inc.
Configuration
Genetic Framework Version: 1 Demo
www.rubicite.com
Reports:
Info on Genetic Algorithms (Click)
Reports & Diagrams
General
Crossover
Mutation
Population Size
100
Mutation Rate
25.0
%
Bias Exponent
1
Traveling Salesperson Problem:
"A salesman must travel to N number of cities, visiting each city once before returning to the start city. The goal is for the salesman to travel the shortest distance. Note: The red line connecting the dots when the genetic algorithm is run represents the salesman's path, and the blue dot represents a city."
Rank
Fitness
Bias
1 5476.19 100
2 5476.19 100
3 5476.19 100
4 5476.19 100
5 5476.19 100
6 5476.19 100
7 5476.19 100
8 5476.19 100
9 5476.19 100
10 5476.19 100
11 5476.19 100
12 5476.19 100
13 5476.19 100
14 5476.19 100
15 5476.19 100
16 5476.19 100
17 5476.19 100
18 5476.19 100
19 5476.19 100
20 5476.19 100
21 5476.19 100
22 5476.19 100
23 5476.19 100
24 5476.19 100
25 5476.19 100
26 5476.19 100
27 5476.19 100
28 5476.19 100
29 5476.19 100
30 5476.19 100
31 5476.19 100
32 5476.19 100
33 5476.19 100
34 5476.19 100
35 5476.19 100
36 5476.19 100
37 5476.19 100
38 5476.19 100
39 5476.19 100
40 5476.19 100
Working... Generation: 6018
Generate Custom Data
TSP_Dataset.tsp
Tip: Play around with different crossover and mutation settings to increase the rate of evolution!
Begin
Go
Pause
Stop
Contact Us: support@rubicite.com
Exit

GA Applications

- Automated design, including research on composite material design and design of automotive components for crashworthiness, weight savings, and other characteristics.
- Automated design of mechatronic systems using bond graphs and genetic programming
- Automated design of industrial equipment using catalogs of exemplar lever patterns.
- Automated design of sophisticated trading systems in the financial sector.
- Building phylogenetic trees
- Chemical kinetics (gas and solid phases)
- Configuration applications, particularly physics applications of optimal molecule configurations
- Container loading optimization
- Code-breaking, using the GA to search large solution spaces of ciphers for the one correct decryption
- Topology
- Electronic circuit design
- File allocation for a distributed system
- Game Theory: Equilibrium Resolution.
- Gene expression profiling analysis
- Linguistic analysis, including Grammar induction and other aspects of Natural language processing (NLP)
- Mobile communications infrastructure optimization
- Molecular structure optimization (Chemistry)
- Multiple criteria production scheduling
- Multiple population topologies and interchange methodologies

GA Applications

- Mutation testing
- Neural networks
- Optimization of data compression systems, for example using wavelets
- Music
- Protein folding and protein/ligand docking
- Representing rational agents in economic models such as the cobweb model
- Bioinformatics: RNA structure prediction
- Bioinformatics: [Multiple Sequence Alignment]
- Bioinformatics: Multiple sequence alignment
- Scheduling applications, including job-shop scheduling. Selection of optimal mathematical model to describe biological systems.
- Software engineering
- Solving the machine-component grouping problem required for cellular manufacturing systems allocation and strategies.
- Timetabling problems, such as designing a non-conflicting class timetable for a large university
- Training artificial neural networks when pre-classified training examples are not readily obtainable
- Traveling Salesman Problem
- Finding hardware bugs
- Wireless Sensor/Ad-hoc Networks
- Data Center/Server Farm

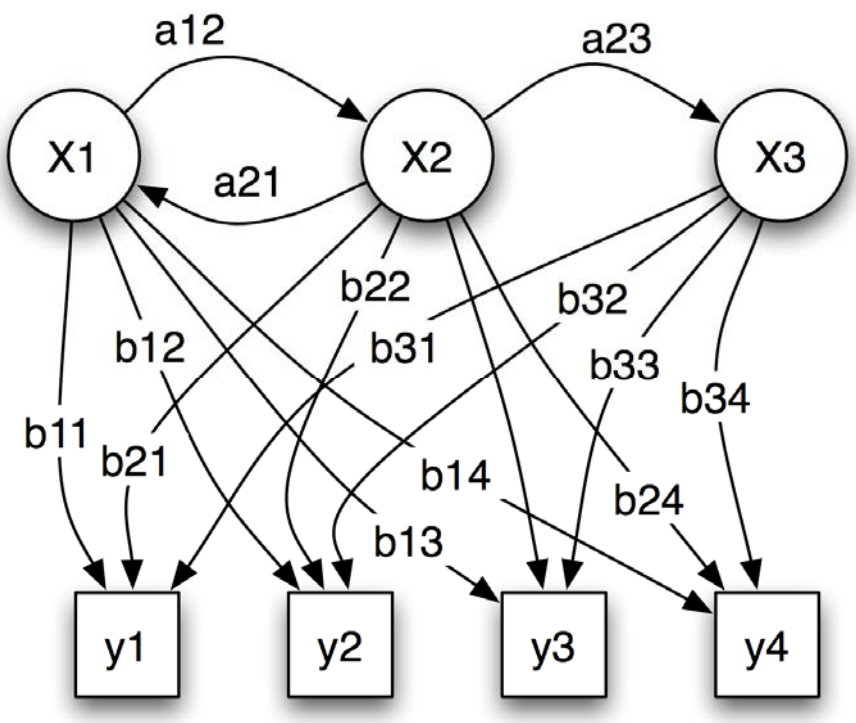
Related Techniques

- Tabu Search: SA-like. Search by testing *multiple* mutations with a tabu list (unnacceptable solutions). Choose the minimum energy solution.
- Ant Colony Optimization: To solve a path problem, first find *a solution*, then search the space in parallel to drop longer segments to get to the shortest path.
- Bacteriologic Algorithms: Based on GA. To find solutions good for *a population,* not a single best solution.
- Memetic Algorithms: Based on the concept of Universal Darwinism due to Dawkins which says principles of evolution are applicable not only to genes but all complex systems that exhibit inheritance, variation, and selection. Memes, unlike genes, can adapt themselves. A memetic algorithm performs local search during the evolutionary cycle.

Related Techniques

- Cultural Algorithm: Similar to GA, with an additional knowledge component, called the belief space.
- Cross-Entropy Method: Generate candidate solutions via a parameterized probability distribution. The parameters are updated via cross-entropy minimization to generate better samples in the next iteration.
- Others: Evolution Strategies, Evolutionary Programming, Extremal Optimization, Gaussian Adaptation, Genetic Programming, Interactive Evolutionary Algorithms, Harmony Search, Reactive Search Optimization

Hidden Markov Models



- Markov model whose states are observable only through probabilistic observations
- Model observation sequences with short-term stationary characteristics, occasional changes
- Originally used in speech recognition, later applied to biological sequence analysis

Problems and Solutions

1. Given an observation sequence and the model, what is the probability that the observation sequence was generated by the model?

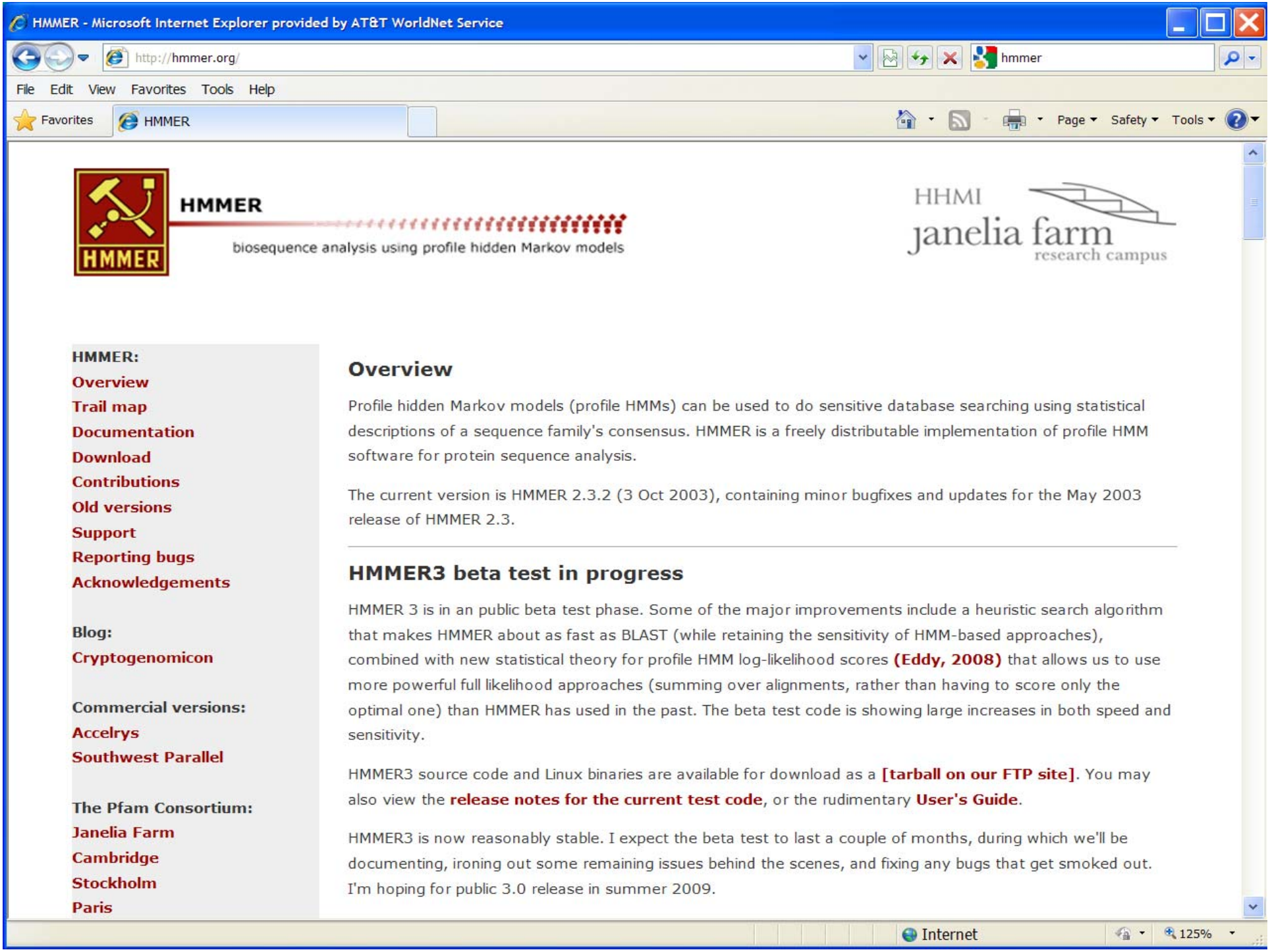
   Forward-Backward Algorithm

2. Given an observation sequence how to choose a state sequence optimum in some sense?

   Viterbi Algorithm

3. How adjust model parameters to maximize probability of observations given the model?

   Baum-Welch Algorithm, etc.


HMMER - Microsoft Internet Explorer provided by AT&T WorldNet Service
http://hmmer.org/
hmmer
File Edit View Favorites Tools Help
Favorites
HMMER
Page Safety Tools
HMMER
biosequence analysis using profile hidden Markov models
HHMI
janelia farm
research campus
HMMER:
Overview
Trail map
Documentation
Download
Contributions
Old versions
Support
Reporting bugs
Acknowledgements
Blog:
Cryptogenomicon
Commercial versions:
Accelrys
Southwest Parallel
The Pfam Consortium:
Janelia Farm
Cambridge
Stockholm
Paris
Overview
Profile hidden Markov models (profile HMMs) can be used to do sensitive database searching using statistical descriptions of a sequence family's consensus. HMMER is a freely distributable implementation of profile HMM software for protein sequence analysis.
The current version is HMMER 2.3.2 (3 Oct 2003), containing minor bugfixes and updates for the May 2003 release of HMMER 2.3.
HMMER3 beta test in progress
HMMER 3 is in an public beta test phase. Some of the major improvements include a heuristic search algorithm that makes HMMER about as fast as BLAST (while retaining the sensitivity of HMM-based approaches), combined with new statistical theory for profile HMM log-likelihood scores (Eddy, 2008) that allows us to use more powerful full likelihood approaches (summing over alignments, rather than having to score only the optimal one) than HMMER has used in the past. The beta test code is showing large increases in both speed and sensitivity.
HMMER3 source code and Linux binaries are available for download as a [tarball on our FTP site]. You may also view the release notes for the current test code, or the rudimentary User's Guide.
HMMER3 is now reasonably stable. I expect the beta test to last a couple of months, during which we'll be documenting, ironing out some remaining issues behind the scenes, and fixing any bugs that get smoked out. I'm hoping for public 3.0 release in summer 2009.
Internet
125%

4768–4778 Nucleic Acids Research, 1994, Vol. 22, No. 22

A hidden Markov model that finds genes in *E.coli* DNA

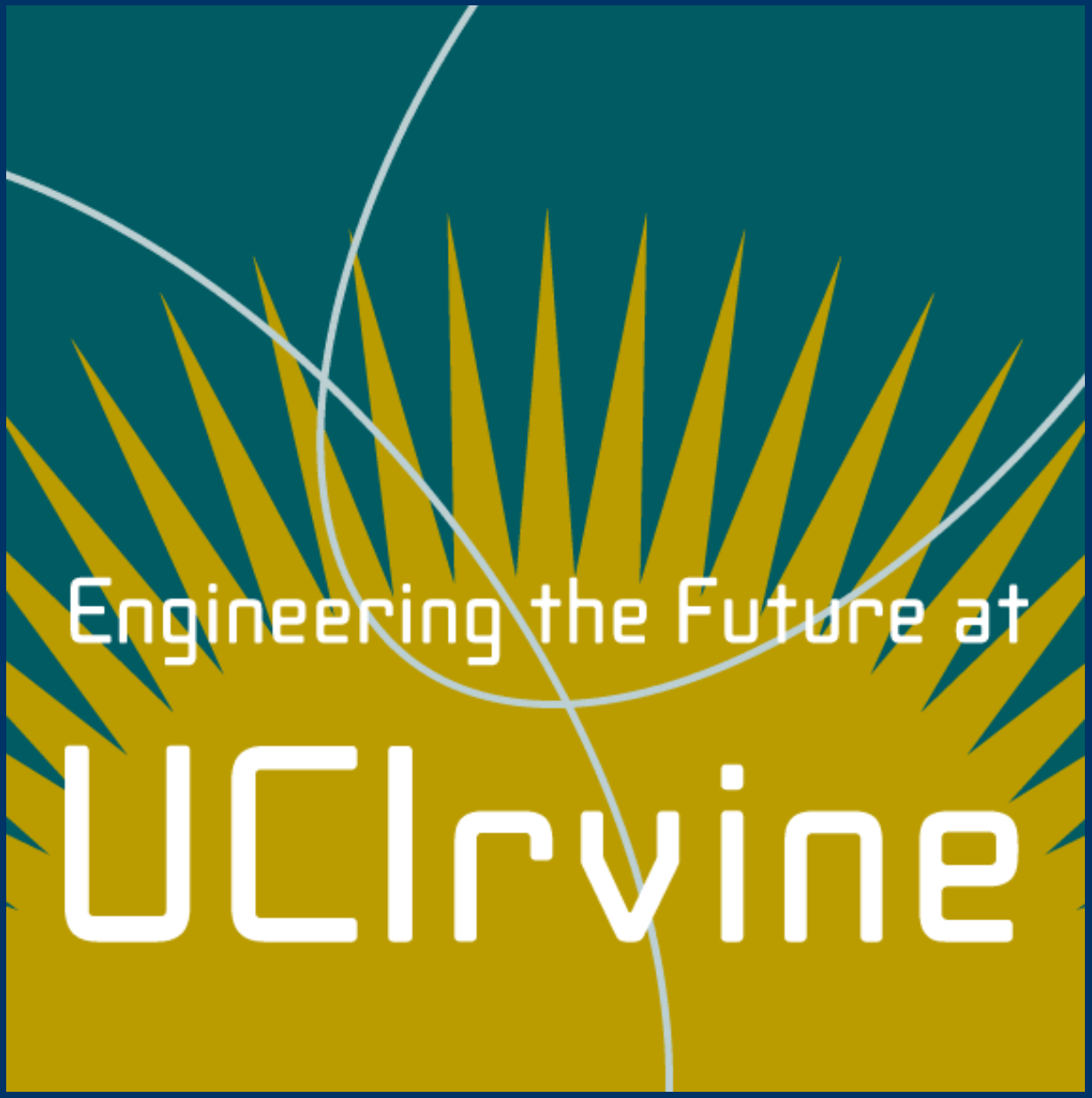
Anders Krogh, I.Saira Mian[1] and David Haussler[2]*
Nordita, Blegdamsvej 17, DK-2100 Copenhagen, Denmark, [1]Sinsheimer Laboratories, University of California, Santa Cruz, CA 95064 and [2]Computer and Information Sciences, University of California, Santa Cruz, CA 95064, USA

Received June 21, 1994; Revised and Accepted September 28, 1994

ABSTRACT

A hidden Markov model (HMM) has been developed to find protein coding genes in *E.coli* DNA using *E.coli* genome DNA sequence from the EcoSeq6 database maintained by Kenn Rudd. This HMM includes states that model the codons and their frequencies in *E.coli* genes, as well as the patterns found in the intergenic region, including repetitive extragenic palindromic sequences and the Shine – Delgarno motif. To account for potential sequencing errors and or frameshifts in raw genomic DNA sequence, it allows for the (very unlikely) possiblity of insertions and deletions of individual nucleotides within a codon. The parameters of the HMM are estimated using approximately one million nucleotides of annotated DNA in EcoSeq6 and the model tested on a disjoint set of contigs containing about 325,000 nucleotides. The HMM finds the exact locations of about 80% of the known *E.coli* genes, and approximate locations for about 10%. It also finds several potentially new genes, and locates several places were insertion or deletion errors/and or frameshifts may be present in the contigs.

non-coding region (reviewed in [10]). Staden and McLachlan [11,3] proposed deviation from average codon usage as a way of determining the probability that the window is coding or not. Later, Gribskov *et al.* [12] used a similar measure as a part of their 'codon preference plot', but their measure did not require the knowledge of an average codon usage from other sources. Most other scoring methods are related to codon usage in some way [13,3]. Recently, neural networks [4,14,15,16] and Markov chains [17,18,19] have been used to analyze coding (and non-coding) regions. In particular, the program GeneMark [20] finds genes in *E.coli* DNA using a Markov model for the coding region related to the one discussed here, and a very simple Markov model for the non-coding regions. Whether looking for signals in the DNA or using window scoring, there remains the problem of combining all the scores and/or signals detected in a given contig to produce a coherent 'parse' into genes separated by intergenic regions. The output of this final parsing step could be a list of genes, each represented by its begin and end position within the contig. Snyder and Stormo have recently proposed an elegant dynamic programming method to accomplish this final step [21]. Other more linguistically motivated approaches to this kind of sequence parsing problem are described in [22,23,24,25].


Engineering the Future at
UCIrvine