Optimization Problems and its solutions by Evolutionary Computing

Dr. Haider Banka
Associate Professor, Department of Computer Science & Engineering
Indian School of Mines, Dhanbad 826004
email: banka.h.cse@ismdhanbad.ac.in

Outline

1. Introduction to Optimization problems
2. Classical Optimization Techniques
3. Genetics Algorithms (GAs)
4. Multi-objective GA
5. Particle Swarm Optimization (PSO)
6. Its Applications

What is Optimization?

- **Optimization** is the mathematical discipline which is concerned with finding the maxima and minima of functions, possibly subject to constraints.

- **Optimization** is an iterative process by which a desired solution (max/min) of the problem can be found while satisfying all its constraint or bounded conditions.

Unconstrained optimization

$$\min\ f(x,y) = x^2 + 2y^2$$

Optimization with constraints

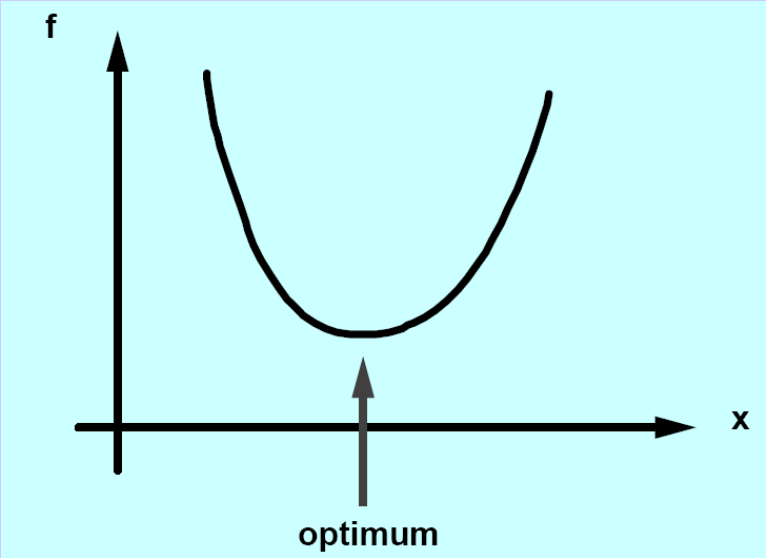
$$\min\ f(x,y) = x^2 + 2y^2$$
$$x > 0$$

or

$$\min\ f(x,y) = x^2 + 2y^2$$
$$-2 < x < 5, y \geq 1$$

or

$$\min\ f(x,y) = x^2 + 2y^2$$
$$x + y = 2$$

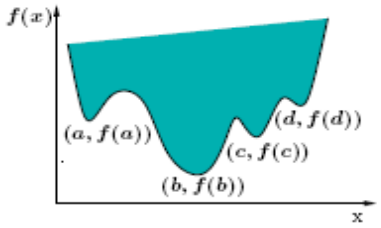


Optimum solution is found while satisfying its constraint (derivative must be zero at optimum).

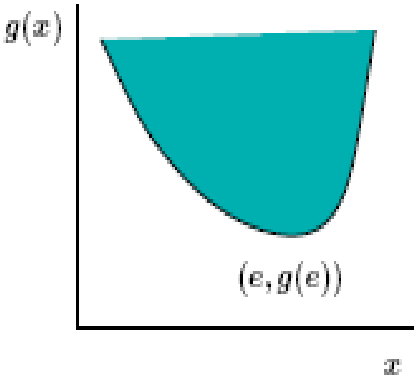
- Optimization problem could be linear or non-linear.
- Non –linear optimization is accomplished by numerical '***Search Methods*** *(basically* iterative in nature*)*
- The search procedure is termed as ***algorithm***.

Fundamentals of Non-Linear Optimization

Global versus Local Optimization



- $f(x)$ is a multimodal function
- There are four local optima: a, b, c, d
- There is only one global optimum: b



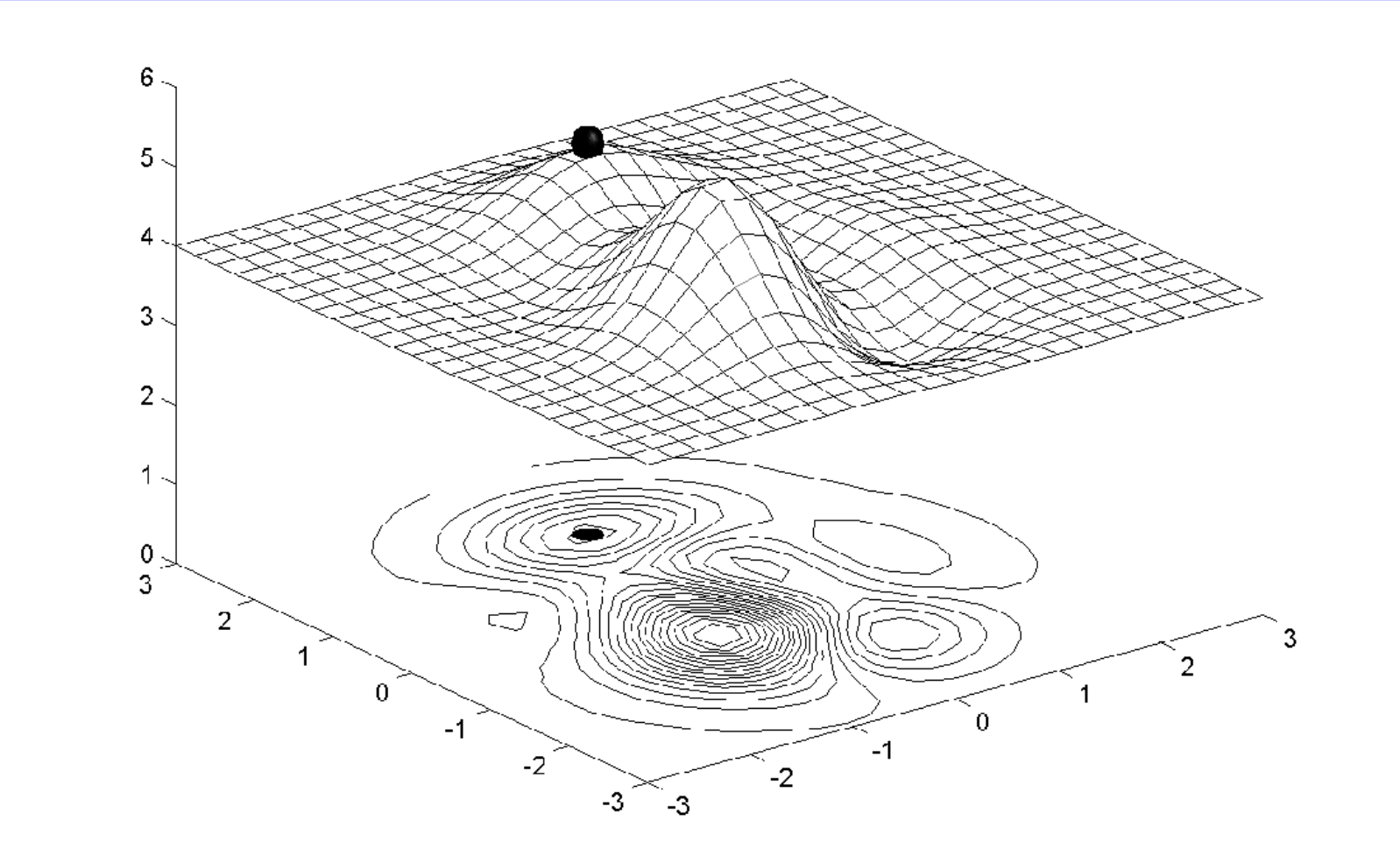
- $g(x)$ is a unimodal function
- Unique local and global minimum e

Local = Global if function is convex

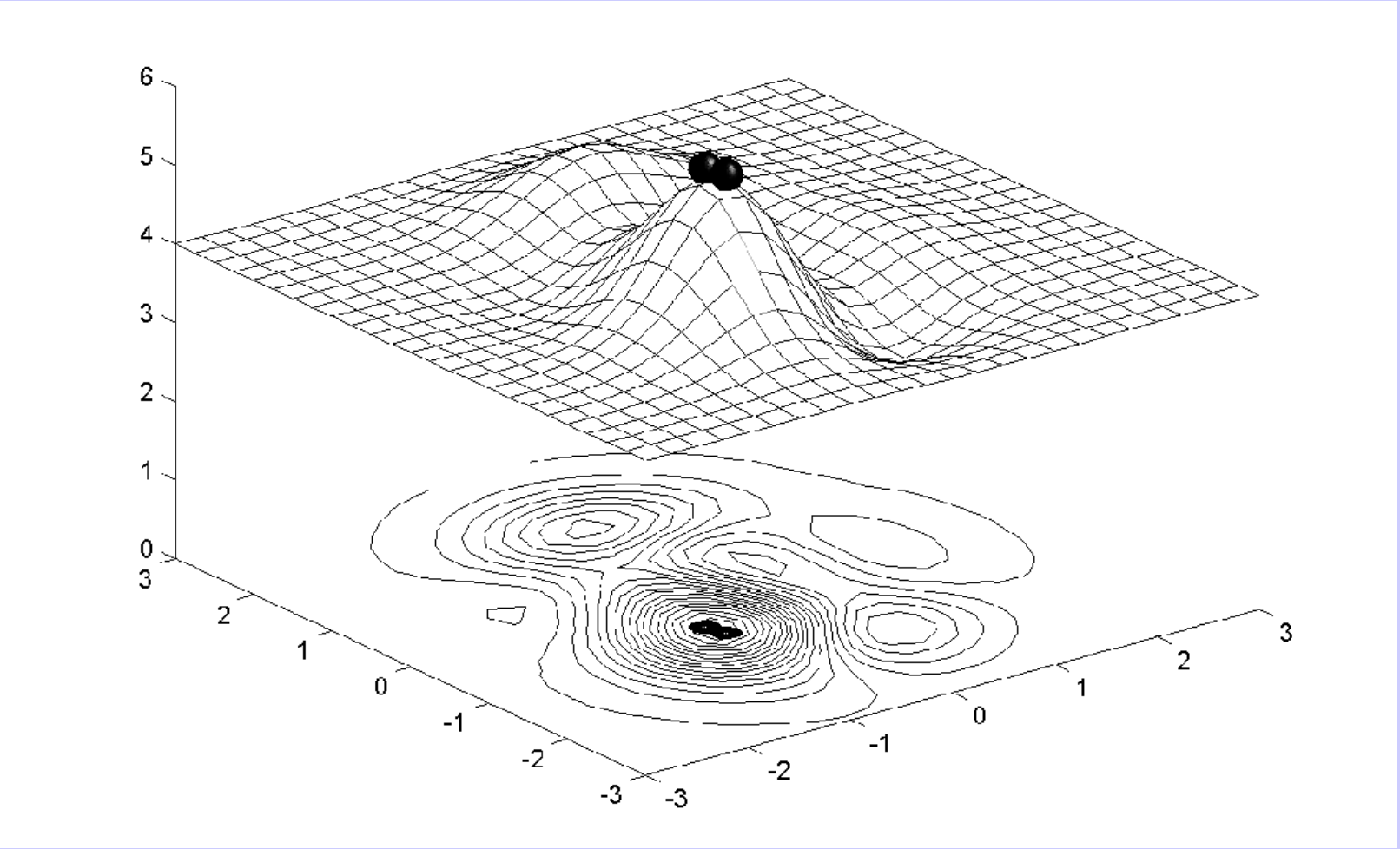
Global versus local optimization.

Local point is equal to global point if the function is convex.

Chromosome locations on the surface of the "peak" function: local maximum



Chromosome locations on the surface of the "peak" function: global maximum



- **Mathematical optimization problem:**

$$\text{minimize} \quad f_0(x)$$
$$\text{subject to } g_i(x) \le b_i, \quad i = 1,....,m$$

- $f_0 : \boldsymbol{R^n} \rightarrow \boldsymbol{R:}$ objective function
- $x=(x_1,.....,x_n)$: design variables (unknowns of the problem, they must be linearly independent)
- $g_i : \boldsymbol{R^n} \rightarrow \boldsymbol{R:}$ $(i=1,...,m)$: inequality constraints

- The problem is a constrained optimization problem

Optimization Methods

- **Global Optimization** – Stochastic techniques

- **Genetic Algorithm (GA)** – Survival of the fittest principle based upon evolutionary theory

- **Particle Swarm Optimization (PSO)** – principle based upon flock of birds, school of fish

- **Simulated Annealing** (SA) method – minimum energy principle of cooling metal crystalline structure

Outline of GA

- Open question of GAs
- Convergence and diversity
- Genetic algorithm search capabilities
- Parameters of GAs
- Genetic operators
- Schema theorem
- Variations of GAs

Convergence and diversity

- Convergence/premature convergence
  - The GA continues while fitter chromosome existed in population.
  - The population will converged by GA so that all individuals as same fit as each other
  - The converged population will not be evolved no matter if the best solution found.
    - If not? =>
- Diversity
  - Diversity maintained in genes
  - Individuals with the same fitness might keep different gene strings
  - Chance to survive if environment changes
  - Chance to break premature convergence by mutation operators.

Phenotype and Genotype

- Phenotype
  - To describe what an individual looks like, i.e., the display after fitness function calculation.
- Genotype
  - To describe genes on chromosome.
- Individuals with the same phenotype may has totally different genotype
- Genotype maintain diversity for chance of changes
- Population with the same genotype individuals still can been evolved by mutation operator
  - Mutation operator is too strong: much diversity in genotype and no population could be converged
  - Mutation is too week: less chance to change and premature convergence will become easily

Exploitation and Exploration

- Exploitation
  - Takes the current search information from the experience of the last search to guide the search toward the direction that might be close to the best solutions.
  - Is focusing more in area promise solution.
  - From Selection operator and Crossover operator.
- Exploration
  - Widens the search to reach all possible solutions around the search space.
  - Implies search achieve entire search space
  - From Mutation operator and crossover operator.
- **Important task**: Balancing exploitation and exploration.
  - Too high exploitation leads to premature convergence
  - Too high exploration leads to non-convergence and to no fitter solution.

Representation - Binary Encoding

- Most widely used representation
- Calculating value by decoding binary string
- The way to encode/decode chromosome affects precision and accuracy seriously.

Chromosome									
1	0	1	0	0	0	1	1	0	……

Representation - Many-Character and Real Valued Encoding

- most natural to use an alphabet of many characters or real numbers to form chromosomes

Chromosome									
32.15	25	40	67	25.0134	67	50	89	128	……

Representation - Permutation Encoding

- Permutation encoding is used in scheduling and ordering problems. (E.g., Tabu search)
- Every chromosome is a string of numbers, which represents numbers in a sequence.
- The kind of encoding usually needs to make some corrections after the crossover and mutation operating procedures because any transformation might create an illegal situation.

Chromosome									
5	6	1	1	4	3	2	8	7	……

Fitness Function

- The way to calculate degree of fitness of individuals to environment (problem)
- Better fitness function could increase selection pressure.
- Also needed to be considered with selection operator.
  - e.g. 1, $f(x) = 25X^2+30X+10$
  - e.g. 2, $f(x) = (25X^2+30X+10) - \min(f(x))$

Genetic operators

- Reproduction (Selection)
  - To select fitter chromosomes to be parents
- Crossover
  - To exchanges information encoded in chromosomes between parents
- Mutation
  - To randomly change information encoded in chromosomes

Genetic Operators - Reproduction (Selection)

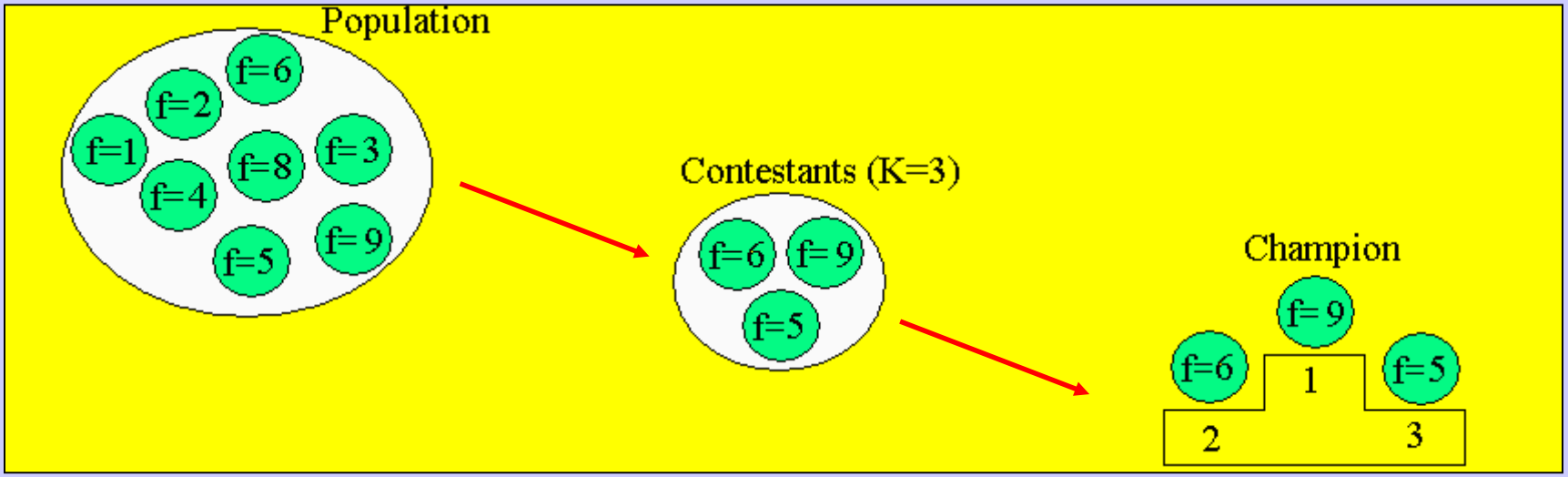
- **Roulette wheel selection**
  - The most common method
  - Each individual is assigned a slice of a circular “roulette wheel”, the area of the slice being proportional to the individual’s fitness.
  - The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel’s marker is selected to be in the pool of parents for the next step.
  - The selective capability depends on the variance of the fitness in the population
  - population converges, the variance of the fitness is low and the difference between the fitness of the individuals is small. Thus the probabilities of the different individuals which are selected are similar so that it is difficult to select a better individual among the other individuals with similar fitness values.

Genetic Operators - Reproduction (Selection)

- **Rank selection**
  - The individuals in the population are ranked according to fitness, and the expected value of each individual depends on its rank rather than on its absolute fitness.
  - Roulette-wheel selection operator is introduced to select the fitter individual
  - Better selecting pressure while population converges
  - Worse distinguishability for individuals with higher difference of fitness. Thus, convergence might be slow and search might be directed to wrong direction.

Genetic Operators - Reproduction (Selection)

- **Tournament selection**
  - Individuals are chosen at random from the population.
  - A random number *r* is then chosen between 0 and 1. If $r<k$ (where *k* is a parameter, e.g., 0.75),
  - the fitter of the two individual is selected as parent; otherwise the less fit individuals is selected.
  - Much faster and more performed than Roulette-Wheel selection



Genetic Operators - Reproduction (Selection)

- **Elitism**
  - Forces the GA to retain some number of the best individuals at each generation.
  - Such individuals can be lost if they are not selected for reproduction or if they are destroyed by crossover or mutation.
  - Many researchers have found that elitism significantly improves the GA's performance.
- **Steady-State selection**
  - only a few individuals are replaced in each generation: usually a small number of the least fit individuals are replaced by offspring resulting from crossover and mutation of the fittest individuals.
  - A refinement of this method is steady state with no duplicates in which we do not allow children that are duplicates of chromosomes which already exist in the population.
  - This method is often used in evolving rule-based systems in which incremental learning

Genetic Operator - Crossover

- **Single-point crossover**
- End point effect and hitchhiker problem

crossover point

PARENT1	0	1	1	0		0	1	1	1	0
PARENT2	1	1	0	0		1	0	1	0	0

CHILD1	0	1	1	0		1	0	1	0	0
CHILD2	1	1	0	0		0	1	1	1	0

Genetic Operator - Crossover

- **Multiple-point crossover**

			CP1				CP2			CP3			
PARENT1	0	1	1	1	0	0	1	0	1	0	0	1	0
PARENT2	1	0	0	1	1	0	0	0	1	1	1	0	1

CHILD1	0	1	0	1	1	0	1	0	1	1	1	0	1
CHILD2	1	0	1	1	0	0	0	0	1	0	0	1	0

Genetic Operator - Crossover

- **Parameterised uniform crossover**
  - exchanges bits of two parents at each position of the chromosomes with a probability *p* (typically 0.5-0.8).
  - No position bias, no endpoint effect, affecting schema theorem.

Genetic Operator - Mutation

- Repeat for each bit:
  1. Generate a random number r between 0 and 1.
  2. If $r < m_r$ (i.e., mutation rate) then flip the bit.
- For real number representation
  - Randomly creating a number for the mutated bit
- For tree representation
  - Tree mutation: randomly creating a branch replacing the current branch
  - Point mutation: change this bit value by random number/ random factor

Ending iteration

- When does this iteration stop?
  - Answer is “We did not know ahead of design of simulation”.
- Three criteria for stopping iteration
  - Population converges: Phenotypes of all individuals are same.
  - Evolutionary stability: The phenotype of all individuals does not change through several rounds.
  - Fixed generation number: But more means better.

Optimum achievement

- Have we got fitness optimum while iteration stop?
  - The answer is “We do not know even the simulation has been completed while this algorithm is used for unknown-result problems.”
- How to guarantee the fitness achievement?
  - Different initialization design
  - Different parameter settings
  - Conclusion of results from more experiments.
  - More generation for each experiments.

Parameter setting

- Population
  - Size of population vs. scale of search space
- Representation
  - Precision and accuracy of potential solution
- Fitness function
  - Selecting pressure concerned
- Crossover rate
  - C_r : 0.6 ~0.8
- Mutation rate
  - m_r : 0.0001 ~ 0.001

Open questions of GAs

- Have we found the solution?
  - Actually, we are not sure…
- Parameter setting
  - Many parameters should be decided ahead of time
- Selecting pressure
  - Difference of fitness among individuals
- Balance of exploitation and exploration
  - Perform search quality and search performance
- Ending of iteration/generation
  - When should we stop iteration?
  - Actually, we do not know…

Simple genetic Algorithm

Basic Operations:
Selection, Cross-over, Mutation

- Basic Term:

Chromosome, Population, Fitness/Objective Function

Multi-objective GA

Dominance Criteria

Non-dominated sorting

Crowding distance

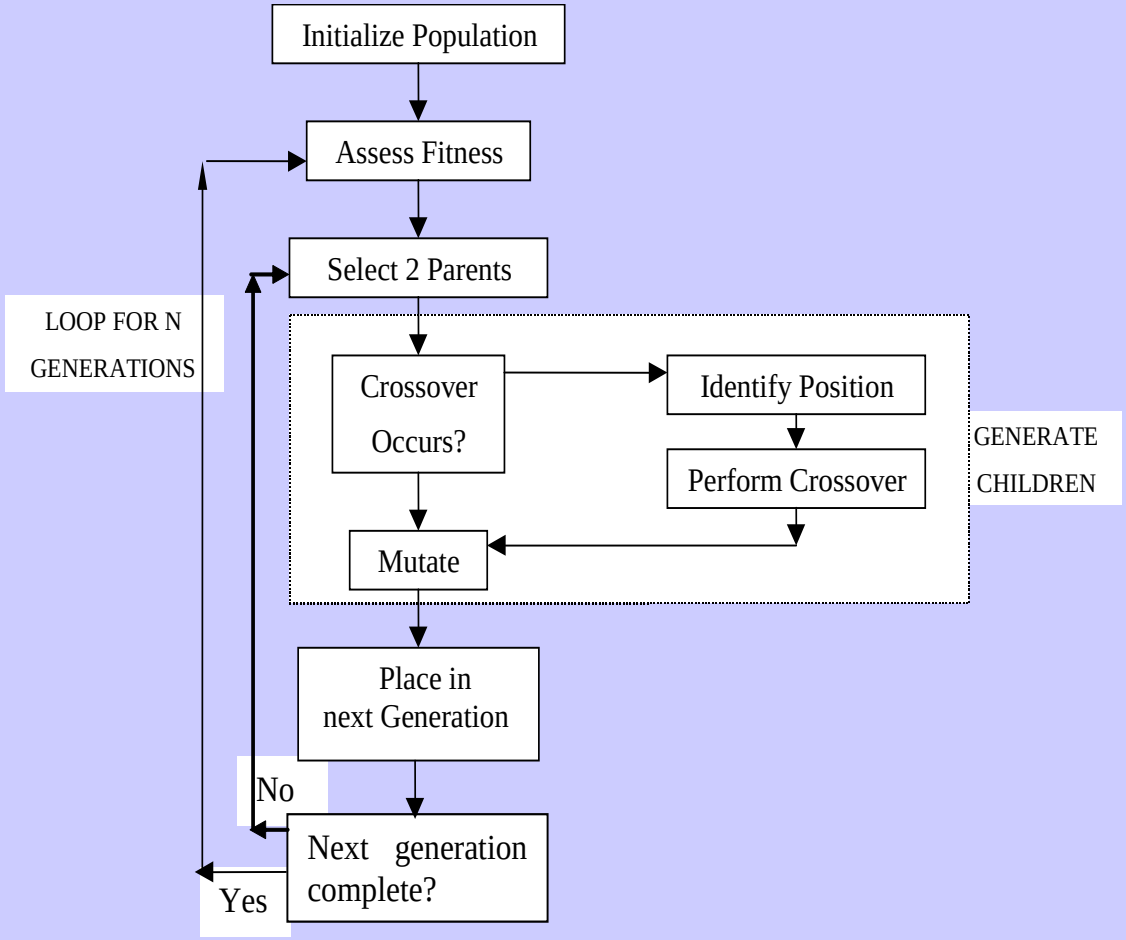
Crowding tournament selection operator

The Non-dominated sorting Genetic Algorithm (NSGA II)

Genetic Algorithm

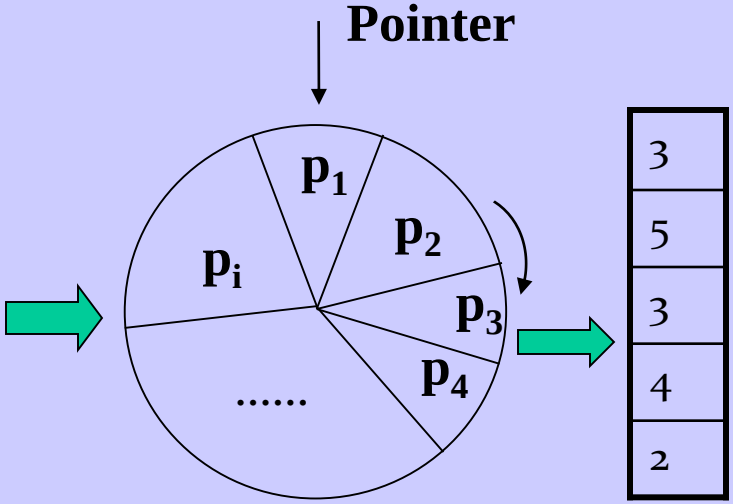
- Genetic algorithms are global stochastic optimization techniques based on natural genetics.
- Robust and non-problem specific.
- GAs code the parameter set of the optimization problem as finite-length string.
- GAs start the searching from a population of random points, improve the quality of the population over time by genetic operations: selection, crossover, mutation;
- The best fitted solution will be evolved toward objective function.

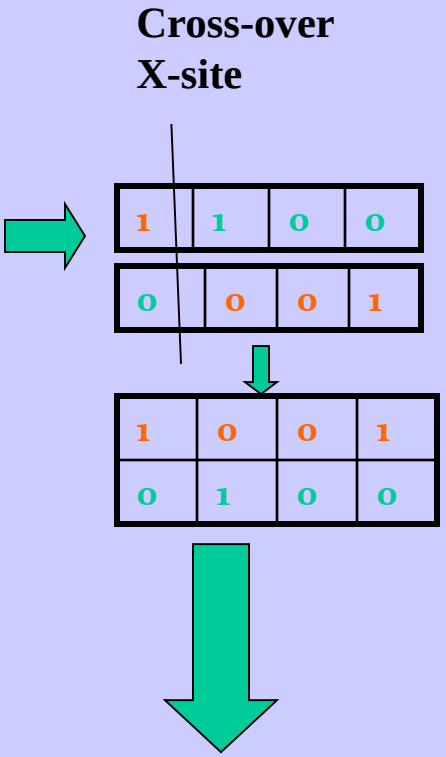
The Simple Genetic Algorithm

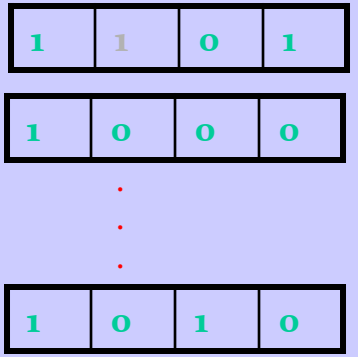


Parent population

Populations of Chromosome (s)				x	$f(x) = x^* x$
1	0	1	0	10	100
0	1	1	0	6	36
1	1	0	0	12	144
1	0	1	0	10	100
0	0	0	1	3	1
0	0	1	1	3	9







mutation

Child population

1	0	0	1	9	81
0	1	0	0	4	16
1	1	0	1	13	169
1	0	1	0	10	100
1	0	1	0	10	100
0	0	1	1	3	9

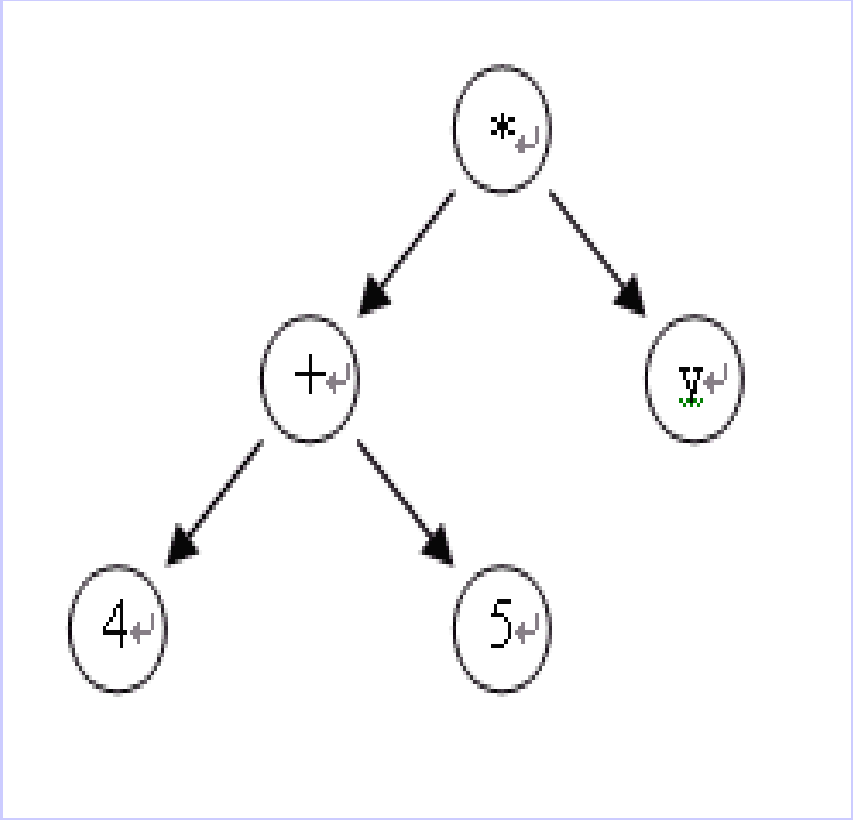
Other Approaches

- Evolutionary Programming (EP)
  - Fogel in the early 1960's, it has no "genomic" representation. Each individual in the population is an algorithm chosen at random over an appropriate sample space. **Mutation is the only genetic operator used; EP does not use crossover**
- Evolution Strategies (ES)
  - Schwefel, also in the 1960s, as an optimisation tool. ES uses a real-valued chromosome with a **population size of one and mutation as the only genetic operator**. In each generation the parent is mutated to produce a descendant; if the descendant it fitter it becomes the parent for the next generation, otherwise the original parent is retained.

- Genetic Programming (GP)
  - Koza in the late 1980's, the aim of GP is the automatic programming of computers; allowing programs to evolve to solve a given problem. The population consists of programs expressed as parse trees; operators used include crossover, mutation and architecture-altering operations patterned after gene duplication and gene deletion in nature
- Many others, often tailored to problem at hand

Representation - Tree Encoding

- i.e., Genetic programming (GP)
- Allowing the search space to be open-ended so that tree can grow large in uncontrolled ways preventing the formation of more structured, hierarchical candidate solutions.



Steps in the GA development

1. Specify the problem, define constraints and optimum criteria;
2. Represent the problem domain as a chromosome;
3. Define a fitness function to evaluate the chromosome performance;
4. Construct the genetic operators;
5. Run the GA and tune its parameters.

Schema Theorem

Who will survive and who will die - the ultimate dilemma

Schema Theorem

- Some definition:
  - Building Block : a useful pattern (fitness-wise)
  - Schema : 10****000******
  - **Building blocks** are represented by **schemata**
- The GA allows the emergence of building blocks and combine them to high quality solutions
- The GA do not evolve chromosomes themselves but do many more schema parallel.

Schema

Don’t care symbol: *

*	*	*	1	0	*	1	*	*	*

order of a schema: $o(S)$ = # fixed positions

defining length $\delta(S)$ = distance between first and last fixed position

a schema S matches $2^{l - o(S)}$ strings

a string of length l is matched by 2^l schemata

Schema Theorem

- **m(h,t)** : #instances of schema h in the population at generation t
- **f(h,t)** : observed average fitness of schema h at generation t

  $f(h,t) = \sum_{x \in h \; x \in P(t)} f(x)/m(h,t)$
- E(m(h,t)) : expected #instances of schema h
- selection :

  $\mathbf{E(m(h,t+1))} = \sum_{x \in h \; x \in P(t)} f(x)/\langle f \rangle = \mathbf{m(h,t)}\, f(h,t)/\langle f \rangle$
- crossover: probability that a schema h of defining length d(h) is not destroyed by crossover: $\mathbf{(1 - p_c\, d(h)/(l-1))}$
- mutation: probability that schema h of order o(h) is not destroyed by mutation : $\mathbf{(1 - p_m)^{o(h)}}$

Schema Theorem

$$E(m(h,t+1)) \geq m(h,t)\, f(h,t)/\langle f \rangle\, (1 - p_c\, d(h)/(l-1))\, (1 - p_m)^{o(h)}$$

- simple GA increases the number of schemata with
  - low order
  - short defining length
  - above average fitness
- **implicit parallelism**
  - simultaneous evaluation of a large number of schemata in a population of n strings
  - one string implicitly samples 2^l different schemata
- **building block hypothesis**
  - GA works by recombining instances of good schemata to form instances of equally good or better higher-order schemata by means of crossover

THE SCHEMA THEOREM

- SELECTION EFFECTS

$$m(H,t+1) = m(H,t)\cdot\frac{f(H)}{\bar{f}} \qquad m(H,t+1) = m(H,t)\cdot\frac{\bar{f}+c\cdot\bar{f}}{\bar{f}} = (1+c)\cdot m(H,t) \qquad m(H,t) = m(H,0)\cdot(1+c)^{t}$$

- CROSSOVER EFFECTS

$$P_s(H) \geq 1 - P_c\cdot\frac{\delta(H)}{l-1} \qquad \Rightarrow \qquad m(H,t+1) \geq m(H,t)\cdot\frac{f(H)}{\bar{f}}\cdot\left[1-\left(P_c\cdot\frac{\delta(H)}{l-1}\right)\right]$$

- MUTATION EFFECTS

$$P_s(H) = (1-P_m)^{o(H)} \qquad P_m << 1 \Rightarrow (1-P_m)^{o(H)} \approx 1 - o(H)\cdot P_m$$

- COMBINED EFFECTS: THE SCHEMA THEOREM

$$m(H,t+1) \geq m(H,t)\cdot\frac{f(H)}{\bar{f}}\cdot\left\{1-\left(P_c\cdot\frac{\delta(H)}{l-1}\right)-o(H)\cdot P_m\right\}$$

"The short, low definition order schemata with average fitness above the mean population fitness (*Building Blocks* or BBs) get a number of instances that grow exponentially. The GA makes this work in an implicit parallel way for all the population and schemata."

Disadvantage of GAs

- No guarantee for optimal solution within finite time –
- May need extensive parameter tuning
  - Search capability is sensitive to parameter settings and operator adopted
  - Representation, fitness function, even position of bits respected influence judgement of achievement of fitness optimum (Global/Local)
- Often computationally expensive, i.e. slow

Benefits of GAs

- Handles huge search spaces
- Balances exploration and exploitation
- Easy to try - not knowledge intensive
- Easy to combine with other methods
- Provides many alternative solutions
- Can continually evolve solutions to fit with a continually changing problem

Multi-objective Genetic Algorithms (MOGAs)

- Deal with multiple, often competing objectives.
- Present a set of Pareto optimal solutions

- Example: purchasing of a car (finance available, comfort, distance to be driven by each day, no. of passengers riding, fuel consumption and cost, depreciation value, road conditions etc.)

Dominance Criteria

Definition: If there are *M* objective functions, a solution $x^{(1)}$ is said to dominate solution $x^{(2)}$, lf both conditions 1 and 2 are true:

1) The solution $x^{(1)}$ is *no worse* than $x^{(2)}$ in all the *M* objectives.

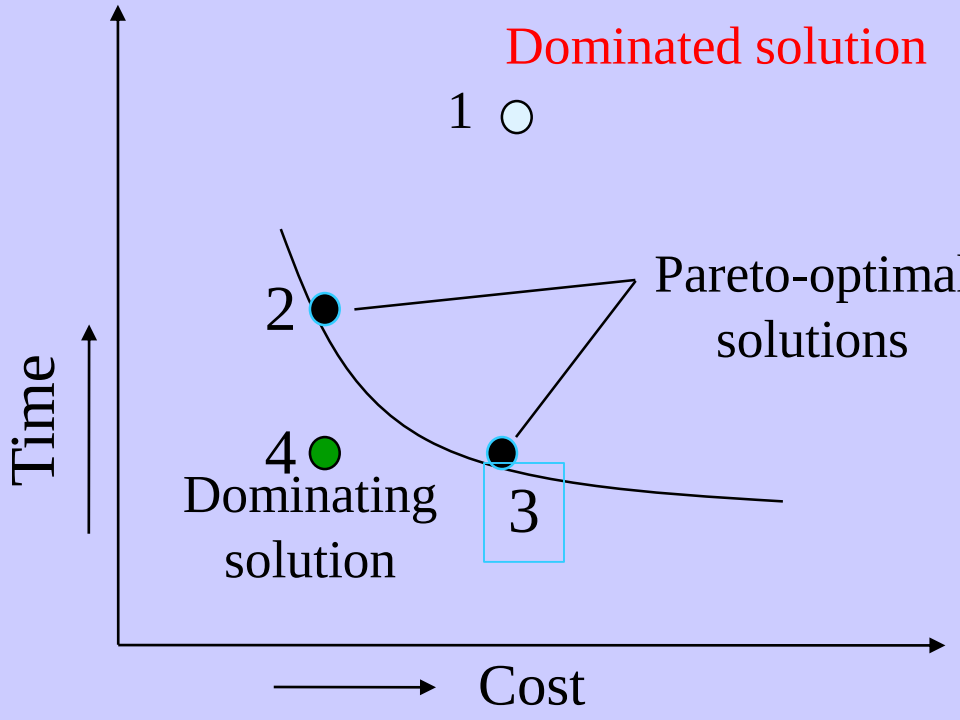
2) The solution $x^{(1)}$ is *Strictly better* than $x^{(2)}$ in *at least one* of the *M* objectives.

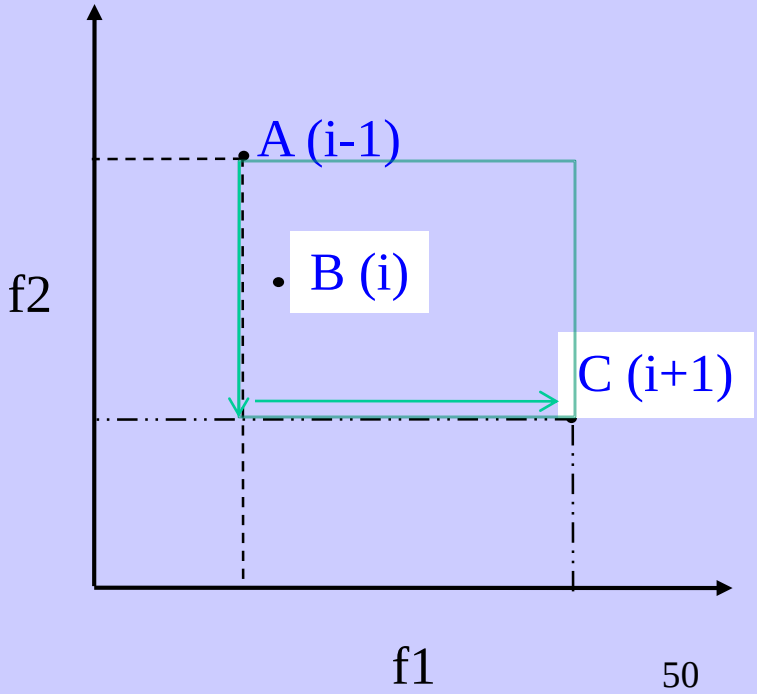
Otherwise, the two solutions are non-dominating to each other.

When a solution $x^{(1)}$ dominates a solution $x^{(2)}$, then rank of $x^{(1)} < x^{(2)}$

Pareto optimal solution

- A solution x is pareto-optimal if there doesn't exist any other solutions that dominate x.
- equally good; non-dominated;





Algorithm for finding non-dominated set in a population P of size $|P|$

Step 1) Set counter i=1 and $P' = \varnothing$

Step 2) for each solution $j \in P(j \neq i)$, check if j dominates i. If yes, go to Step 4)

Step 3) If more solution left in P, increment j by one and go to Step 2). Else Set $P' = P' \cup \{i\}$.

Step 4) Increment I by one. If $i \leq |P|$ then go to Step 2). Else declare P' as the non-dominated set

The process is repeated with $P = P - P'$ until $P = \varnothing$

Crowding distance

1) Let the no. of solutions in F be $l = |F|$ and assign $d_i = 0$ for $i = 1, 2, ..., l$

2) For each objective function $f_k, k = 1, 2, ..., M$, sort set in its worse order (ascending/descending)

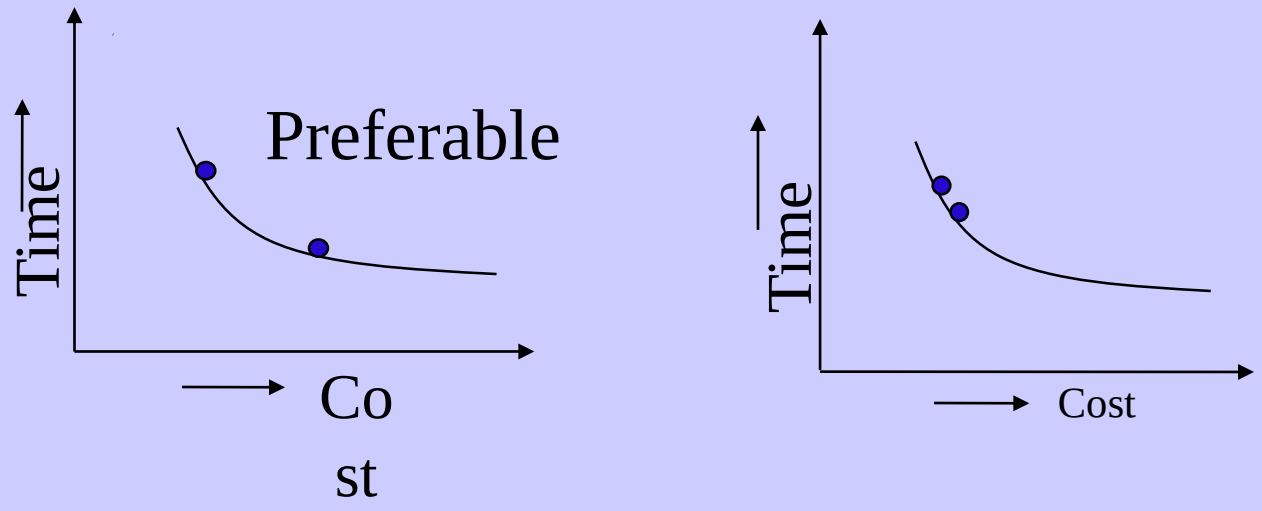
3) Set $d_1 = d_l = \infty.$

4) For $j = 2$ to $(l - 1)$ increment d_j by $(f_{k_{j+1}} - f_{k_{j-1}})$

Goals of multi-objective solutions

✓Find a set of solutions as close as possible to the Pareto optimal front. (Non dominated sorting)

✓Find a set of sparsely spaced solutions, as far as possible. (Crowding distance operator)



Crowding Selection Operator

A solution i *wins tournament* with another solution j *if any one of* the following is true.

1) Solution i *has better rank, i.e.,* $r_i < r_j$

2) Both the solutions are in the same front, i.e., $r_i = r_j$ *, but* $d_i > d_j$
solution i *is less densely located in the search space*

NSGA II Algorithm

1. Initialize the population randomly.
2. Calculate the multi-objective fitness functions.
3. Rank the population using dominance criteria.
4. Calculate the crowding distance.
5. Do selection using crowding selection operator.
6. Do crossover and mutation to generate children population.
7. Combine parent and children population.
8. Replace the parent population by the best members of the combined population.

Evolutionary Rough Feature Selection

Preprocessing (Normalization, Thresholding, and Discretization)

Making of distinction table

Fitness functions for Evaluation

The complete algorithm

Experimental results and comparison (Colon, Leukemia, and Lymphoma datasets)

(Shall be discussed in next session)

Swarm Intelligence

- Swarm Intelligence (SI) is the property of a system whereby
  the collective behaviors of (unsophisticated) agents
  interacting locally with their environment
  cause coherent functional global patterns to emerge.
- SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model.
- Leverage the power of complex adaptive systems to solve difficult non-linear stochastic problems

Swarm Intelligence

- Characteristics of a swarm:
  - Distributed, no central control or data source;
  - Limited communication
  - No (explicit) model of the environment;
  - Perception of environment (sensing)
  - Ability to react to environment changes.

Origins and Inspiration of PSO

- Population based stochastic optimization technique inspired by *social behaviour of bird flocking or fish schooling*.
- Developed by **Jim Kennedy,** Bureau of Labor Statistics, U.S. Department of Labor and **Russ Eberhart,** Purdue University
- A concept for optimizing nonlinear functions using particle swarm methodology

- Inspired by simulation social behavior
- Related to bird flocking, fish schooling and swarming theory

  - steer toward the center

  - match neighbors' velocity

  - avoid collisions
- Suppose
  - a group of birds are randomly searching food in an area.
  - There is only one piece of food in the area being searched.
  - All the birds do not know where the food is. But they know how far the food is in each iteration.
  - So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

What is PSO?

- In PSO, each single solution is a "bird" in the search space.
- Call it "particle".
- All of particles have fitness values
  - which are evaluated by the fitness function to be optimized, and
- have velocities
  - which direct the flying of the particles.
- The particles fly through the problem space by following the current optimum particles.

PSO Algorithm

- Initialize with randomly generated particles.
- Update through generations in search for optima
- Each particle has a *velocity* and *position*
- Update for each particle uses two "best" values.
  - Pbest: best solution (fitness) it has achieved so far. (The fitness value is also stored.)
  - Gbest: best value, obtained so far by any particle in the population.

- PSO algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology.
- A PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.

- Population-based search procedure in which individuals called **particles** change their **position** (state) with time. $\vec{r}$

  → individual has position $\vec{x}_i$

  & individual changes velocity $\vec{v}_i$

- Particles **fly** around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a **neighboring** particle, making use of the best position encountered by itself and its neighbor.

Particle Swarm Optimization (PSO) Process

1. Initialize population in hyperspace
2. Evaluate fitness of individual particles
3. Modify velocities based on previous best and global (or neighborhood) best positions
4. Terminate on some condition
5. Go to step 2

PSO Algorithm

- Update each particle, each generation

a

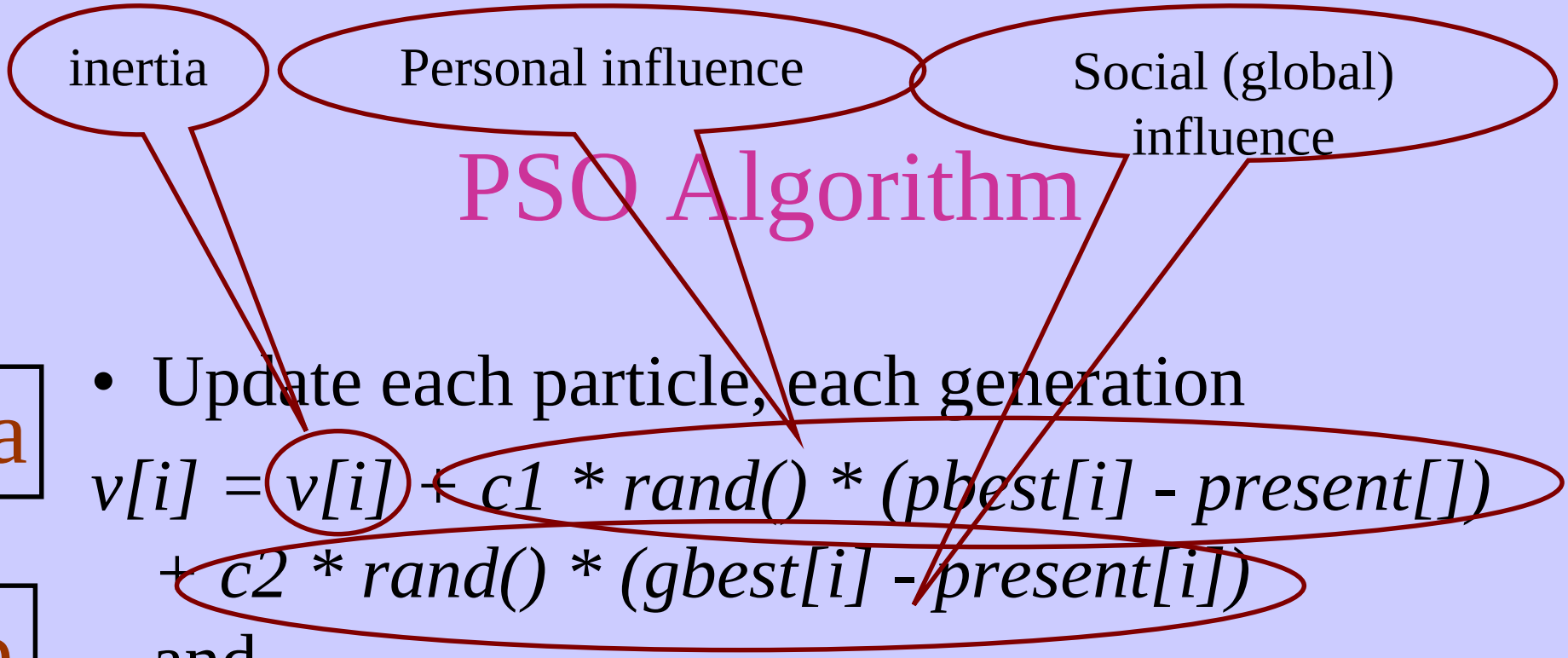
$$v[i] = v[i] + c1 * rand() * (pbest[i] - present[]) + c2 * rand() * (gbest[i] - present[i])$$

b

and

$$present[i] = persent[i] + v[i]$$

where c1 and c2 are learning factors (weights)



PSO Algorithm

a

b

- Update each particle, each generation

$$v[i] = v[i] + c1 * rand() * (pbest[i] - present[]) + c2 * rand() * (gbest[i] - present[i])$$

and

$$present[i] = present[i] + v[i]$$

where c1 and c2 are learning factors (weights)

PSO Algorithm

- **Inertia Weight**

$$v_{id}^{new} = w_i \cdot v_{id}^{old} + c_1 \cdot rand_1 \cdot (p_{id} - x_{id}) + c_2 \cdot rand_2 \cdot (p_{gd} - x_{id})$$

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new}$$

d is the dimension, c_1 and c_2 are positive constants, $rand_1$ and $rand_2$ are random numbers, and w is the inertia weight

Velocity can be limited to V_{max}

PSO and GA Comparison

- Commonalities
  - PSO and GA are both population based stochastic optimization
  - both algorithms start with a group of a randomly generated population,
  - both have fitness values to evaluate the population.
  - Both update the population and search for the optimium with random techniques.
  - Both systems do not guarantee success.

PSO and GA Comparison

- Differences
  - PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity.
  - They also have memory, which is important to the algorithm.
  - Particles do not die
  - the information sharing mechanism in PSO is significantly different
    - Info from best to others, GA population moves together

- PSO has a memory

  →not “what” that best solution was, but “**where**” that best solution was
- **Quality**: population responds to quality factors *pbest* and *gbest*
- **Diverse** response: responses allocated between *pbest* and *gbest*
- **Stability**: population changes state only when *gbest* changes

- There is no selection in PSO

  →all particles survive for the length of the run

  →PSO is the only EA that does not remove candidate population members
- In PSO, topology is constant; a neighbor is a neighbor
- Population size: Jim 10-20, Russ 30-40

PSO Velocity Update Equations

- **Global version vs Neighborhood version**

  → change p_{gd} to p_{ld}.

  where p_{gd} is the *global best position*

  and p_{ld} is the *neighboring best position*

Inertia Weight

- Large inertia weight facilitates global exploration, small on facilitates local exploration
- *w* must be selected carefully and/or decreased over the run
- Inertia weight seems to have attributes of temperature in simulated annealing

V_{max}

- An important parameter in PSO; typically the only one adjusted
- Clamps particles velocities on each dimension
- Determines “fineness” with which regions are searched

  →if too high, can fly past optimal solutions

  →if too low, can get stuck in local minima

PSO – Pros and Cons

- Simple in concept
- Easy to implement
- Computationally efficient
- Application to combinatorial problems?

  → Binary PSO

Thank you