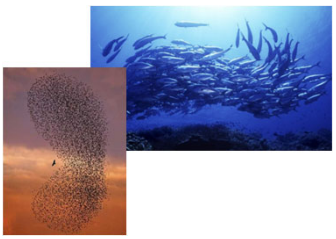
§ 4.4 Swarm Intelligence

- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)



Social insects: Examples

Leafcutter ants *(Atta)*

Weaver ants *(Oecophylla)*

Swarm Intelligence Development

Beni, a professor at the University of California, first proposed the concept of swarm intelligence in his *cellular automata system*.

Beni

A social insect colony is...

- **Flexible** – The colony can respond to internal perturbations and external challenges
- **Robust** – Tasks are completed even if some individuals fail
- **Self-organized** – Paths to solutions are emergent rather than predefined
- **Decentralized** – There is no central control(ler) in the colony

Swarm Intelligence (SI)

- A computational technique for solving distributed problems inspired from biological examples provided by
  - social insects such as ants, termites, bees, and wasps and by herd, flock, and
  - shoal phenomena such as fish shoals
- An approach to controlling and optimizing distributed systems
- Resilient, decentralized, self-organized technique

SI Organizing Principles

SI has the following notable features:

- Autonomy

The system does not require outside management or maintenance. Individuals are autonomous, controlling their own behavior both at the detector and effector levels in a self-organized way.

- Adaptability

Interactions between individuals can arise through direct or indirect communication.

Features

SI Organizing Principles

SI has the following notable features:

- Scalability:

SI abilities can be performed using groups consisting of a few, up to thousands of individuals with the same control architecture.

- Flexibility:

No single individual of the swarm is essential, that is, any individual can be dynamically added, removed, or replaced.

Features

SI Organizing Principles

SI has the following notable features:

- Robustness:

No central coordination takes place, which means that there is no single point of failure

- Massively parallel:

Tasks performed by each individual within its group are the same

- Self-organization:

The intelligence exhibited is not present in the individuals, but rather emerges somehow out of the entire swarm.

Communication

SI Communication Forms

Indirect Communication

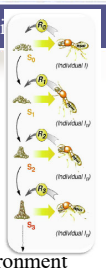
- Implicit communication that takes place between individuals via the surrounding environment.
- Known as Stigmergy communication..

Direct Communication

- Explicit communication that can also take place between individuals.
- Examples:
  - waggle dance of the honeybee,
  - trophallaxis (food or liquid exchange, e.g., mouth-to-mouth food exchange in honeybees),
  - …

Sti

Stigmergy



- Stigmergy: stigma (sting) + ergon (work)

  "stimulation by work"

- Characteristics of stigmergy

➢ Indirect agent interaction modification of the environment

➢ The information is local: it can only be accessed by insects that visit the locus in which it was released

➢ Work can be continued by any individual

Application

SI: Main application

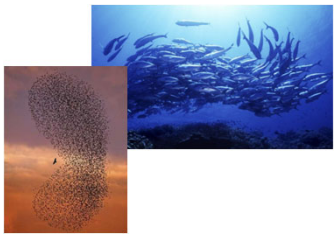
- SI principles have been successfully applied in a variety of problem domains and applications:
  - ➢ Ant colony optimization (ACO),

    Which focuses on discrete optimization problems
  - ➢ Particle swarm optimization (PSO)

    Which focuses on nonlinear optimization problems with constraints

§ 4.4 Swarm Intelligence

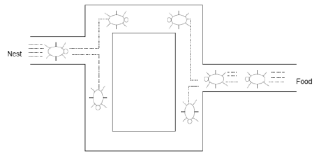
- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)



Ant

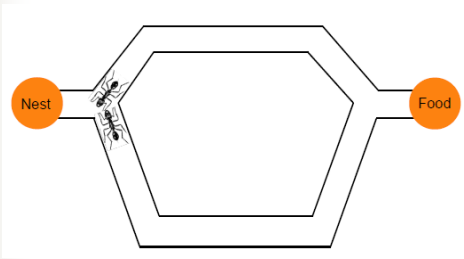
Ant

- Why are ants interesting?
  - Ants solve complex tasks by simple local means.
  - Ants productivity is better than the sum of their single activities.
  - Ants are grand masters in search and exploitation.



Foraging Behavior

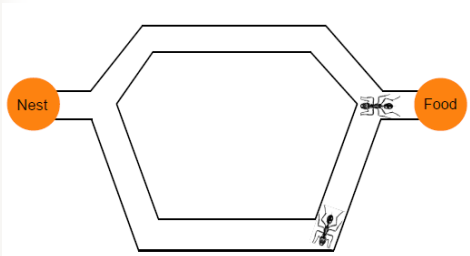
Foraging behavior of Ants (Double bridge experiment)



Ants start with equal probability

Foraging Behavior

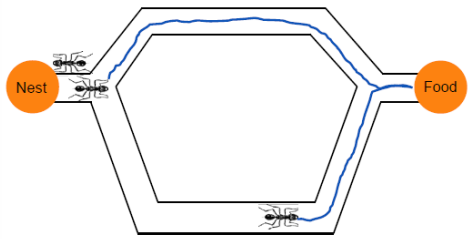
Foraging behavior of Ants (Double bridge experiment)



The ant on shorter path has a shorter to-and-fro time from it's nest to the food

Foraging Behavior

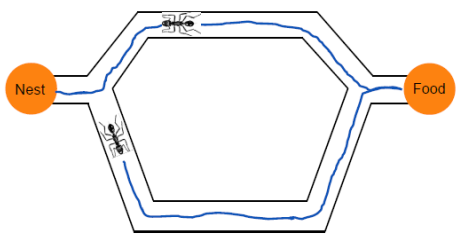
Foraging behavior of Ants (Double bridge experiment)



The density of pheromone on the shorter path is higher

Foraging Behavior

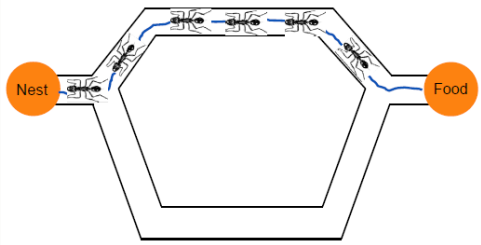
Foraging behavior of Ants (Double bridge experiment)



The next ant takes the shorter route

Foraging Behavior

Foraging behavior of Ants (Double bridge experiment)



After some time, the shorter path is almost exclusively used

Heuristic

From nature to computers

- In an iterative fashion, each ant moves from state S_i to state S_j guided by two main factors:

1. Heuristic information:

- A measure of the heuristic preference for moving from state S_i to state S_j
- This information is known *a priori* to the algorithm run, and is not modified during

Pheromone Trail

From nature to computers

2. Artificial pheromone trail:

- A measurement of the pheromone deposition from ants previous transitions from state S_i to state S_j
- This information is modified during the algorithm run by the artificial ants

Main Functions

Ant Colony Optimization (ACO)

- It is a population-based metaheuristic used to find approximate solutions to difficult optimization problems
- ACO is structured into two main functions:

1. Ant Solutions Construct ( ):

– Performs the solution construction process

2. Pheromone Update ( )

– Performs pheromone trail updates

– Includes also pheromone trail evaporation

Metaheuristic

What is Metaheuristic?

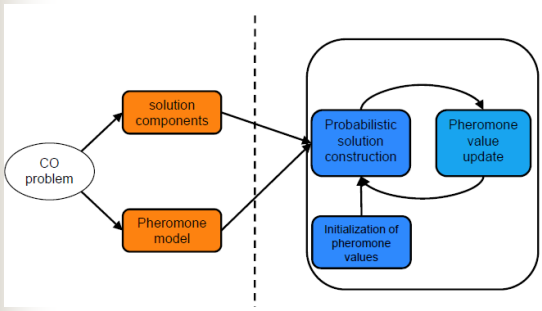
- "A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems"
- In other words: "a metaheuristic is a *general-purpose algorithmic framework* that can be applied to different optimization problems with relatively few modifications"

Examples

Simulated annealing; Tabu search; Iterated local search

Metaheuristic

What is Metaheuristic?



Properties

Properties of the artificial ant

- Each artificial ant has an *internal memory*
- Starting in an initial state Sinitial each ant tries to build a *feasible solution* to the given problem, moving in an iterative fashion through its search space
- The guidance factors for ants movement take is *a transition rule* which is applied before every move from state S_i to state S_j

Properties

Properties of the artificial ant

- The amount of pheromone each ant deposits is governed by a problem specific *pheromone update rule*
- Pheromone deposition may occur at every state transition during the solution construction (pheromone trial update)
- Ants may retrace their paths once a solution has been constructed and only then deposit pheromone, all along their individual paths

Ant System

The original Ant System

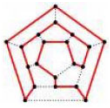
- Developed by Dorigo et al. (1996)
- Ant system (AS)
  - First ACO algorithm
  - Pheromone updated by all ants in the iteration
- Travelling Salesman Problem (TSP) was used as a test-bed for this algorithm

TSP

Travelling Salesman Problem (TSP)

- TSP description:
  - Visit cities in order to make sales
  - Save on travel costs
  - Visit each city once

A Hamiltonian cycle is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex.



TSP

Solution for TSP

- A connected graph $G=(V, E)$, where
  - V is a set of vertices (cities)
  - E is a set of edges (connection between cities)
- A variable called pheromone is associated with each edge and can be read and modified by ants

TSP

Solution for TSP

- Ant system is an *iterative algorithm* at each iteration
  - A number of artificial ants are considered
  - Each ant build a solution by walking from vertex to vertex
  - Each vertex is visited one time only
  - An ant selects the following vertex to be visited according to a stochastic mechanism that is biased by the pheromone
- At the end, the pheromone values are updated in order to bias ants in the future iteration to construct solutions similar to the previously constructed

TSP

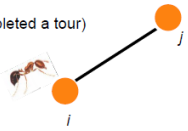
Ant System and the TSP

- The following *steps* is used to solve the TSP:
  - Pheromone trail
  - Memory
  - Awareness of environment
  - Probability function

Ant System and the TSP

1. Pheromone trail
   - Iteration is defined as the interval in *(t,t+1)* where each of the *N* ants moves once
   - Epoch➔ *n* iterations (when each ant has completed a tour)
   - Intensity of trail: $\tau_{ij}(t)$
   - Trail update function after each epoch:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{N}\Delta\tau_{ij}^{k}$$



   - ρ is the evaporation rate
   - $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on path *(i,j)* by the ant k and is given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{If ant } k \text{ used edge } (i,j) \text{ in its tour,} \\ 0 & \text{otherwise,} \end{cases}$$

where Q is a constant and L_k is the tour length of *k*th ant.

Ant System and the TSP

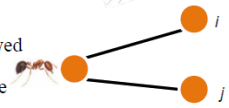
2. Memory
   - Prevents town repeats
   - Tabu list

3. Awareness of environment
   - City distance
   - Visibility: $\eta_{ij} = \frac{1}{d_{ij}}$

   where $d_{ij} = \sqrt{\left[(x_i - x_j)^2 + (y_i - y_j)^2\right]}$

4. Probability function

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{j\in\text{allowed}}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & \text{if } j \in \text{allowed} \\ 0 & \text{otherwise} \end{cases}$$



Ant System and the TSP

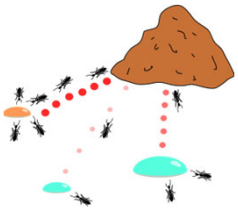
Various improvements were made which gave rise to several other ant algorithms which collectively form the main ACO algorithms.

such as:

– Max–Min Ant System

– Ant Colony System

– …



Max–Min Ant System (MMAS)

- Developed by Stutzle and Hoos
- The differences between MMAS and AS are:
  - Only best ant updates pheromone
  - Pheromone trail values are restricted to an interval $[\tau_{\min}, \tau_{\max}]$
  - Trails are initialized to their maximum value $\tau_{\max}$
  - The modified pheromone update is as follows:

$$\tau_{ij} \leftarrow \left[(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{best}\right]_{\tau_{\min}}^{\tau_{\max}}$$

where

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/L_{best} & \text{if } (i,j) \text{ belongs to the best tours} \\ 0 & \text{otherwise} \end{cases}$$

Ant Colony System (ACS)

- Developed by Dorigo & Gambardella
- The differences between ACS and AS are:
  - Pheromone update is done by all ants after each construction step only to last edge traversed.
  - Two pheromone update functions:

Local pheromone update: $\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi.\tau_0$
where $\varphi \in (0,1]$ is the pheromone deacy function and τ_0 is the initial value of the pheromone.

Offline pheromone update:

$$\tau_{ij} = \begin{cases} (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij} & \text{if ant (i, j) belongs to best tour,} \\ 0 & \text{otherwise,} \end{cases}$$

Solving a Problem by ACO

- Represent the problem in the form a weighted graph, on which ants can build solutions
- Define the meaning of the pheromone trails
- Define the heuristic preference for the ant while constructing a solution
- Choose a specific ACO algorithm and apply to problem being solved
- Tune the parameters of the ACO algorithm

Applications

Some applications of ACO

- Scheduling
- Routing problems
  - Traveling Salesman Problem (TSP)
  - Vehicle routing
  - Network routing

Comparison

Ant Foraging and ACO

Biology (Ant Foraging)	ACO Algorithm
Ant	Individual (agent) used to build (construct) a solution
Ant Colony	Population (colony) of cooperating individuals
Pheromone Trail	Modification of the environment caused by the artificial ants in order to provide an indirect mean of communication with other ants of the colony. Allows assessment of the quality of a given edge on a graph
Pheromone Evaporation	Reduction in the pheromone level of a given path due to aging

Conclusion

Conclusion

SI is a rich source of inspiration for our computer systems, has many features that are desirable for distributed computing such as *auto-configuration*, *auto-organization*, *autonomy*, *scalability*, *flexibility, robustness, emergent behavior, and adaptability.*

- ACO is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems.
- Artificial ants implement a randomized construction heuristic which makes probabilistic decisions.

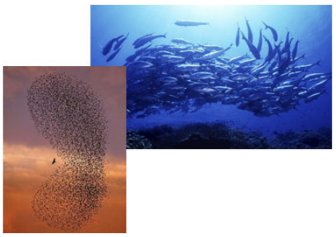
Conclusion

Conclusion

- The *cumulated search experience* is taken into account by the adaptation of the *pheromone trail*.
- ACO shows great performance with the "ill-structured" problems like *network routing*.
- In ACO, local search is extremely important to obtain good results.

§ 4.4 Swarm Intelligence

- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)



Contents

Contents

- Introduction to Particle Swarm Optimization (PSO)
  - Origins
  - Concept
  - PSO Algorithm
- PSO for the Bin Packing Problem (BPP)
  - Problem Formulation
  - Algorithm
  - Simulation Results

Origins

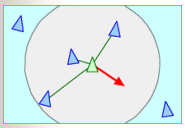
Introduction to the PSO: **Origins**

- *Inspired from the nature* social behavior and dynamic movements with communications of insects, birds and fish

Origins

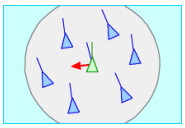
Introduction to the PSO: **Origins**

- In 1986, Craig Reynolds described this process in 3 simple behaviors:



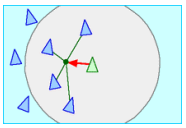
Separation

avoid crowding local flockmates



Alignment

move towards the average heading of local flockmates



Cohesion

move toward the average position of local flockmates

Origins

Introduction to the PSO: **Origins**

- Application to optimization: Particle Swarm Optimization
- Proposed by James Kennedy & Russell Eberhart (1995)
- Combines self-experiences with social experiences

Concept

Introduction to the PSO: **Concept**

- Uses a number of agents (**particles**) that constitute a swarm moving around in the search space looking for the best solution
- Each particle in search space adjusts its "flying" according to its own flying experience as well as the flying experience of other particles

Concept

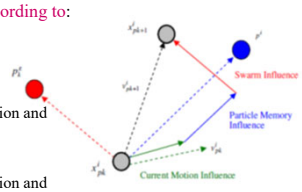
Introduction to the PSO: **Concept**

- Collection of flying particles (swarm) - Changing solutions
- Search area - Possible solutions
- Movement towards a *promising area* to get the global optimum
- Each particle keeps track:
  - its best solution, personal best, *pbest*
  - the best value of any particle, global best, *gbest*

Concept

Introduction to the PSO: **Concept**

- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues
- Each particle modifies its position according to:
  - its current position
  - its current velocity
  - the distance between its current position and *pbest*
  - the distance between its current position and *gbest*



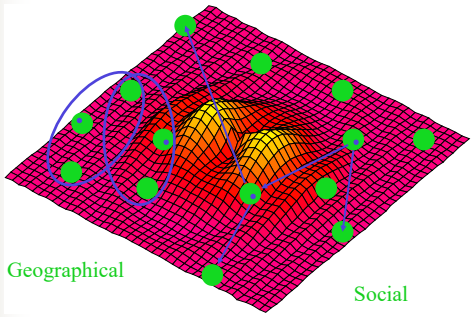
Concept

Introduction to the PSO: **Concept**

All the particles share information, and the next step is to fly to the weighted center of their own optimal position and the global or neighborhood optimal position

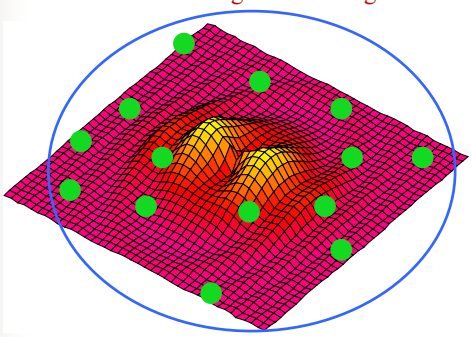
Neighborhood

Introduction to the PSO: Algorithm - Neighborhood



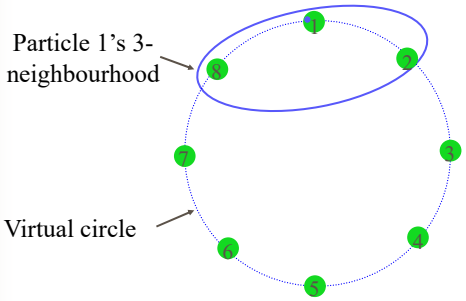
Neighborhood

Introduction to the PSO: Algorithm - Neighborhood



Neighborhood

The circular neighbourhood



Parameters

Introduction to the PSO: Algorithm - Parameters

- Algorithm parameters
  - A : Population of agents
  - p_i : Position of agent a_i in the solution space
  - f : Objective function
  - v_i : Velocity of agent's a_i
  - $V(a_i)$: Neighborhood of agent a_i (fixed)
- The neighborhood concept in PSO is not the same as the one used in other meta-heuristics search, since in PSO each particle's neighborhood never changes (is fixed)

Algorithm

Introduction to the PSO: Algorithm

```
[x*] = PSO()
P = Particle_Initialization();
For i=1 to it_max
  For each particle p in P do
    fp = f(p);
    If fp is better than f(pBest)
      pBest = p;
    end
  end
  gBest = best p in P;
  For each particle p in P do
    v = v + c1*rand*(pBest – p) + c2*rand*(gBest – p);
    p = p + v;
  end
end
```

Algorithm

Introduction to the PSO: Algorithm

- **Particle update rule**

$$p = p + v$$

with

$$v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)$$

where

- p: particle's position
- v: path direction
- c_1: weight of local information
- c_2: weight of global information
- *pBest*: best position of the particle
- *gBest*: best position of the swarm
- *rand*: random variable

Parameters

Introduction to the PSO: Algorithm - Parameters

- Number of particles usually between 10 and 50
- Usually $c_1 + c_2 = 4$ (empirically chosen value)
- If velocity is too low → algorithm too slow
- If velocity is too high → algorithm too unstable

Algorithm

Introduction to the PSO: Algorithm

- Create a 'population' of agents (particles) uniformly distributed over X
- Evaluate each particle's position according to the objective function
- If a particle's current position is better than its previous best position, update it
- Determine the best particle (according to the particle's previous best positions)

Algorithm

Introduction to the PSO: Algorithm

- Update particles' velocities:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{inertia} + \underbrace{\mathbf{c}_1\mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{personal\ influence} + \underbrace{\mathbf{c}_2\mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{social\ influence}$$

- Move particles to their new positions:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}$$

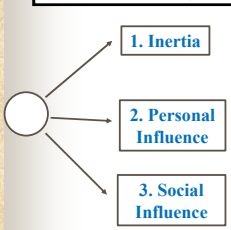
- Go to step 2 until stopping criteria are satisfied

Algorithm

Introduction to the PSO: Algorithm

Particle's velocity:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{inertia} + \underbrace{\mathbf{c}_1\mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{personal\ influence} + \underbrace{\mathbf{c}_2\mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{social\ influence}$$



- Makes the particle move in the same direction and with the same velocity

- Improves the individual
- Makes the particle return to a previous position, better than the current

- Makes the particle follow the best neighbors direction

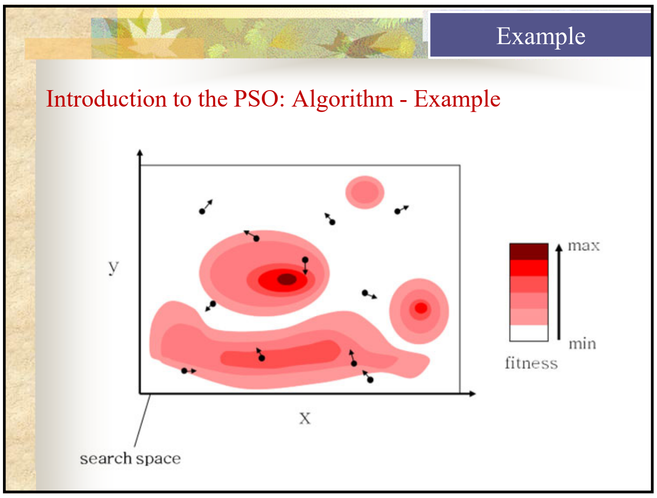
Algorithm

Introduction to the PSO: Algorithm

- *Intensification*: explores the previous solutions, finds the best solution of a given region
- *Diversification*: searches new solutions, finds the regions with potentially the best solutions

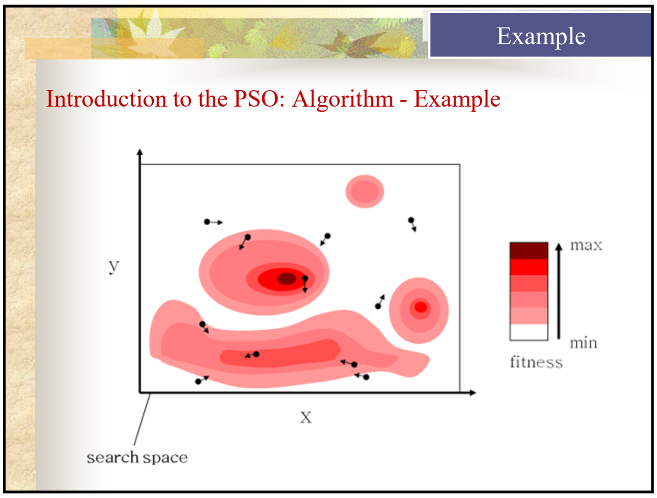
In PSO:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{Diversification}} + \underbrace{\mathbf{c}_1\mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t) + \mathbf{c}_2\mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{Intensification}}$$



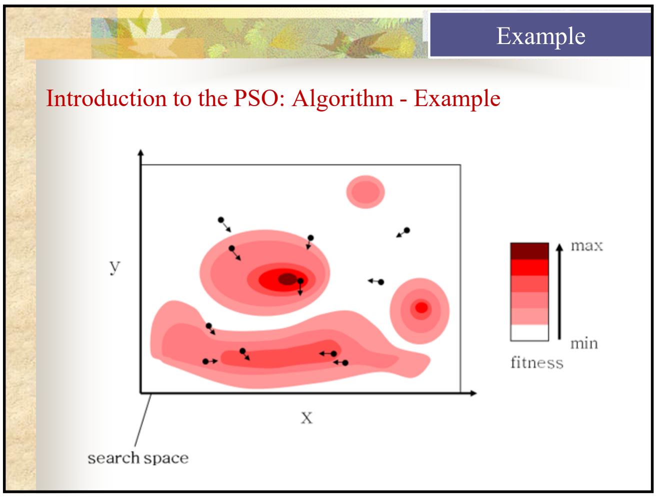
Example

Introduction to the PSO: Algorithm - Example



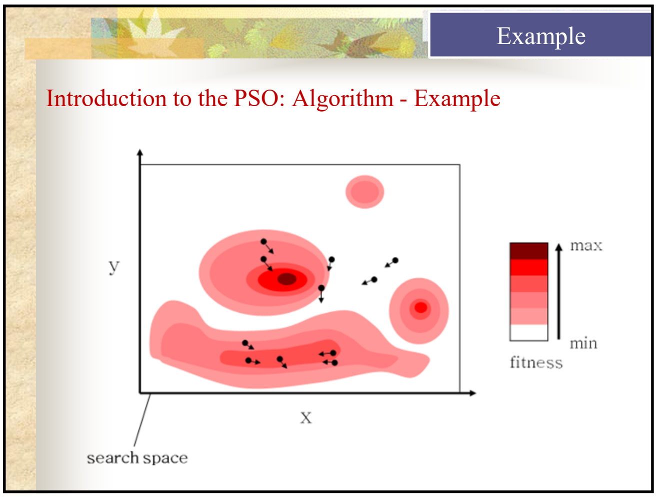
Example

Introduction to the PSO: Algorithm - Example



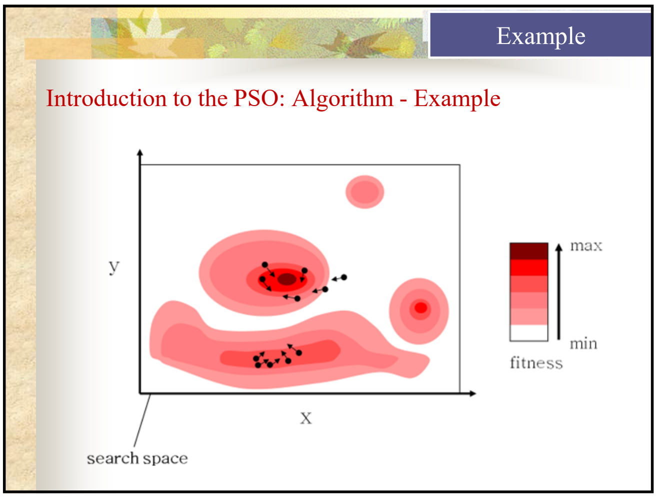
Example

Introduction to the PSO: Algorithm - Example



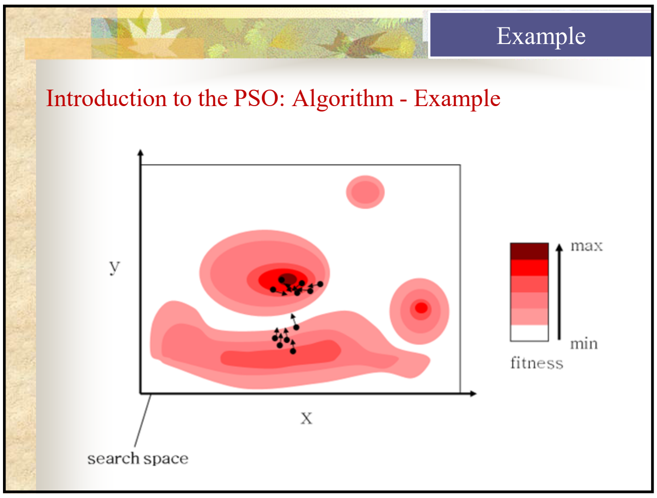
Example

Introduction to the PSO: Algorithm - Example



Example

Introduction to the PSO: Algorithm - Example

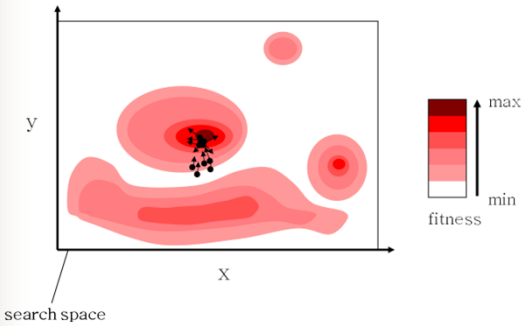


Example

Introduction to the PSO: Algorithm - Example

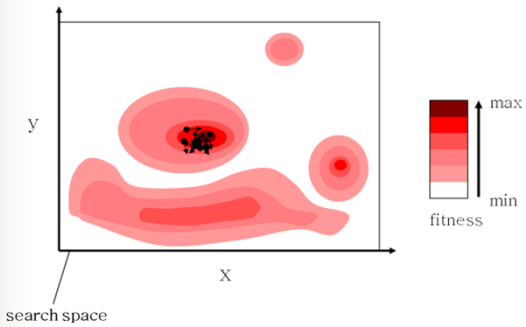
Example

Introduction to the PSO: Algorithm - Example



Example

Introduction to the PSO: Algorithm - Example



Characteristics

Introduction to the PSO: Algorithm Characteristics

- Advantages
  - Insensitive to scaling of design variables
  - Easily parallelized for concurrent processing
  - Very few algorithm parameters
  - Very efficient global search algorithm
- Disadvantages
  - Tendency to a fast and premature convergence in mid optimum points
  - Slow convergence in refined search stage (weak local search ability)