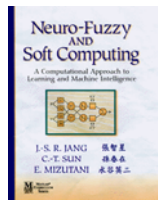
Chapter 7: Derivative-Free Optimization

- ✶ Introduction (7.1)
- ✶ Genetic Algorithms (GA) (7.2)
- ✶ Simulated Annealing (SA) (7.3)
- ✶ Random Search (7.4)
- ✶ Downhill Simplex Search (DSS) (7.5)



Jyh-Shing Roger Jang et al., *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, First Edition, Prentice Hall, 1997

Introduction (7.1)

✶Four of the most popular derivative-free optimization are:

- Genetic Algorithms (GA)
- Simulated Annealing (SA)
- Random search method
- Downhill simplex search

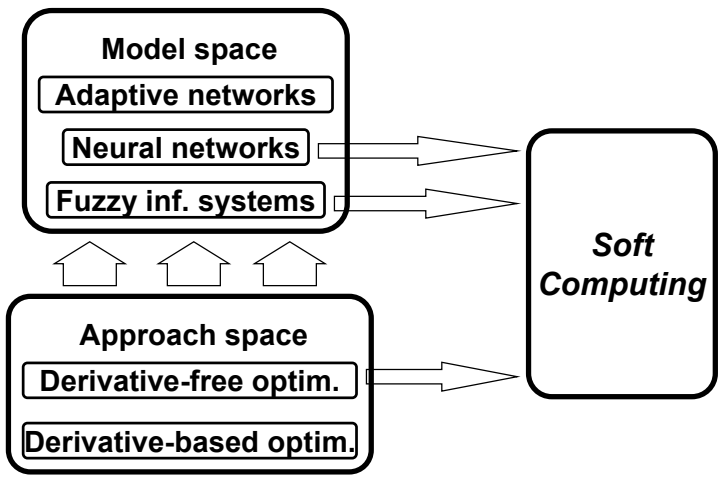
Introduction (7.1) (cont.)

✶ They are characterized by the following:

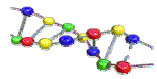
- They do not require functional derivative information, they rely exclusively on repeated evaluations of the objective function, and the subsequent search direction after each evaluation follows certain heuristic guidelines
- They search a global optimum & are slower than derivative-based optimization methods
- They are based on simple intuitive concepts that pertain to nature's wisdom including evolution & thermodynamics
- They are flexible since they do not require any differentiability of the objective function

Introduction (7.1) (cont.)

✶The big picture

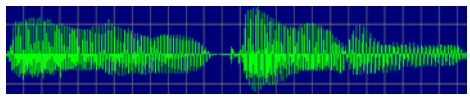


Genetic Algorithms (7.2)



✶ Motivation

– Look at what evolution brings us?
  - Vision
  - Hearing
  - Smelling
  - Taste
  - Touch
  - Learning and reasoning



– Can we emulate the evolutionary process with today's fast computers?

Genetic Algorithms (7.2) (cont.)

✶ Terminology:

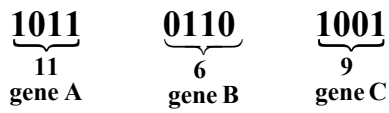
– Fitness function
– Population
– Encoding schemes
– Selection
– Crossover
– Mutation
– Elitism

Genetic Algorithms (7.2) (cont.)

✶GA Principle

– Each point in a parameter space is encoded into binary strings called chromosomes

– Example: The 3D point (11,6,9) can be represented as a concatenated binary string (or chromosome)



Genetic Algorithms (7.2) (cont.)

✶GA Principle (cont.)

– Each point is associated with a fitness value (function value)

– GA keeps a set of points as a population (or gene pool), which is then evolved repeatedly toward a better overall fitness value

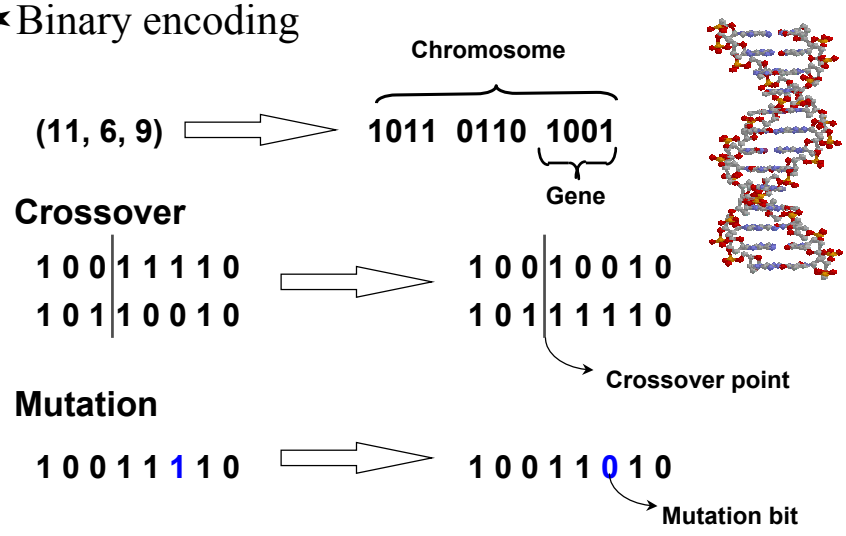
– In each generation, the GA constructs a new population using genetic operators such as crossover & mutation: members with higher fitness values are more likely to survive & to participate in mating (crossover) operations

Genetic Algorithms (7.2) (cont.)

- GA Principle (cont.)
  - After a number of generations, the populations contains members with better fitness values (Darwin: random mutation & natural selection)
  - It is a population-based optimization that upgrade entire populations for better performance

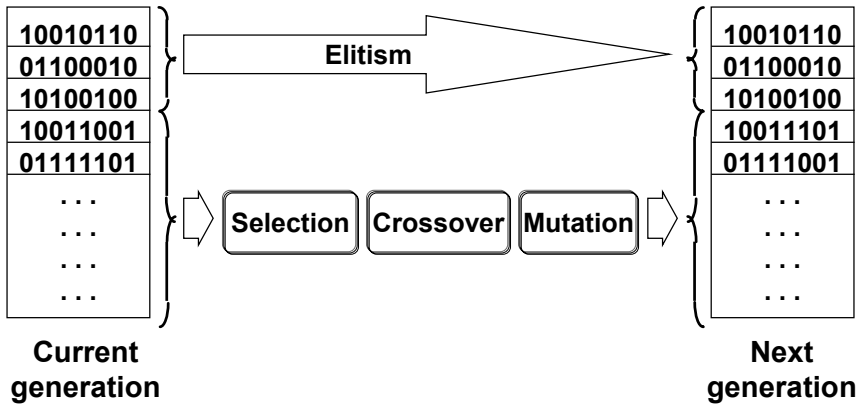
Genetic Algorithms (7.2) (cont.)

- Binary encoding



Genetic Algorithms (7.2) (cont.)

✶ Flowchart



Genetic Algorithms (7.2) (cont.)

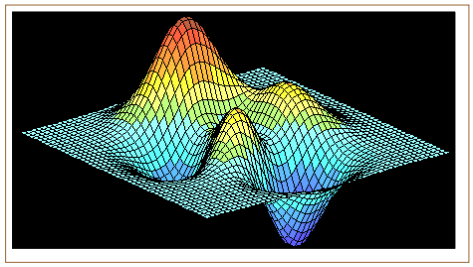
✶ Algorithm:

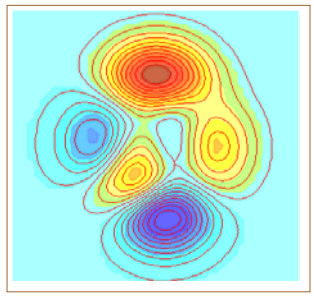
- Step 1: Initialize a population with randomly generated individuals and evaluate the fitness value of each individual
- Step 2:
  (a) Select 2 members from the population with probabilities proportional to their fitness values
  (b) Apply crossover with a probability equal to the crossover rate
  (c) Apply mutation with a probability equal to the mutation rate
  (d) Repeat (a) to (d) until enough members are generated to form the next generation
- Step 3: Repeat step 2 & 3 until a stopping criterion is met

Genetic Algorithms (7.2) (cont.)

- 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).



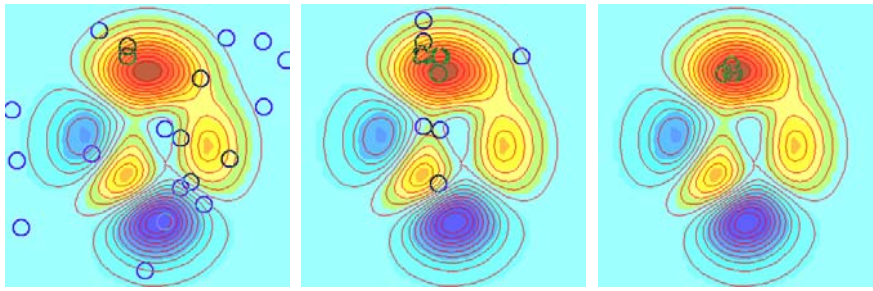


Genetic Algorithms (7.2) (cont.)

- Use of GA on the domain [-3,3] * [-3,3]
- Use 8-bit binary coding for each variable $\Rightarrow$ the search space is $2^8 * 2^8 = 65{,}536$
- Each generation contains 20 points (or individuals)
- Use of a simple one-point crossover scheme with rate = 1.0
- Use of a uniform mutation with rate = 0.01
- Apply elitism to keep the best individuals across generations

Genetic Algorithms (7.2) (cont.)

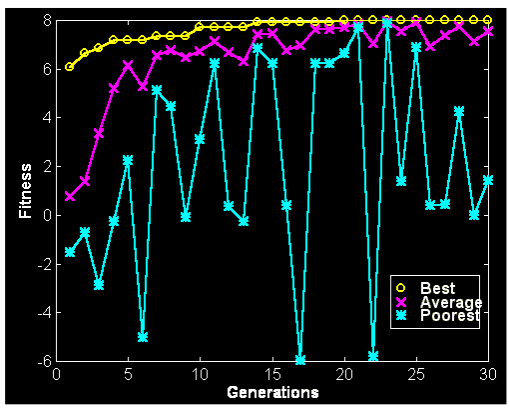
★GA process:



Initial population 5th generation 10th generation

Genetic Algorithms (7.2) (cont.)

★Performance profile



Simulated Annealing (SA) (7.3)

✶Introduction

- Introduced by Metropolis et al. in 1953 and adapted by Kirkpatrick (1983) in order to find an optimum solution to a large-scale combinatorial optimization problems
- It is a derivative–free optimization method
- SA was derived from physical characteristics of spin glasses

Simulated Annealing (cont.) (7.3)

✶Introduction (cont.)

- The principle behind SA is similar to what happens when metals are cooled at a controlled rate
- The slowly decrease of temperature allows the atoms in the molten metal to line themselves up an form a regular crystalline structure that possesses a low density & a low energy

Simulated Annealing (cont.) (7.3)

✶ Introduction (cont.)

- The value of an objective function that we intend to minimize corresponds to the energy in a thermodynamic system
- High temperatures corresponds to the case where high-mobility atoms can orient themselves with other non-local atoms & the energy can increase: function evaluation accepts new points with higher energy

Simulated Annealing (cont.) (7.3)

✶ Introduction (cont.)

- Low temperatures corresponds to the case where the low-mobility atoms can only orient themselves with local atoms & the energy state is not likely to increase: Function evaluation is performed only locally and points with higher energy are more & more refused
- The most important element of SA is the so-called annealing schedule (or cooling schedule) which express how fast the temperature is being lowered from high to low

Simulated Annealing (cont.) (7.3)

✶ Terminology:

– Objective function E(x): function to be optimized

– Move set: set of next points to explore

$\Delta E = x_{new} - x$ is a random variable with pdf = g(.,.)

– Generating function: pdf for selecting next point

$g(\Delta x, T) = (2\pi T)^{-n/2} \exp[-|| \Delta x||^2 / 2T]$

(n is the explored space dimension)

Simulated Annealing (cont.) (7.3)

✶ Terminology (cont.)

– Acceptance function $h(\Delta E, T)$: to determine if the selected point should be accepted or not. Usually

$h(\Delta E, T) = 1/(1+\exp(\Delta E/(cT))$.

– Annealing (cooling) schedule: schedule for reducing the temperature T

Simulated Annealing (cont.) (7.3)

✶ Basic step involved in a general SA method

- Step 1 [Initialization]: Choose an initial point x and a high temperature T and set the iteration count k to 1
- Step 2 [Evaluation]: Evaluate the objective function $E = f(x)$
- Step 3 [Exploration]: Select Δx with probability $g(\Delta x, T)$, and set $x_{new} = x + \Delta x$
- Step 4 [Reevaluation]: Compute the new value of the objective function $E_{new} = f(x_{new})$

Simulated Annealing (cont.) (7.3)

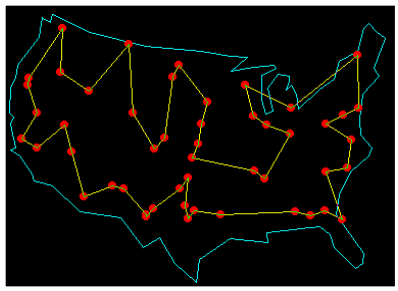
✶ Basic step involved in a general SA method (cont.)

- Step 5 [Acceptance]: Test if x_{new} is accepted or not by computing $h(\Delta E, T)$ where $\Delta E = E_{new}-E$
- Step 6: Lower the temperature according to the annealing schedule ($T = \eta T$; $0 < \eta < 1$)
- Step 7: $k = k + 1$, if k reaches the maximum iteration count, then stop otherwise go to step 3

Simulated Annealing (cont.) (7.3)

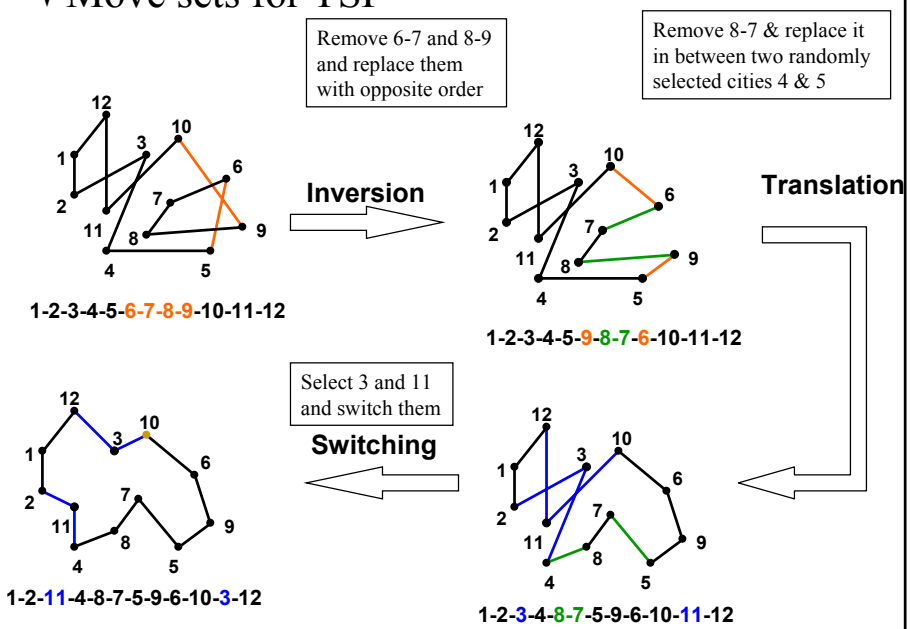
✶Example: Travel Salesperson Problem (TSP)

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



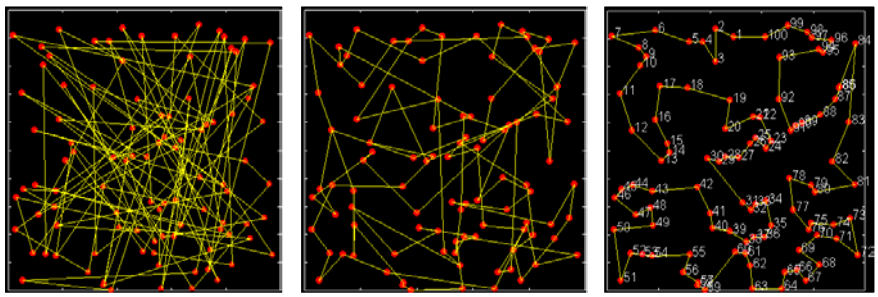
NP hard problem with [(n – 1)!] / 2 possible tours to explore

✶Move sets for TSP



Simulated Annealing (cont.) (7.3)

★A 100-city TSP using SA



Initial random path **During SA process** **Final path**

Random Search (7.4)

★This method explores the parameter space of an objective function sequentially in a random fashion to find the optimal point that maximizes the objective function

★This method is simple and converges to the global optimum surely on a compact set

Random Search (cont.) (7.4)

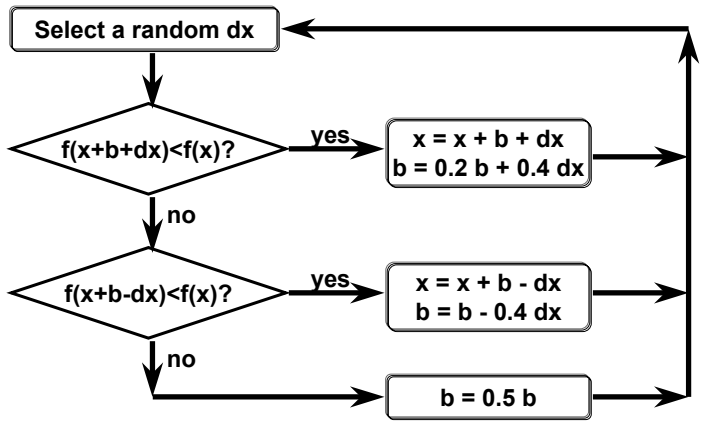
- The steps involved are:
  - Step 1: Choose an initial point x as the current point
  - Step 2: Add a random vector dx to the current point x & evaluate the objective function at x + dx
  - Step 3: if f(x + dx) < f(x) set x = x + dx
  - Step 4: Stop the process if max iteration is reacted otherwise go back to step 2 to find a new point

Random Search (cont.) (7.4)

- This method is blind since the search directions are guided by a random number generator, therefore an improved version is developed
- It is based on 2 observations:
  - If a search in a direction results in a higher value of the objective function the the opposite direction should provide a lower value of the objective function
  - Successive successful (failure) searches in a certain direction should bias subsequent searches toward (or against) this direction

Random Search (cont.) (7.4)

✶Flowchart:



Downhill Simplex Search (7.5)

✶It is a derivative free method for multidimensional function optimization

✶It has an interesting geometrical interpretation

✶Principle:

– It starts with an initial simplex and replaces the point having the highest function value in a simplex with another point

Downhill Simplex Search (cont.) (7.5)

★It is based on 4 operations:

– Reflection

– Reflection & expansion

– Contraction

– Shrinkage

Downhill Simplex Search (cont.) (7.5)

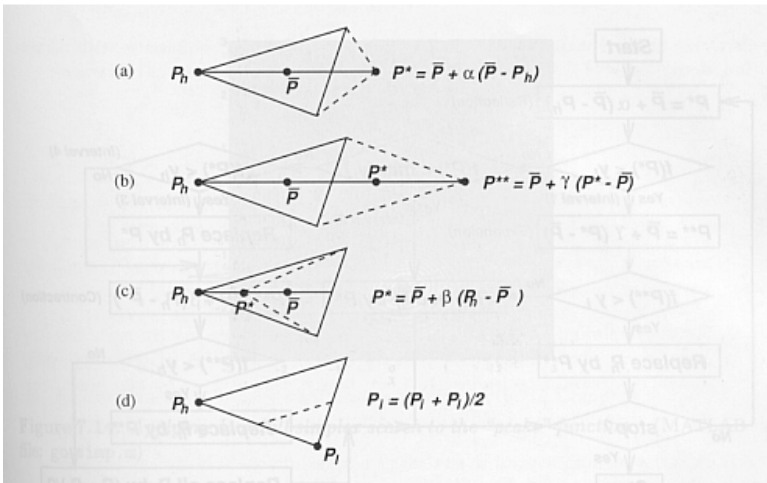


Figure 7.11. *Outcomes for a cycle in the downhill simplex search after (a) reflection away from* $\mathbf{P}_h$*; (b) reflection and expansion away from* $\mathbf{P}_h$*; (c) contraction along one dimension connecting* $\mathbf{P}_h$ *and* $\bar{\mathbf{P}}$*; (4) shrinkage toward* $\mathbf{P}_l$ *alone all dimensions.*

Downhill Simplex Search (cont.) (7.5)

✶Cycle of the DSS

– Start with a point P0 and set up the simplex
– Pi = P0 + λi ei (i = 1, …, n). Let's define the following quantities:

- l = argmin{yi} (l for "low")
- h = argmax{yi} (h for "high")
- yi is the function value at the simplex point Pi

Downhill Simplex Search (cont.)

✶Cycle of the DSS (cont.)

- Interval 1: $\{y; y \leq y_l\}$
- Interval 2: $\{y; y_l < y \leq \max_{i \neq h} \{y_i\}\}$
- Interval 3: $\{y; \max_{i \neq h} < y \leq y_h\}$
- Interval 4: $\{y; y_h < y\}$

Downhill Simplex Search (cont.) (7.5)

★Cycle of the DSS (cont.)

– 4 Steps are needed for each cycle of the DSS

A. Reflection: Define the reflection point $\overset{*}{\mathbf{P}}$ and its value $\overset{*}{\mathbf{y}}$ as:

(Choose an opposite direction of P_h)

$$\overset{*}{\mathbf{P}} = \bar{\mathbf{P}} + \alpha\,(\bar{\mathbf{P}} - \mathbf{P_h}) \qquad (\bar{\mathbf{P}} \text{ centroid of } (\mathbf{n+1}) \text{ points})$$

$$\overset{*}{\mathbf{y}} = \mathbf{f}\left(\overset{*}{\mathbf{P}}\right) \qquad (\alpha > 0)$$

Downhill Simplex Search (cont.) (7.5)

★Cycle of the DSS (cont.)

A. Reflection (cont.)

Test: 1. If $\overset{*}{\mathbf{y}} \in$ Interval 1 $\Rightarrow$ go to Expansion (continue on this direction)

2. If $\overset{*}{\mathbf{y}} \in$ Interval 2 $\Rightarrow$ replace P_h with $\overset{*}{\mathbf{P}}$ and terminate the cycle

3. If $\overset{*}{\mathbf{y}} \in$ Interval 3 $\Rightarrow$ replace P_h with $\overset{*}{\mathbf{P}}$ and go to contraction

4. If $\overset{*}{\mathbf{y}} \in$ Interval 4 $\Rightarrow$ go to contraction (change direction)

Downhill Simplex Search (cont.) (7.5)

✶Cycle of the DSS (cont.)

B. Expansion: Define the expansion point

$$\overset{**}{\mathbf{P}} = \bar{\mathbf{P}} + \gamma(\overset{*}{\mathbf{P}} - \bar{\mathbf{P}})$$

$$\overset{**}{\mathbf{y}} = \mathbf{f}\left(\overset{**}{\mathbf{P}}\right) \quad (\gamma > 1)$$

Test: if $y^{**} \in$ interval 1, replace P_h with $\overset{**}{\mathbf{P}}$ and terminate cycle. Otherwise, replace P_h with the original reflection point $\overset{*}{\mathbf{P}}$ & terminate the cycle

Downhill Simplex Search (cont.) (7.5)

✶ Cycle of the DSS (cont.)

C. Contraction: Define $\overset{**}{\mathbf{P}} = \bar{\mathbf{P}} + \beta(\mathbf{P_h} - \bar{\mathbf{P}})$

$$\overset{**}{\mathbf{y}} = \mathbf{f}\left(\overset{**}{\mathbf{P}}\right); 0 < \beta < 1$$

Test: if $\overset{**}{\mathbf{y}} \in$ Interval 1, 2 or 3 replace P_h with $\overset{**}{\mathbf{P}}$ and terminate the cycle. Otherwise go to shrinkage

D. Shrinkage: Replace P_i with $(P_i + P_l) / 2$ and terminate cycle