Stochastic Search

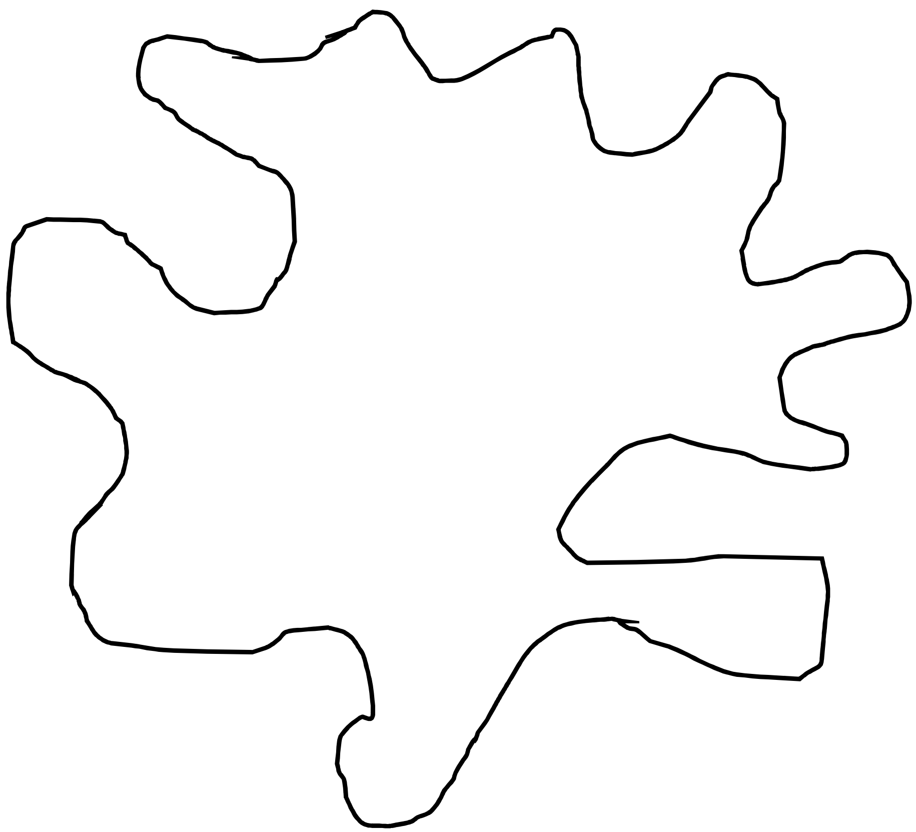
GA and PSO

Prithwijit Guha
Dept. of EEE, IIT Guwahati

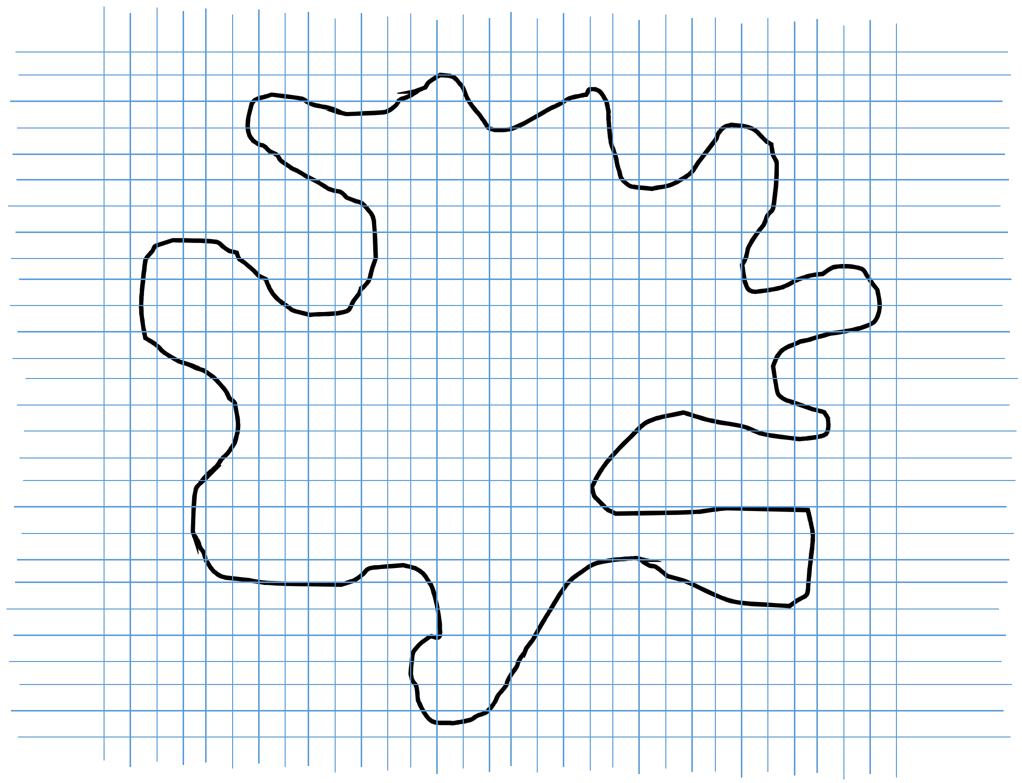
Overview

- An Application of Random Sampling
- Stochastic Search
- The Traveling Salesman Problem
- Genetic Algorithm
- Particle Swarm Optimization

How to Find the Area of this Region?

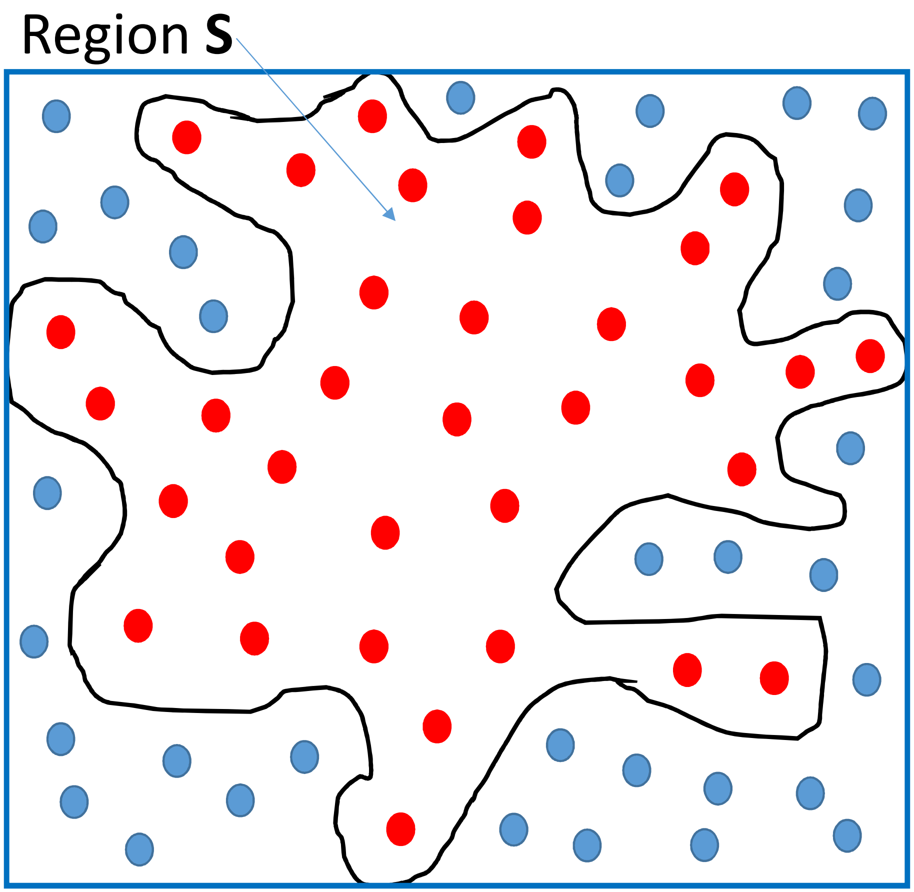


Calculus Approach



Decompose Region into Small Squares

Approximating Area by Random Sampling



Minimum Bounding Rectangle $\boldsymbol{R_S}$ of **S**

$\boldsymbol{N}$ Number of Points are Sampled from a Uniform Distribution $\boldsymbol{U}(\boldsymbol{R_S})$ Supported on $\boldsymbol{R_S}$

- $\boldsymbol{N_s}$ Points within $\boldsymbol{S}$
- $(\boldsymbol{N} - \boldsymbol{N_s})$ Points within $\boldsymbol{R_s} - \boldsymbol{S}$

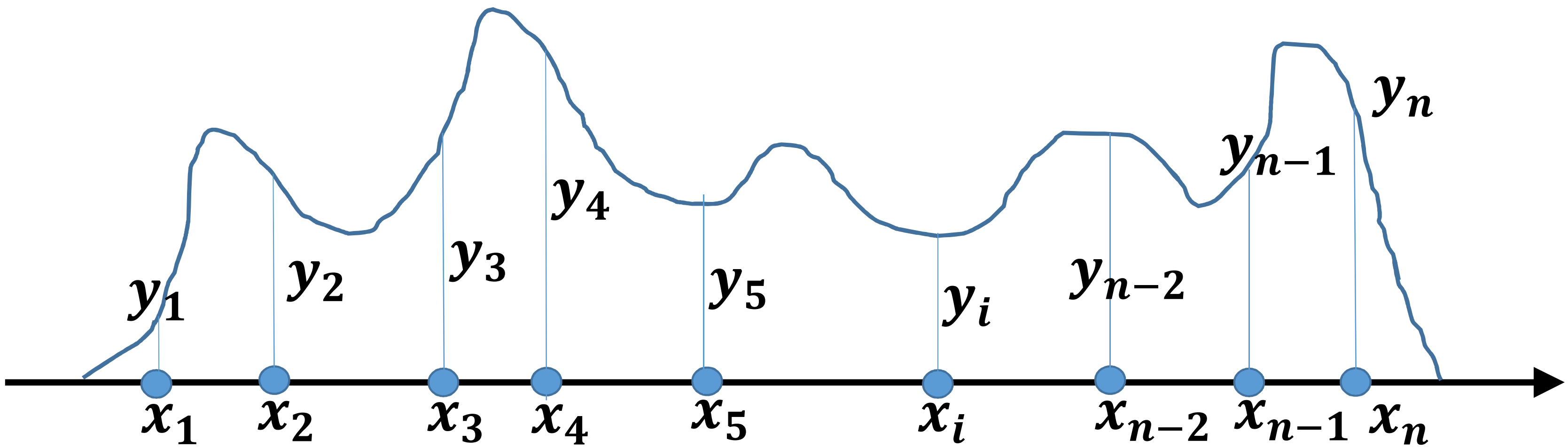
$$\hat{A}(\boldsymbol{S}) = \left(\frac{N_s}{N}\right) A(\boldsymbol{R_S})$$

$$\lim_{\boldsymbol{N}\to\infty} \hat{A}(\boldsymbol{S}) = A(\boldsymbol{S})$$

Certain Issues

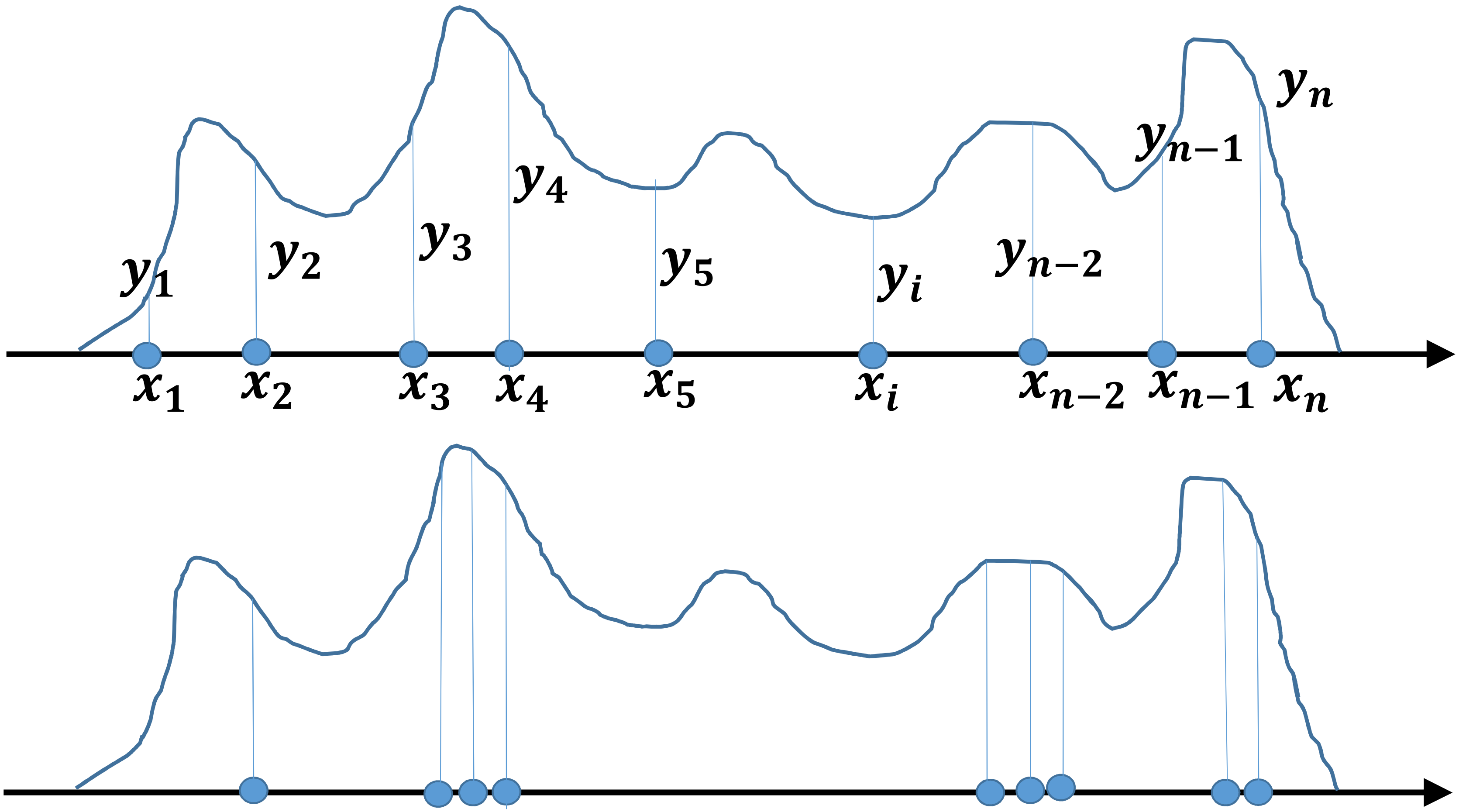
- Objective Function Computation can be Costly
  - Problems Involving Millions of Variables and Constraints
  - Example: Deep Neural Networks and Big Training Data
- Often Limited Function Evaluations are Allowed
- Estimation of $\nabla f(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{R}^n$ involves $(\mathrm{n}+1)$ Function Evaluations
- Objective Function may not be Differentiable
  - Example: $\min\left\{\max_{i=1,\ldots m} f_i(\boldsymbol{x}), g(\boldsymbol{x})\right\}$
  - Certain Combinatorial Optimization Problems (e.g. TSP)

Optimization by Random Sampling



- Maximize $y = f(x)$ with Only n Function Evaluations
- Randomly Generate n Solutions $x_i; i = 1, \ldots n$
- Evaluate $y_i = f(x_i); i = 1, \ldots n$
- Optimal Solution $x_j: j = argmax_{r=1 \ldots n}\, y_r$

Random Yet Directed Search



Resampling Near More Successful Ones

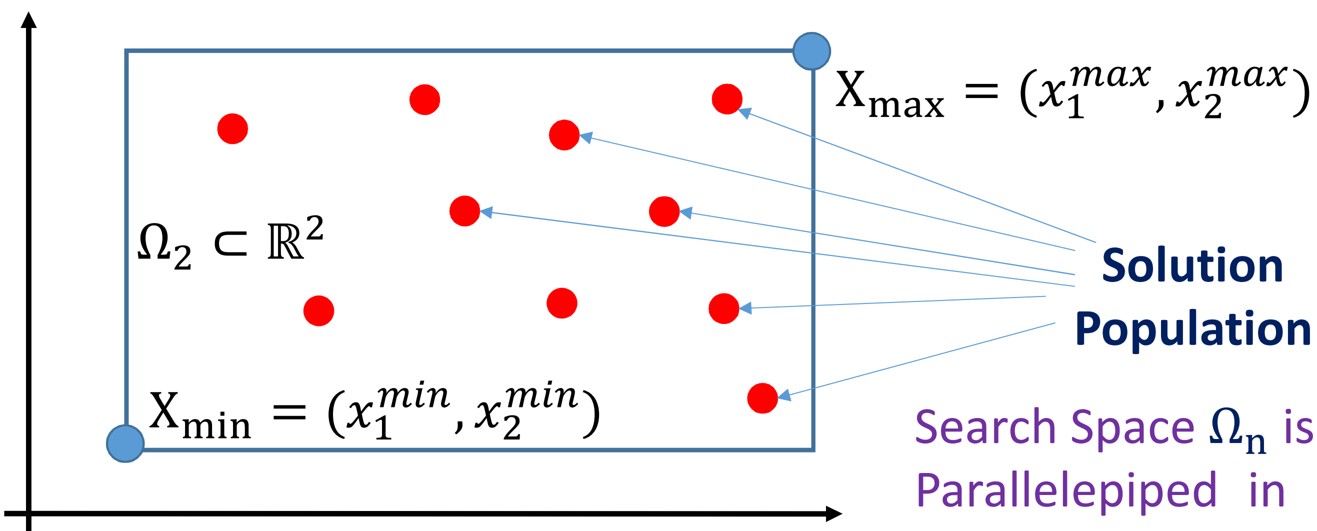
Stochastic Search (SS)

Maximize $y = f(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{R}^n$

Search Space $\Omega_n \subset \mathbb{R}^n$specified by User or Constraints

Search Space Ω_n defined by Vectors $\mathrm{X}_{\mathrm{min}}$ and $\mathrm{X}_{\mathrm{max}}$

Solution Population is Initialized and Iteratively Generated in Ω_n



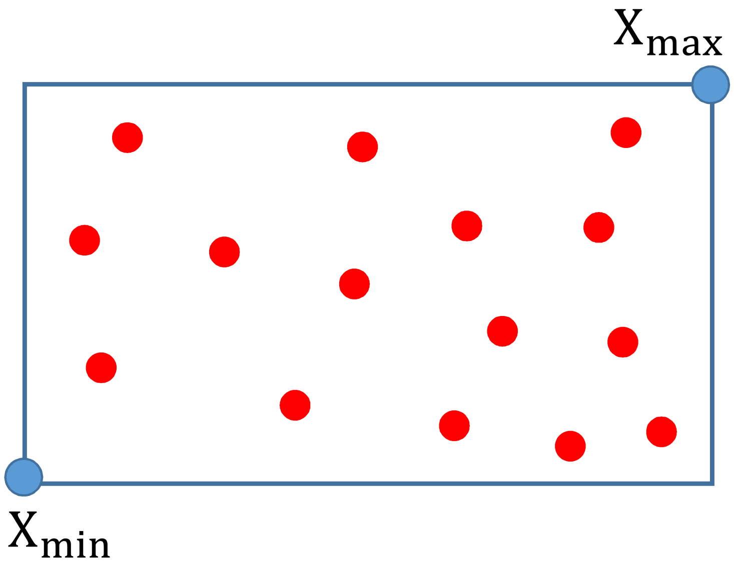
Search Space Ω_n is a Rectangle in $\mathbb{R}^2$; Rectangular Parallelepiped in $\mathbb{R}^3$ and a Hyper Rectangular Parallelepiped in $\mathbb{R}^n$

SS: Initialization

$S_t = \{\boldsymbol{x}_i^t; i = 1, \dots m\}$: Population (or Solution Set) at Iteration t

$Y_t = \{y_i^t = f(\boldsymbol{x}_i^t); i = 1, \dots m\}$: Set of Evaluated Solutions

$S_0 = \{\boldsymbol{x}_i^0; i = 1, \dots m\}$: Initial Population (or Solution Set)



$\boldsymbol{x}_i^t[j]$: Dimension j of $\boldsymbol{x}_i^t; j = 1, \dots n$

$$\boldsymbol{x}_i^0[j] = (1 - \alpha_{ij})X_{min}[j] + \alpha_{ij}X_{max}[j]$$

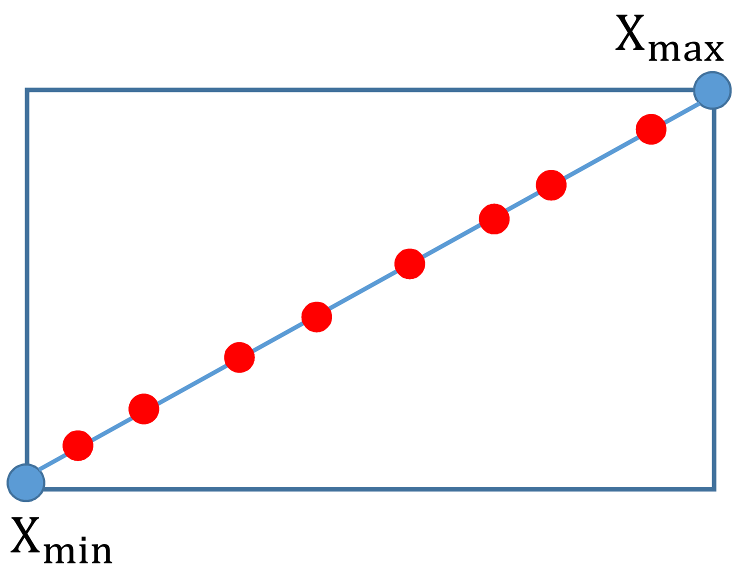
$\alpha_{ij} \in (0,1)$: Sampled from U(0,1)

SS: Initialization (Alternate)

$S_t = \{\boldsymbol{x}_i^t; i = 1, \dots m\}$: Population (or Solution Set) at Iteration t

$Y_t = \{y_i^t = f(\boldsymbol{x}_i^t); i = 1, \dots m\}$: Set of Evaluated Solutions

$S_0 = \{\boldsymbol{x}_i^0; i = 1, \dots m\}$: Initial Population (or Solution Set)



$$\boldsymbol{x}_i^0 = (1 - \alpha_i)X_{min} + \alpha_i X_{max}$$

$\alpha_i \in (0,1)$: Sampled from U(0,1)

SS: Objective Functions

SS is designed to Maximize $y = f(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{R}^n$

Optimization Problem: Minimize $g(\boldsymbol{x}); \boldsymbol{x} \in \mathbb{R}^n$

$$f(\boldsymbol{x}) = -g(\boldsymbol{x})$$

Equation Solving: $h(\boldsymbol{x}) = 0; \boldsymbol{x} \in \mathbb{R}^n$

$$f(\boldsymbol{x}) = -\{h(\boldsymbol{x})\}^2$$

SS: Evaluating Solution Fitness Scores

Recall: $S_t = \{\boldsymbol{x}_i^t; i = 1, \ldots m\}$; $Y_t = \{y_i^t = f(\boldsymbol{x}_i^t); i = 1, \ldots m\}$

Y_t may contain Negative Values as Well

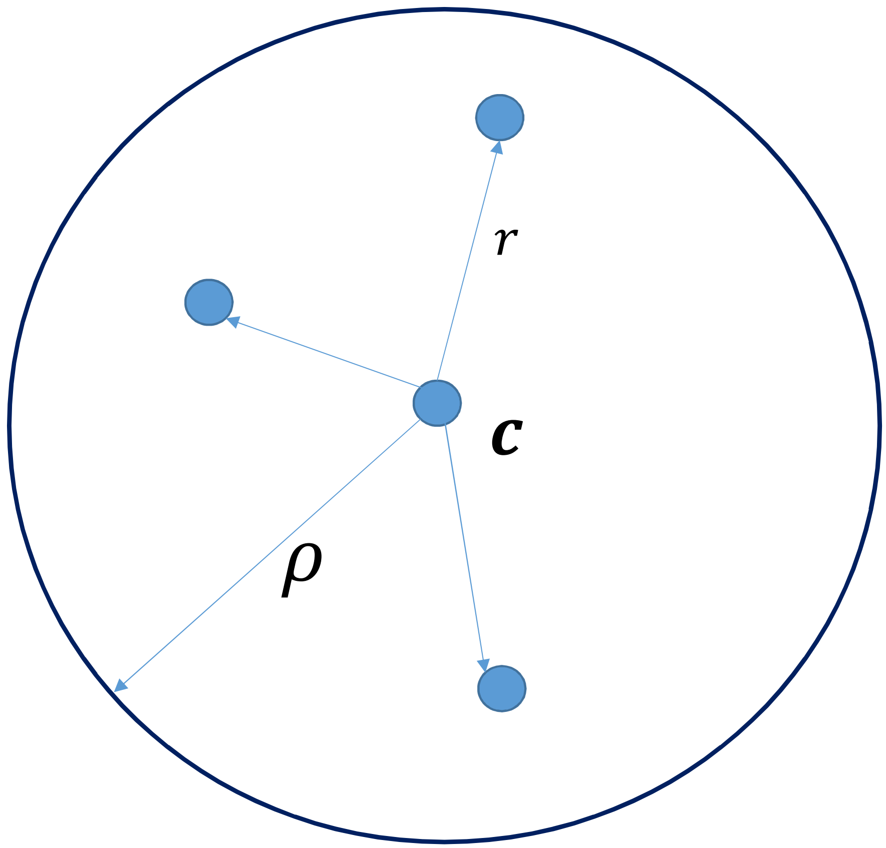
Transform y_i^t to Non-negative Scores u_i^t

$$U_t = \{u_i^t : u_i^t = y_i^t - \min(Y_t)\,; i = 1, \ldots m\}$$

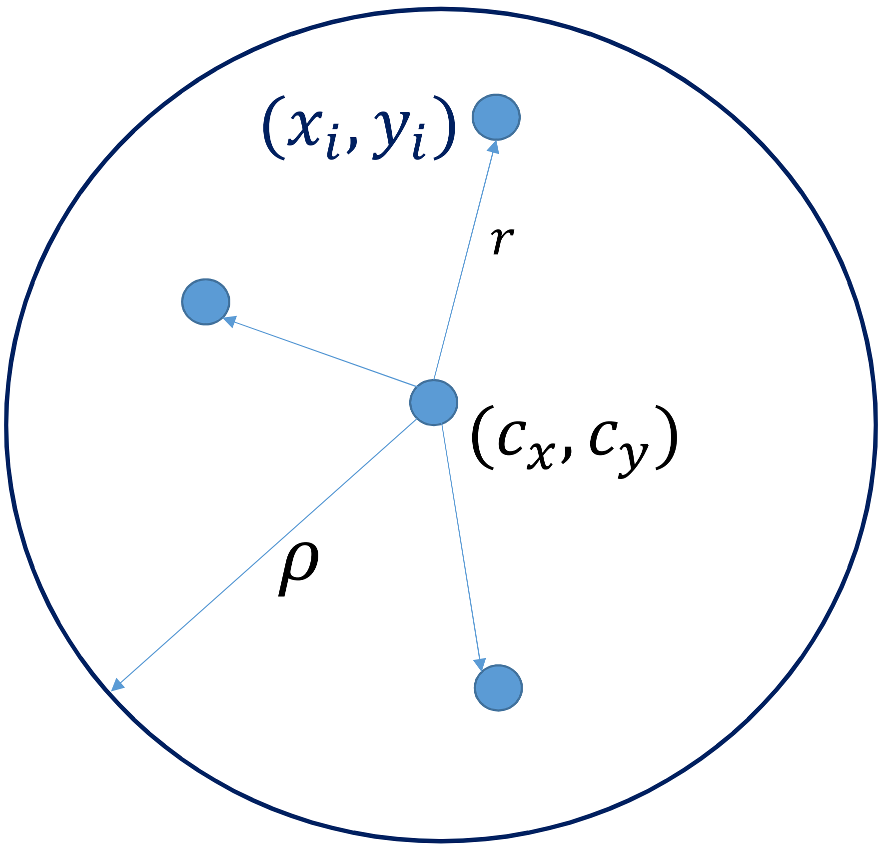
Transform u_i^t to Relative Solution Scores or Fitness Scores p_i^t

$$P_t = \left\{p_i^t : p_i^t = \frac{u_i^t}{\sum_{j=1}^m u_j^t}\,; i = 1, \ldots m\right\}$$

Generating Random Points within a Circle



Generating Random Points within a Circle



Generate Random $\theta_i \in [0,2\pi)$

Generate Random $r_i \in (0,\rho)$

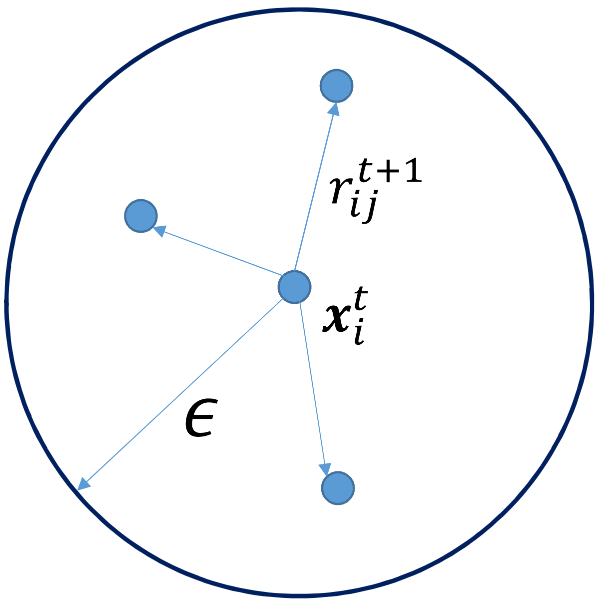
$$x_i = c_x + r_i \cos(\theta_i)$$

$$y_i = c_y + r_i \sin(\theta_i)$$

SS: Children Generation by Random Walk

$\boldsymbol{C}_i^{t+1}$: Set of Children of $\boldsymbol{x}_i^t$ with Fitness Score p_i^t

$$|\boldsymbol{C}_i^{t+1}| = ROUND(m \times p_i^t)$$



Generation of Child $\boldsymbol{x}_{ij}^{t+1} \in \boldsymbol{C}_i^{t+1}$

$$\boldsymbol{x}_{ij}^{t+1} = \boldsymbol{x}_i^t + r_{ij}^{t+1}$$

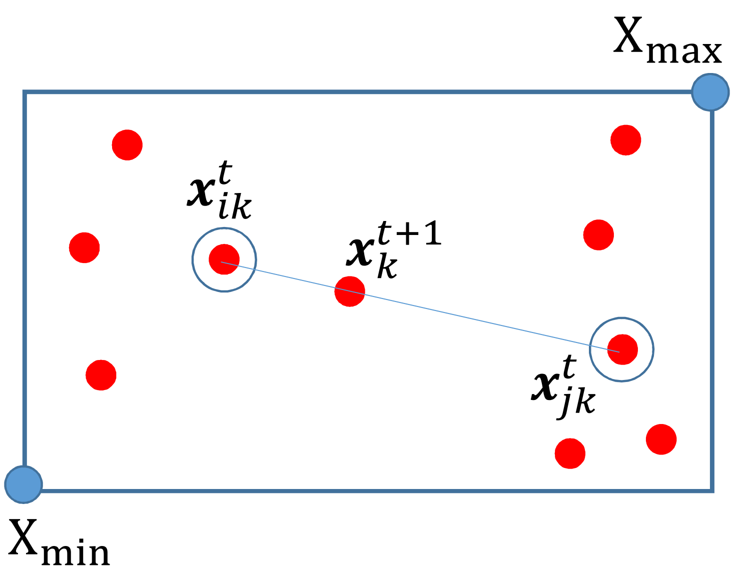
Sample $v_{ij}^{t+1}[d]$ from $\mathrm{U}(-1,1)$; $\mathrm{d} = 1, \ldots n$

Construct $r_{ij}^{t+1} = \lambda \dfrac{v_{ij}^{t+1}}{\left\| v_{ij}^{t+1} \right\|_2}; \lambda \in (0, \epsilon)$

SS: Children From Random Linear Combination

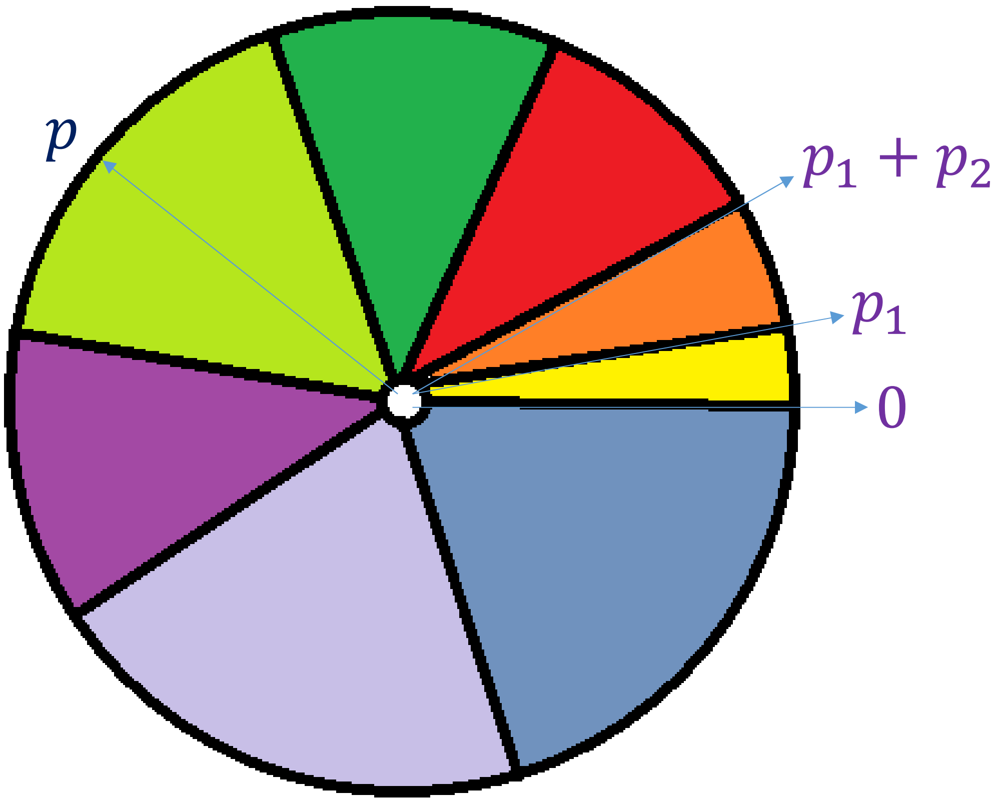
S_{t+1}^{RLC}: Set of Solutions formed by Random Linear Combination

$$S_{t+1}^{RLC} = \{\boldsymbol{x}_k^{t+1} : \boldsymbol{x}_k^{t+1} = (1-\alpha_k)\boldsymbol{x}_{ik}^t + \alpha_k \boldsymbol{x}_{jk}^t; \boldsymbol{x}_{ik}^t, \boldsymbol{x}_{jk}^t \in S_t\}$$



Solutions formed by Random Linear Combination Help in Interchange of Information between Solutions

Parent Selection: Roulette Wheel

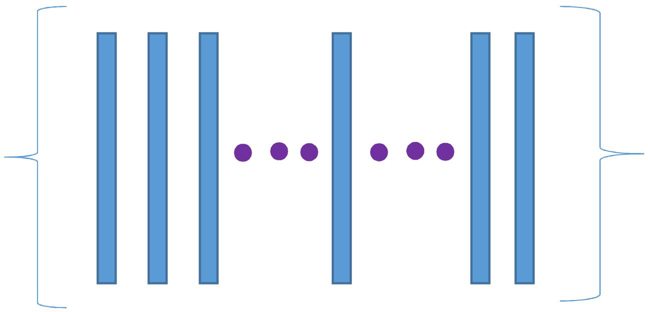


$p \in (0,1)$

Choose $\boldsymbol{x}_k$

$$\sum_{i=1}^{k-1} p_i < p < \sum_{i=1}^{k} p_i$$

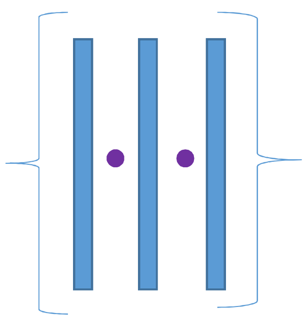
Parent Selection: Tournaments

$\boldsymbol{S} =$ 

$\boldsymbol{S}_k^i \subset \boldsymbol{S}$

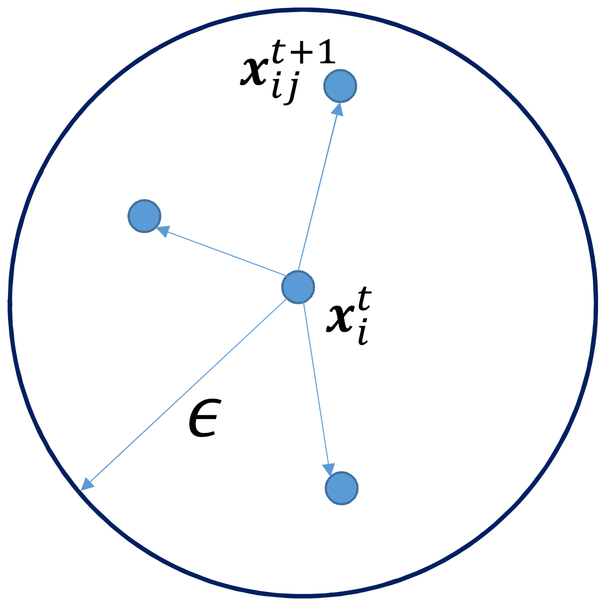
$k < |\boldsymbol{S}|$

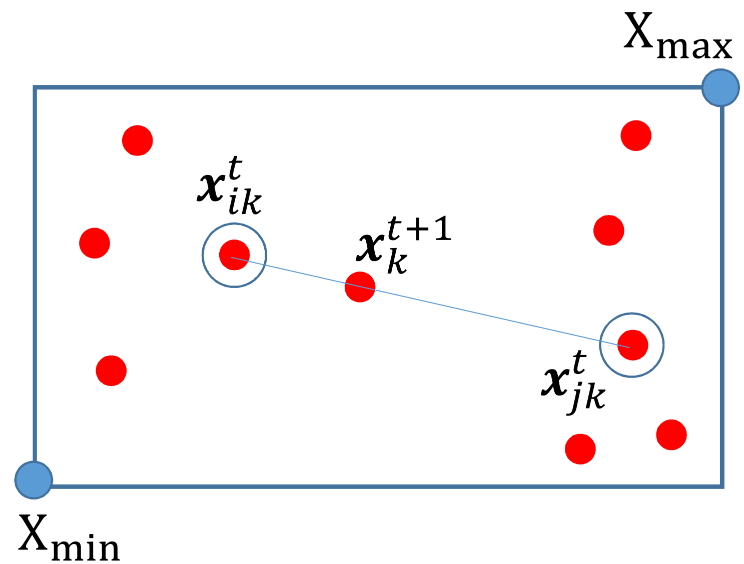
Random Selection of k-Solutions

$\boldsymbol{S}_{\boldsymbol{k}}^{(\boldsymbol{i})} =$ 

Choose the Best Solution from $\boldsymbol{S}_k^{(i)}$

SS: Exploitation and Exploration





Exploitation: Local Search by Dense Sampling in Small Region; Makes the Process Slow but Reaches Optimum

Exploration: Moves to Far Away Places in Search Spaces Looking for Near Optimum Regions Suitable for **Exploitation**

A Good Stochastic Search Execution Strikes a Balance between Exploration and Exploitation

SS: From S_t to S_{t+1}

$S_{t+1}^{RW} = \bigcup_{i=1}^{m} \boldsymbol{C}_i^{t+1}$: Solutions Generated by Random Walk

S_{t+1}^{RLC}: Solutions Generated by Random Linear Combination

S_{t+1}^{RRI}: Solutions Generated by Random Re-initialization

$$S_{t+1} = BEST_m \{S_t \cup S_{t+1}^{RW} \cup S_{t+1}^{RLC} \cup S_{t+1}^{RRI}\}$$

SS: Parameters and Execution

X_{min} and X_{max}: Search Space Bounds

m: Population or Solution Set Size

ϵ: Neighborhood Size for Children Generation

n_{RLC}: Number of Solutions Generated by Random Linear Combination

n_{RRI}: Number of Children Generated by Random Re-initialization

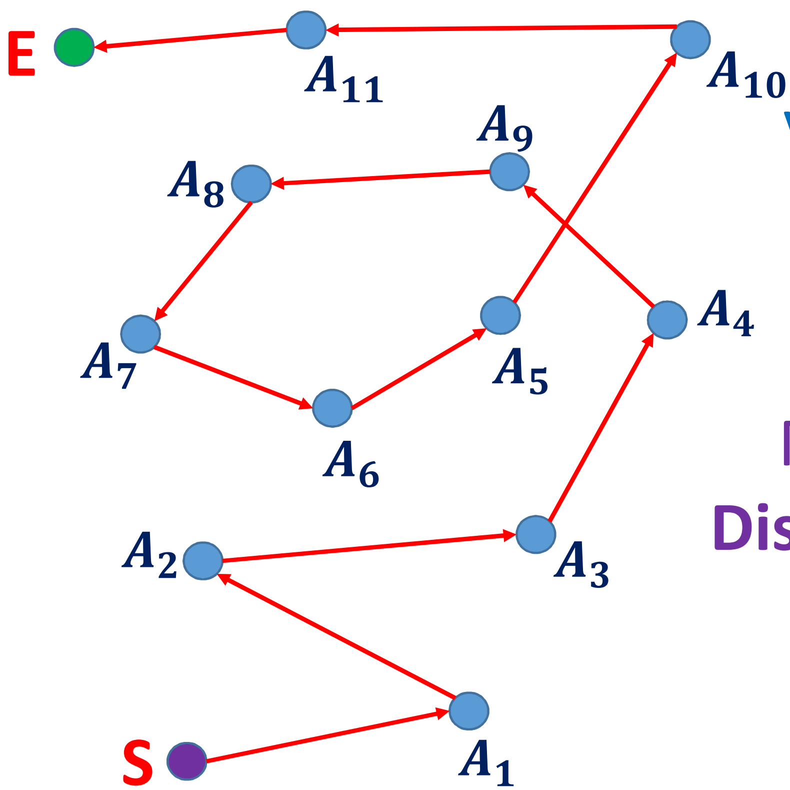
T_{max}: Maximum Number of Iterations $(t = 1, \ldots T_{max})$

Final Solution: Best Solution of $S_{T_{max}}$

Stochastic Search: Algorithm

- Initialize Population S_0 ($|S_0| = m$) in Search Space (X_{min}, X_{max})
- Evaluate Y_0 for Solution Population S_0
- FOR $t = 0 \to T_{max}$
  - Generate Non-negative Set U_t from Y_t
  - Evaluate P_t by Sum Normalizing U_t
  - Generate Children Population S_{t+1}^{RW} , S_{t+1}^{RLC} and S_{t+1}^{RRI}
  - Evaluate Y_{t+1}^{RW} , Y_{t+1}^{RLC} and Y_{t+1}^{RRI}
  - Generate $\{S_{t+1}, Y_{t+1}\} = \mathrm{BEST_m}\left\{ S_t \cup S_{t+1}^{RW} \cup S_{t+1}^{RLC} \cup S_{t+1}^{RRI}\right\}$
- END FOR
- Best Solution: $\mathrm{BEST}\left\{S_{T_{max}}, Y_{T_{max}}\right\}$

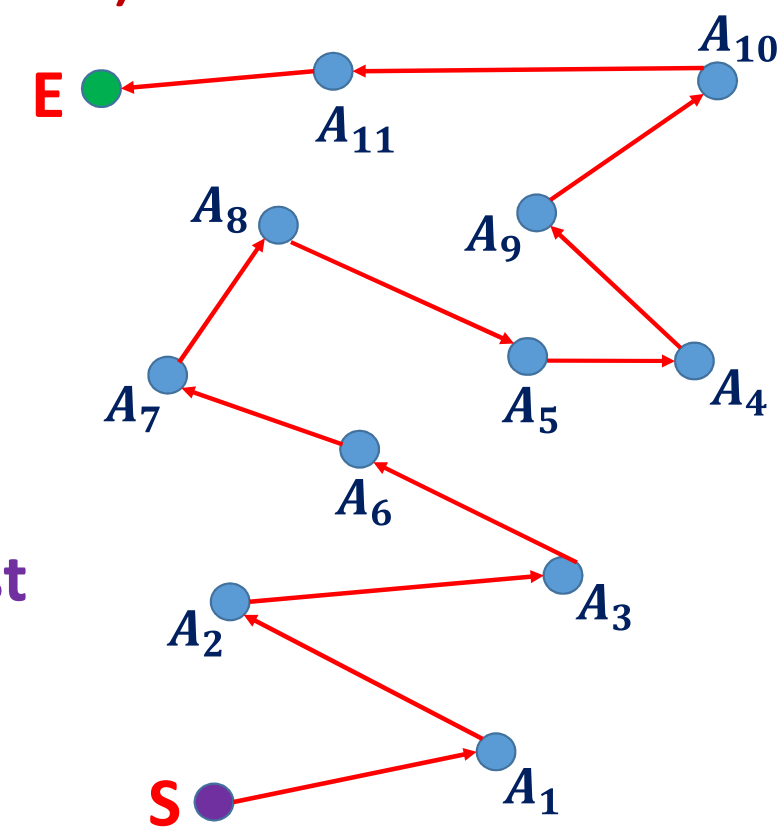
Travelling Salesman Problem (TSP)



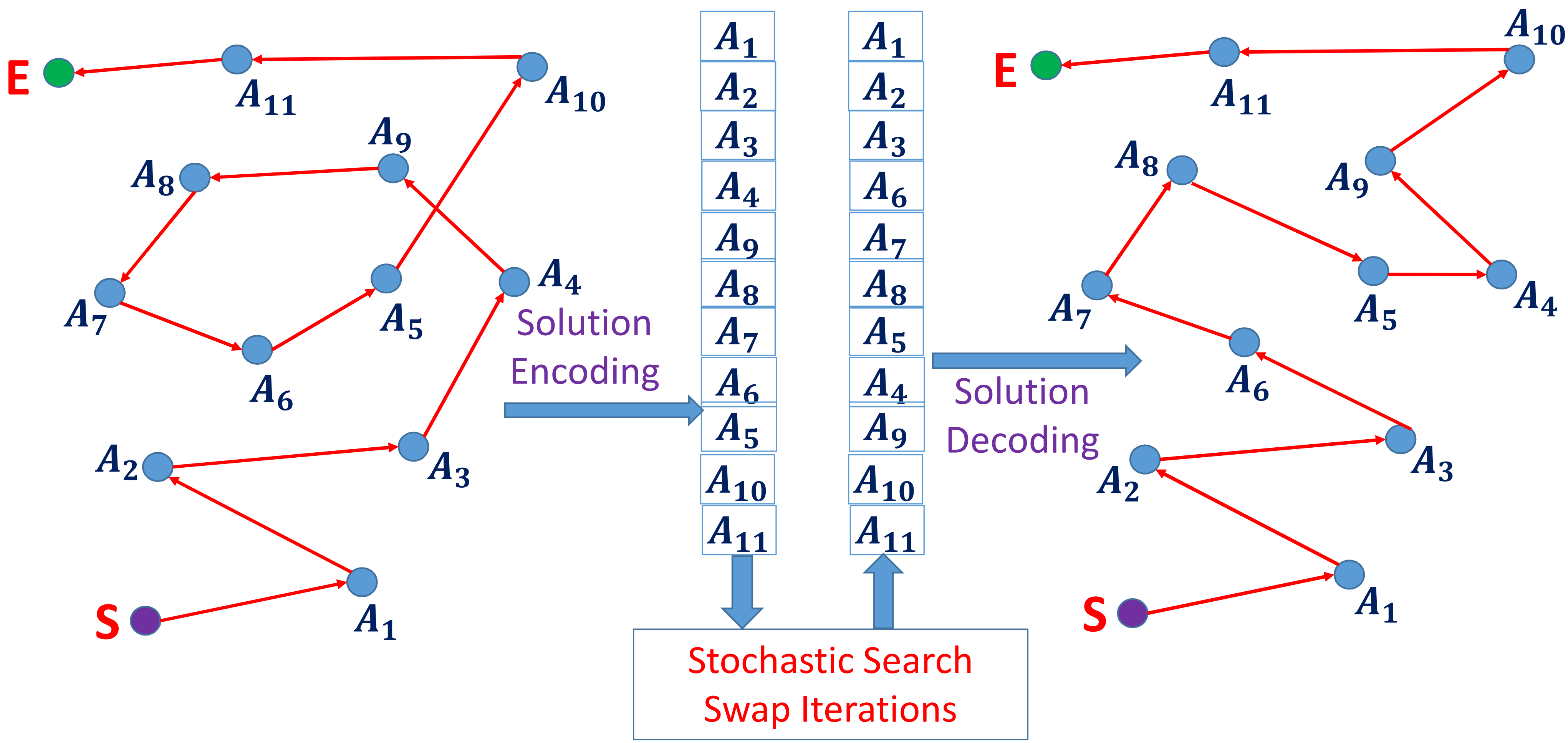
Visit All Stations
From **S** to **E**

Minimize Travel
Distance/Time/Cost

Explore
Station Sequences



TSP: Stochastic Search Formulation



TSP: Solution Encoding and Objective Function

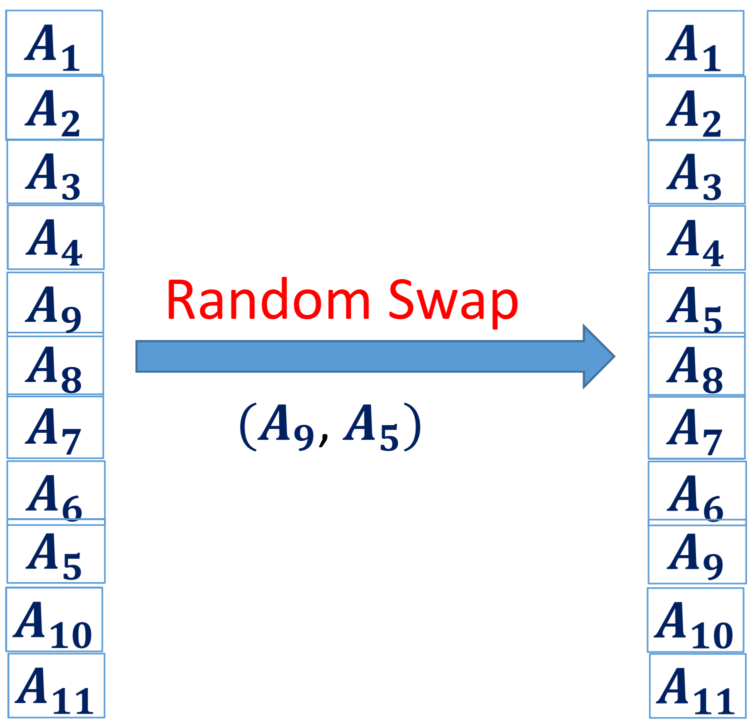
$$S_t = \left\{ \begin{array}{ccccc} \boxed{A_1} & & \boxed{A_1} & & \boxed{A_1} \\ \boxed{A_2} & & \boxed{A_2} & & \boxed{A_2} \\ \boxed{A_3} & & \boxed{A_3} & & \boxed{A_3} \\ \boxed{A_4} & & \boxed{A_5} & & \boxed{A_7} \\ \boxed{A_9} & \bullet\;\bullet\;\bullet\;\bullet & \boxed{A_9} & \bullet\;\bullet\;\bullet\;\bullet & \boxed{A_8} \\ \boxed{A_8} & & \boxed{A_8} & & \boxed{A_6} \\ \boxed{A_7} & & \boxed{A_7} & & \boxed{A_5} \\ \boxed{A_6} & & \boxed{A_4} & & \boxed{A_4} \\ \boxed{A_5} & & \boxed{A_5} & & \boxed{A_9} \\ \boxed{A_{10}} & & \boxed{A_{10}} & & \boxed{A_{10}} \\ \boxed{A_{11}} & & \boxed{A_{11}} & & \boxed{A_{11}} \end{array} \right\}$$

Permutation of Stations as Solutions

$$Y_t = \left\{ \boxed{d_1} \;\bullet\;\bullet\;\bullet\;\bullet\; \boxed{d_i} \;\bullet\;\bullet\;\bullet\;\bullet\; \boxed{d_m} \right\}$$

Corresponding Negated Distance Values

TSP: Children Generation and Iterations



Consider a Population of Size m. A Solution X with Fitness Score p has $k = ROUND(m \times p)$ children.

These k Children will be generated by performing k Swaps on the Parent Solution X.

Genetic Algorithms (GA)

- Nature Inspired Evolutionary Algorithm
- Similar to Stochastic Search in Operation
- Binary Solution Encoding (Bit Strings)
- Uses Genetic Operators for Children Generation
- Advanced Algorithms use other Evolutionary Strategies

GA: Binary Solution Encoding

Real or Integer Valued Solutions are Encoded using Bit Strings

1.35 (Multiply by 100 to obtain the integer 135)

$\downarrow$

1 0 0 0 0 1 1 1

Binary Coded Solution Representation

GA: Binary Crossover

$$X_1 = 1.35\ (10000111)_2 \text{ and } X_2 = 0.45\ (00101101)_2$$

Both Parents are Multiplied by 100 for Uniform Binary Encoding

1.35

1 0 0 0 0 1 | 1 1

0 0 1 0 1 1 | 0 1

0.45

Crossover Point

Binary Crossover

1.33

1 0 0 0 0 1 0 1

0 0 1 0 1 1 1 1

0.47

Children

$$P_1 + P_2 = C_1 + C_2$$

Parents are Chosen by Tournament Selection

Crossover Point Near LSB: Exploitation

GA: Binary Crossover

$$X_1 = 1.35\ (10000111)_2 \text{ and } X_2 = 0.45\ (00101101)_2$$

Both Parents are Multiplied by 100 for Uniform Binary Encoding

1.35

1 0 | 0 0 0 1 1 1

0 0 | 1 0 1 1 0 1

0.45

Crossover Point

Binary Crossover →

1.73

1 0 1 0 1 1 0 1

0 0 0 0 0 1 1 1

0.07

Children

Parents are Chosen by Tournament Selection

$$P_1 + P_2 = C_1 + C_2$$

Crossover Point Near MSB: Exploration

GA: Mutation

$$X_1 = 1.35\ (10000111)_2$$

The Parent is Multiplied by 100 for Binary Encoding

1.35

1 0 0 0 0 1 1 1

Binary Mutation

1.33

1 0 0 0 0 1 0 1

Mutation Point

- Mutation is Flipping a Bit from 1 to 0 or Vice Versa
- Exploitation: Mutation Point Near LSB
- Parent is Selected using Tournament Selection

GA: Mutation

$$X_1 = 1.35\ (10000111)_2$$

The Parent is Multiplied by 100 for Binary Encoding

1.35

1 0 0 0 0 1 1 1

Mutation Point

Binary Mutation

1.99

1 1 0 0 0 1 1 1

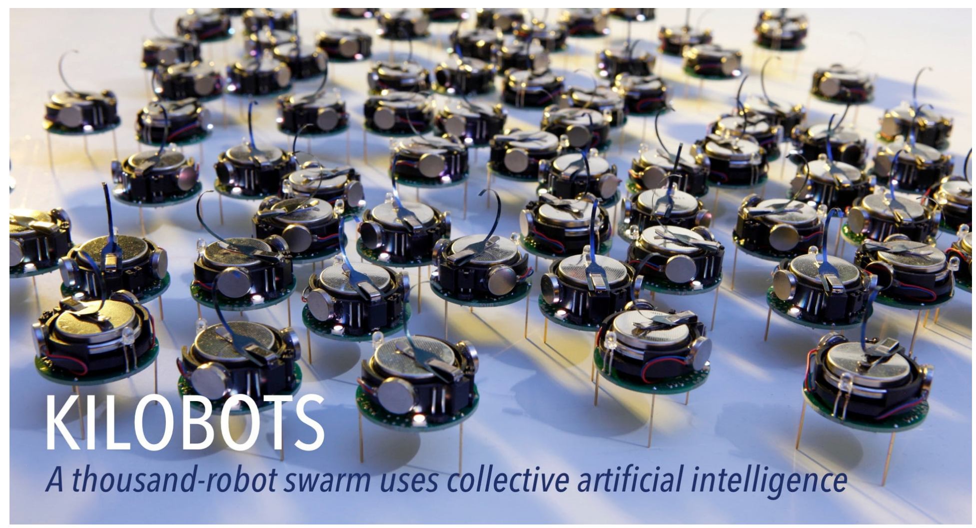
- Mutation is Flipping a Bit from 1 to 0 or Vice Versa
- Exploration: Mutation Point Near MSB
- Parent is Selected using Tournament Selection

Genetic Algorithm

- Initialize Population S_0 ($|S_0| = m$) in Search Space (X_{min}, X_{max})
- Evaluate Y_0 for Solution Population S_0
- FOR $t = 0 \rightarrow T_{max}$
  - Generate Children Population S_{t+1}^{BC} ($|S_{t+1}^{BC}| = m_{BC}$) using Binary Crossover
  - Generate Children Population S_{t+1}^{BM} ($|S_{t+1}^{BM}| = m_{BM}$) using Binary Mutation
  - Decide m_{BC} and m_{BM} ($m_{BC} + m_{BM} = m$) based on Algorithm Progress
  - Evaluate Y_{t+1}^{BC} and Y_{t+1}^{BM}
  - Generate $\{S_{t+1}, Y_{t+1}\} = \mathrm{BEST_m}\left\{S_t \cup S_{t+1}^{BC} \cup S_{t+1}^{BM}\right\}$
- END FOR
- Best Solution: $\mathrm{BEST}\left\{S_{T_{max}}, Y_{T_{max}}\right\}$

Discussions

- SS & GA are both Evolutionary Algorithms
- Solution Breeding from One Generation to Next
- ``Survival of the Fittest” & ``Selection of Better Parents”
- BC-GA and RLC-SS Interchange Information
- Exploration & Exploitation can be achieved by Changing Parameters
- BM-GA (near LSB) and RW-SS Perform Exploitation
- BM-GA (near MSB) and RRI-SS Perform Exploration
- Start Exploring Search Space when Algorithm Progress gets Stalled
- Solutions Do Not Aggressively ``Seek’’ the Maxima


KILOBOTS
A thousand-robot swarm uses collective artificial intelligence

Swarming Behavior

- Try to Follow the Neighbor's Collective Behavior (Motion)
- Try to Remain Close to Neighbors (Proximity)
- Avoid Collisions with Neighbors (Good Neighbor)
- Loosely Following a Leader (Global Goal)

Particle Swarm Optimization (PSO) Algorithm

James Kennedy & Russell Eberhart (1995)

Discussions

- Swarm of particles
- Each particle residing at a **position** in the search space
- **Fitness** of each particle = the quality of its position
- Particles fly over the search space with a certain velocity
- **Velocity** (both direction and speed) of each particle is influenced by its own best position found so far and the best solution that was found so far by its **neighbors.**
- Eventually the swarm will converge to **optimal** positions.

Notations

$\boldsymbol{S}_t = \left\{\boldsymbol{q}_i^{(t)}; i = 1, \dots m\right\}$: Population or Solution Set at iteration t

$\boldsymbol{q}_i^{(t)} = \left\langle \boldsymbol{x}_i^{(t)}, \boldsymbol{v}_i^{(t)}, \boldsymbol{p}_i^{(t)}, y_i^{(t)} \right\rangle$: The Particle i at iteration t

$\boldsymbol{x}_i^{(t)}$: Position of Particle i at iteration t

$\boldsymbol{v}_i^{(t)}$: Velocity of Particle i at iteration t

$\boldsymbol{p}_i^{(t)}$: Best Position of Particle i till Iteration t

Notations

$y_i^{(t)}$: Fitness Score of Particle i at Iteration t

$\boldsymbol{g}_t$: Global Best Particle Position at Iteration t

ϕ_p : Cognitive Component of Particle

ϕ_g : Social Component of Particle

ω : Momentum Component for Particle Velocity

$\boldsymbol{X}_{min}$ and $\boldsymbol{X}_{max}$: Search Space Bounds

PSO: Initialization

$$S_0 = \{\boldsymbol{q}_i^{(0)}; i = 1, \dots m\}$$

$$\boldsymbol{x}_i^{(0)}[j] = \left(1 - \alpha_{ij}\right) X_{min}[j] + \alpha_{ij} X_{max}[j]$$

$$\left[\alpha_{ij} \sim \mathrm{U}(0,1)\right] \wedge [j = 1, \dots n]$$

$$\boldsymbol{v}_i^{(0)} = 0$$

$$\boldsymbol{p}_i^{(0)} = \boldsymbol{x}_i^{(0)}$$

$$y_i^{(0)} = f\left(\boldsymbol{x}_i^{(0)}\right)$$

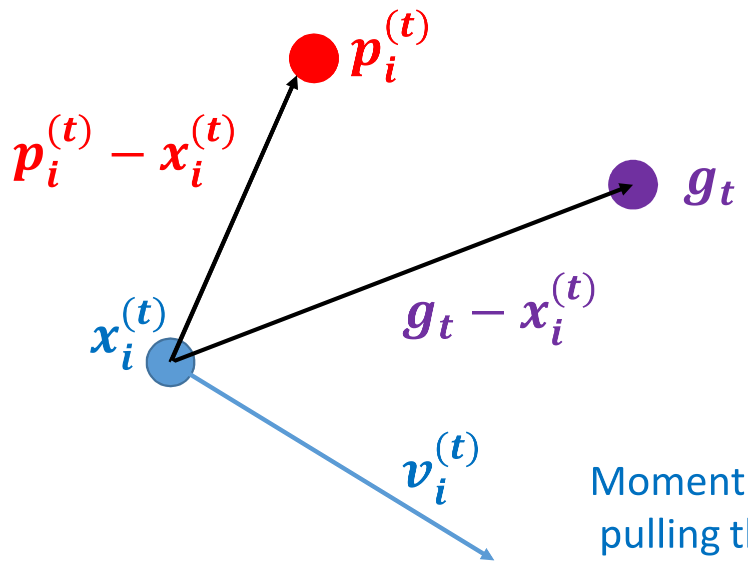
$$k = \underset{i=1,\dots m}{\operatorname{argmin}} \, y_i^{(0)} \Rightarrow g_0 = x_k^{(0)}$$

Objective Function: $y = f(\boldsymbol{x}); y \in \mathbb{R}^1; \boldsymbol{x} \in \mathbb{R}^n$

PSO: Particle Velocity Update

$$\boldsymbol{v}_i^{(t+1)}[j] = \omega \boldsymbol{v}_i^{(t)}[j] + \phi_p \beta_{ij} \left(\boldsymbol{p}_i^{(t)}[j] - \boldsymbol{x}_i^{(t)}[j] \right) + \phi_g \gamma_{ij} \left(\boldsymbol{g}_t - \boldsymbol{x}_i^{(t)}[j] \right)$$

Cognitive component: The force emerging from the tendency to return to its own best solution found so far.



Social component: The force emerging from the attraction of the best solution found so far

Momentum: The force pulling the particle to continue its current direction.

$\beta_{ij} \sim \mathrm{U}(0,1)$
$\gamma_{ij} \sim \mathrm{U}(0,1)$
$j = 1, \dots n$

PSO: Particle Update

Particle Position Update

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \boldsymbol{v}_i^{(t+1)}$$

Particle Fitness Update

$$y_i^{(t+1)} = f\left(\boldsymbol{x}_i^{(t+1)}\right)$$

Particle Best Position Update

$$\boldsymbol{p}_i^{(t+1)} = \left(y_i^{(t)} < f\left(\boldsymbol{p}_i^{(t)}\right)\right) ?\, \boldsymbol{x}_i^{(t+1)} : \boldsymbol{p}_i^{(t)}$$

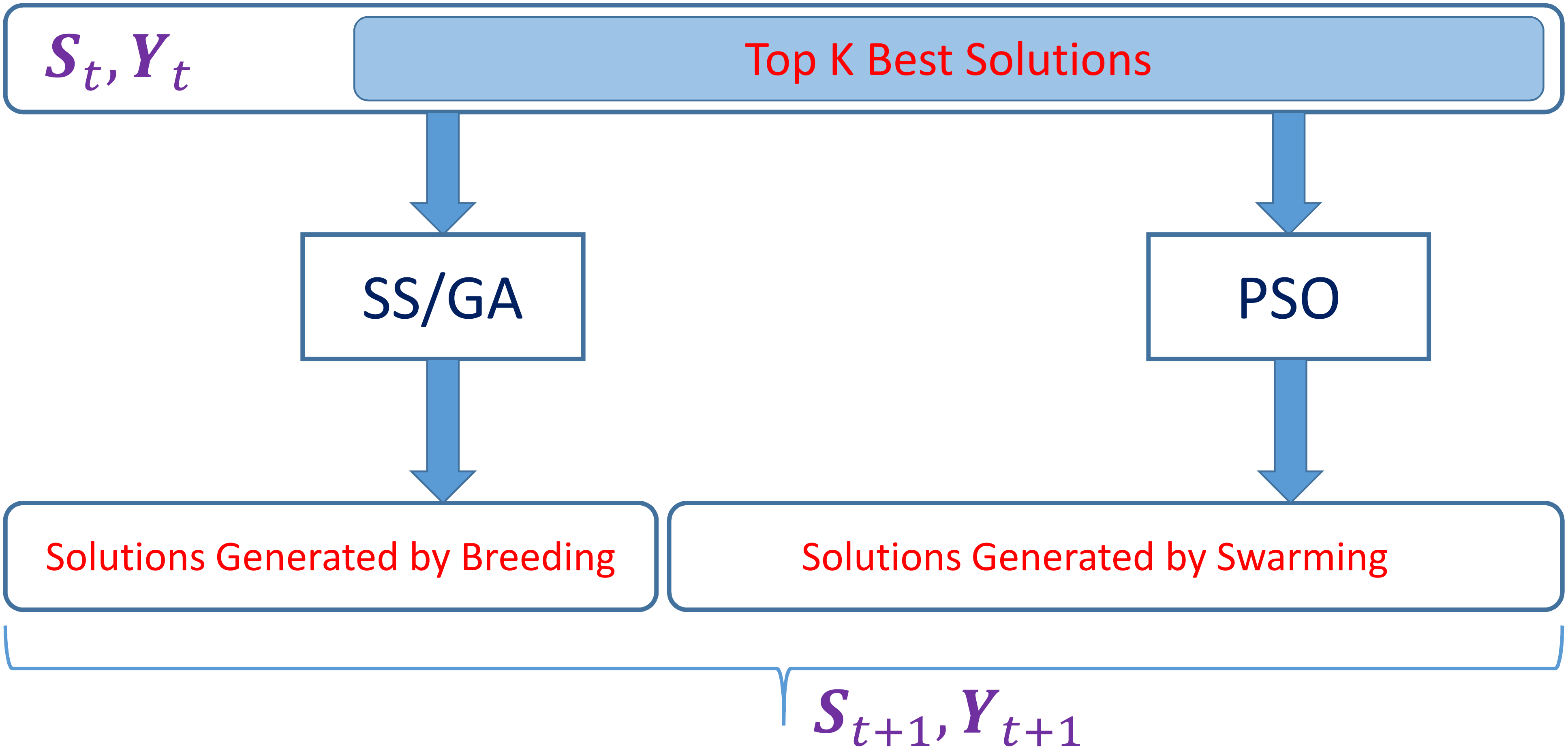
Global Best Position Update

$$k = \underset{i=1,\ldots m}{\operatorname{argmin}}\, f\left(\boldsymbol{p}_i^{(t+1)}\right) \Longrightarrow \boldsymbol{g}_{t+1} = \boldsymbol{p}_k^{(t+1)}$$

PSO: Algorithm

- Search Space: (X_{min}, X_{max})
- Initialize Particle Population S_0 ($|S_0| = m$)
- Evaluate the Global Best Position g_0 for S_0
- FOR $t = 1 \rightarrow T_{max}$
    - Update Particle Velocities $\boldsymbol{v}_i^{(t)}; i = 1, \dots m$
    - Update Particle Positions $\boldsymbol{x}_i^{(t)}; i = 1, \dots m$
    - Update Particle Fitness Scores $y_i^{(t)}; i = 1, \dots m$
    - Update Particle Best Positions $\boldsymbol{p}_i^{(t)}; i = 1, \dots m$
    - Evaluate the Global Best Position g_t for S_t
- END FOR
- Best Solution: $g_{T_{max}}, f\left(g_{T_{max}}\right)$

Breeding Swarms



Summary

- Application of Sampling Strategy in Optimization
- Two Classes of Algorithms
  - Evolutionary Algorithms (Solutions Breed)
  - Swarming Algorithms (Solutions Do Not Breed)
- Stochastic Search
- SS Approach to TSP
- Genetic Algorithm
- Particle Swarm Optimization

Thank You