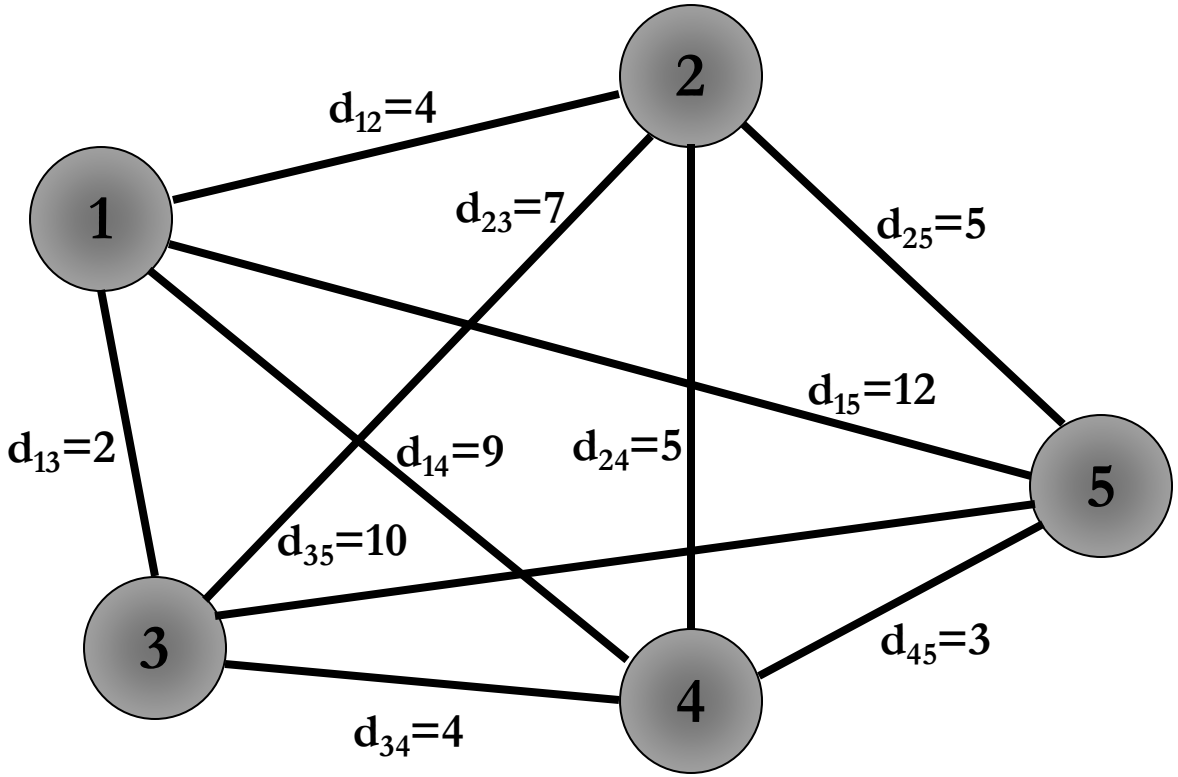
# ACO Applications

## Travelling Salesman Problem

# ACO FOR TSP - EXAMPLE



# ACO FOR TSP - EXAMPLE

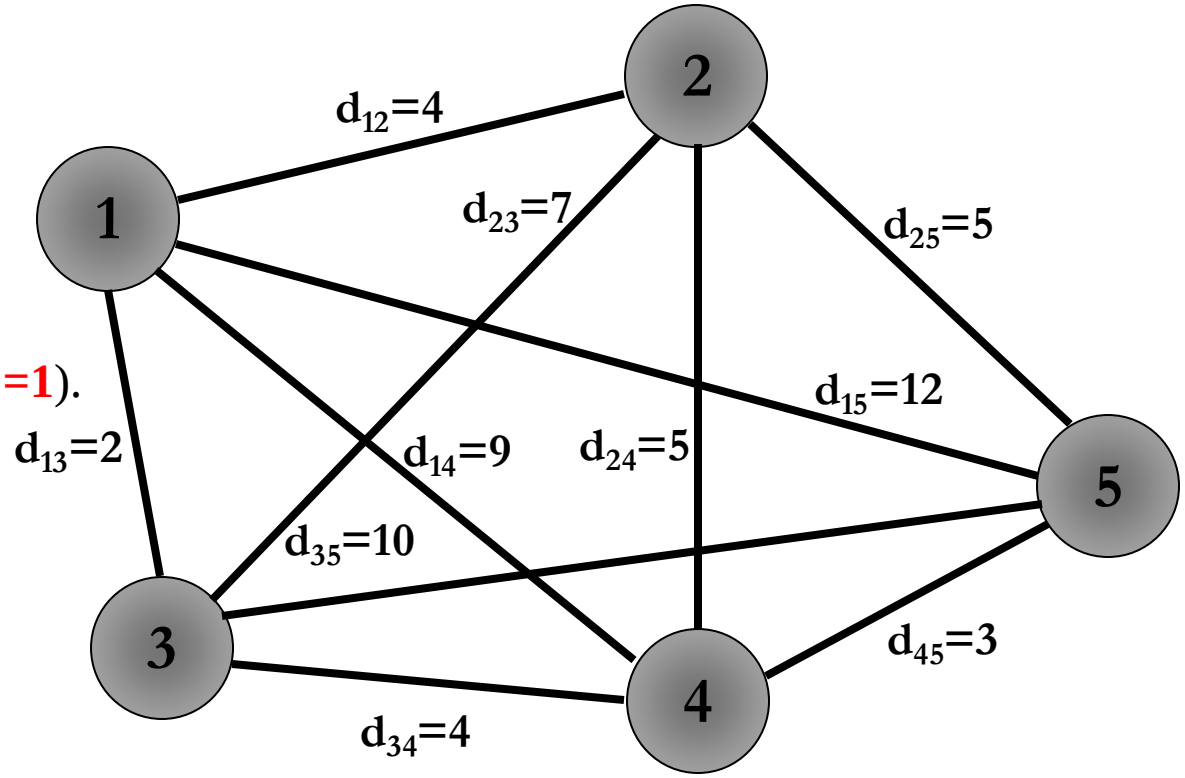
- A small artificial pheromone value is placed on all the edges,
- ***M*** ants are placed on the graph divided among all the nodes,
- ***M*** (number of ants) roundtrips are created in ***n*** (number of cities) steps.

# ACO FOR TSP - EXAMPLE

- A simple iteration:
  - Assume that an ant ***k*** was placed at node 1,
  - The neighbours of node 1 are :

$$N_1 = \{2, 3, 4, 5\}$$

  - Initially, all the edges have the same pheromone value (assume $\tau = 1$).



# ACO FOR TSP - EXAMPLE

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} / d_{ij}^{\beta}}{\sum_{n \in N_i} \tau_{in}^{\alpha} / d_{in}^{\beta}} & if \quad j \in N_i \\ 0 & if \quad j \notin N_i \end{cases}$$

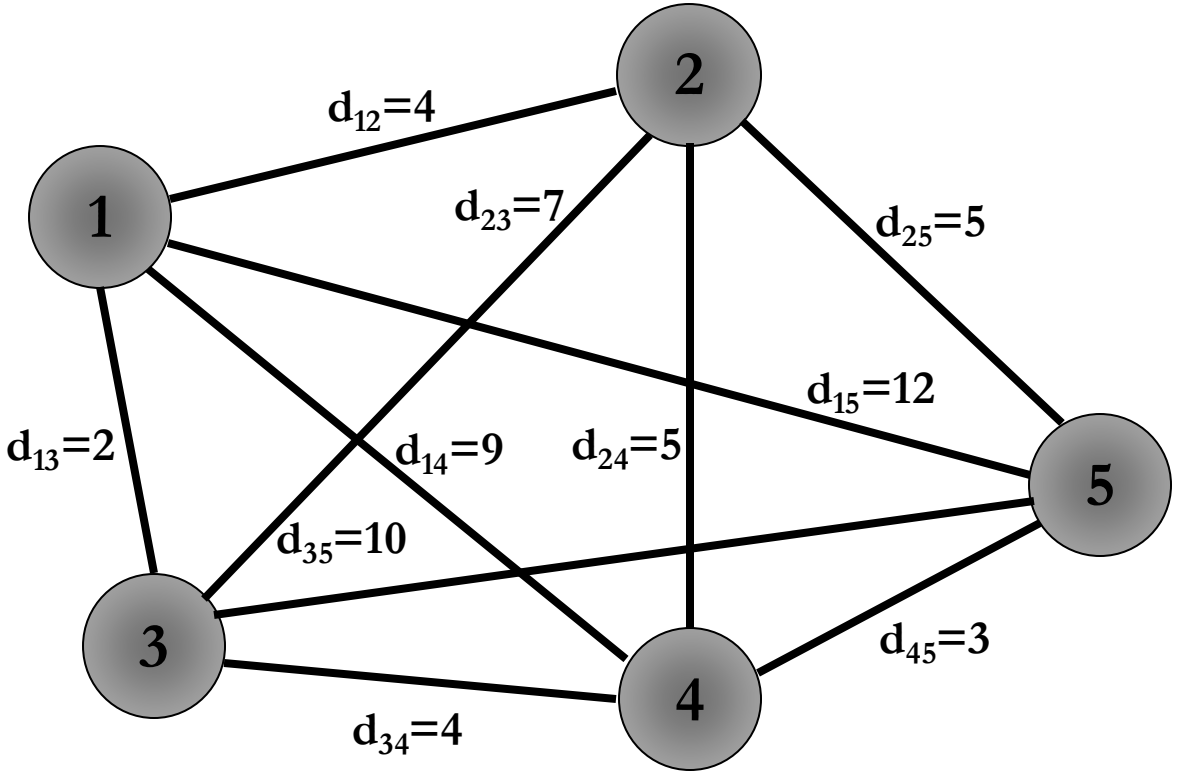
- Assuming $\alpha$ and $\beta$ are both equal to 1,

- Then:

$$\sum_{n \in N_1} \tau_{1n}^{\alpha} / d_{1n}^{\beta} = \frac{1}{4} + \frac{1}{2} + \frac{1}{9} + \frac{1}{12} \cong 0.95$$

$$p_{12}^{k} = \frac{0.25}{0.95} \cong 0.26, p_{13}^{k} = \frac{0.5}{0.95} \cong 0.53$$

$$p_{14}^{k} = \frac{0.11}{0.95} \cong 0.12, p_{15}^{k} = \frac{0.08}{0.95} \cong 0.08$$



# ACO FOR TSP - EXAMPLE

- Node 3 has the highest probability of being selected,

- If node 3 gets selected, ant ***k*** moves to that node and continues with the next iteration where:

$$N_3 = \{2, 4, 5\}$$

Node 3 becomes Tabu choice

# ACO FOR TSP - EXAMPLE

•Hence:

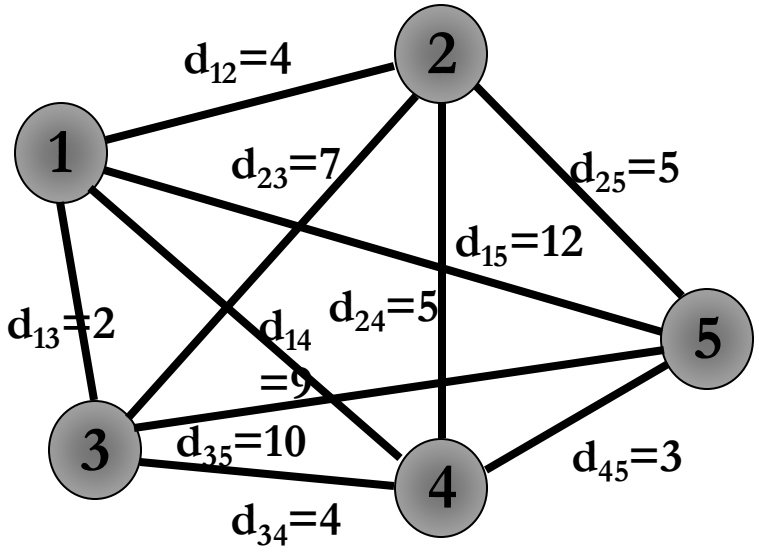
$$\sum_{n \in N_3} \tau_{1n}^{\alpha} / d_{1n}^{\beta} = \frac{1}{7} + \frac{1}{4} + \frac{1}{10} \cong 0.49$$

$$p_{32}^{k} = \frac{0.14}{0.49} \cong 0.29, p_{34}^{k} = \frac{0.25}{0.49} \cong 0.51$$

$$p_{35}^{k} = \frac{0.1}{0.49} \cong 0.20$$

•Another node is selected, and so on …

# ACO FOR TSP - EXAMPLE




•If ant *k* completes the tour:

$$Tour = \{1, 3, 2, 4, 5\}$$

$$Cost = 29$$

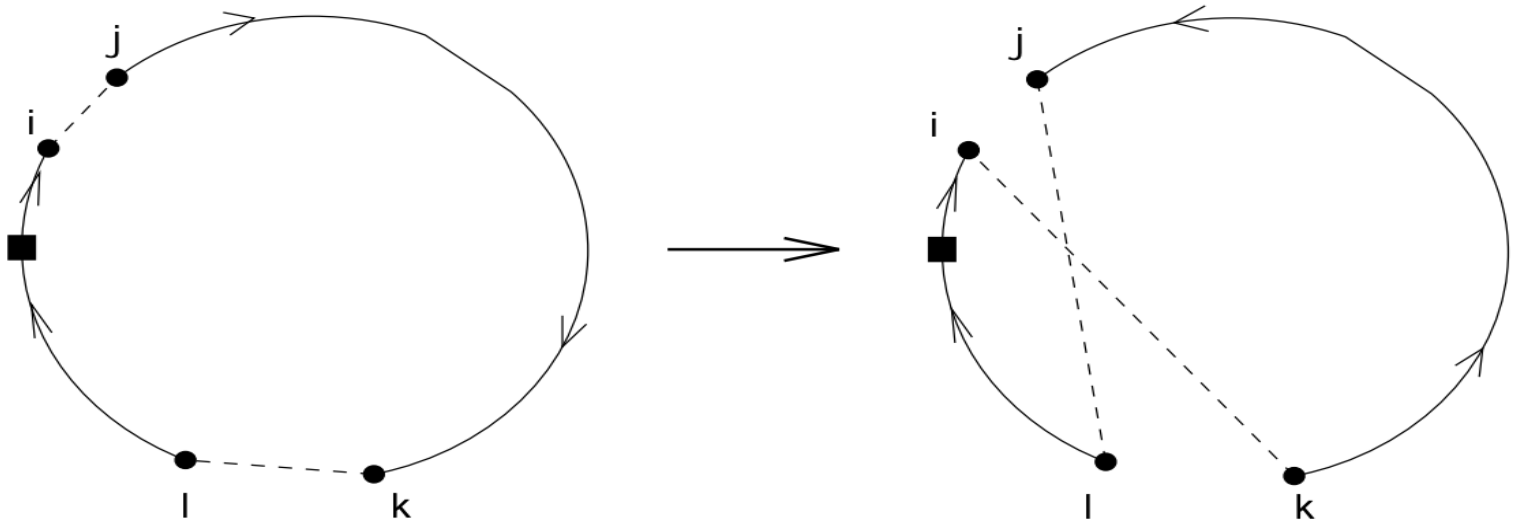
•First, all the pheromone trails get evaporated,

•And then, if this ant is selected to update the pheromone trails, it enforces the edges

$$\{1, 3\}, \{3, 2\}, \{2, 4\},$$
$$\{4, 5\}, \{5, 1\}$$

with the value Q/29.

# LOCAL SEARCH OPTION




- *Two edges* $(i, j)$ and $(k, l)$ are *selected, removed, and replaced* by other two edges $(i, k)$ and $(j, l$
  (or, $(k, i)$ and $(l, j)$) $\rightarrow$ one of the two sub-paths always gets *reverted!*
- *Gain* in terms of tour length: $(i, k) + (j, l) - (i, j) - (k, l)$
- The exchange with the *best gain* is selected and executed
- For each edge $(i, j)$ all the other $n - 1$ edges must be gain-checked

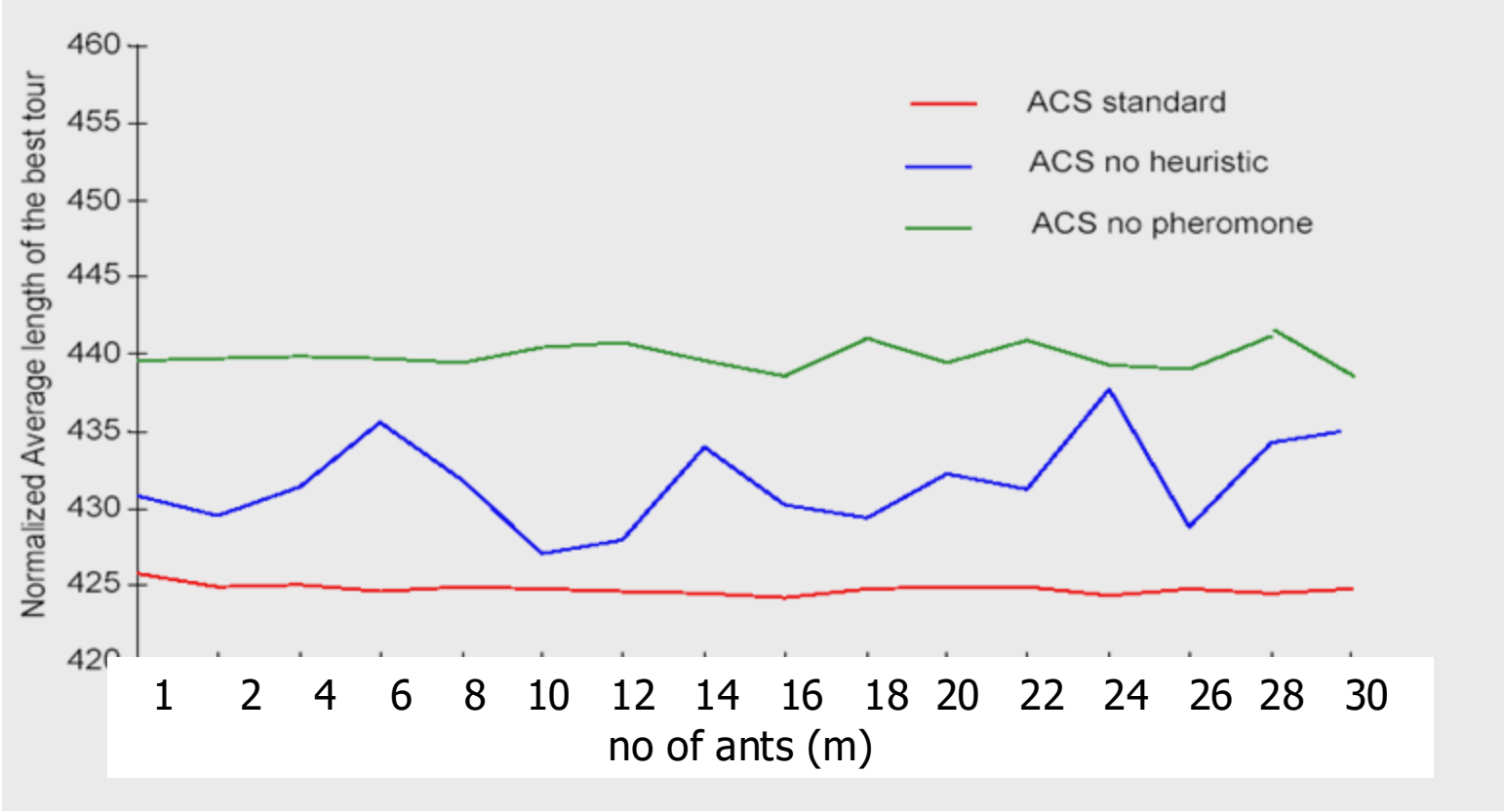
$$n(n-1) = O(n^2)$$ possible moves in the 2-exchange neighborhood

- $d(s, s') = 4$ in an undirected graph, the 2-exchange neighborhood consists in fact of applying *four* move *operations:* *delete* $(i, j)$, *insert* $(i, k)$, *delete* $(k, l)$, *insert* $(j, l)$

# PHEROMONE AND THE HEURISTIC FUNCTION

The importance of the pheromone and the heuristic function [8]

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




Comparison between ACS standard, ACS with no heuristic, and ACS in which ants neither sense nor deposit pheromone. Problem: Oliver30. Averaged over 30 trials, 10,000/*m* iterations per trial.

M. Dorigo and L. M. Gambardella."Ant Colony System: a cooperative learning approach to the traveling salesman problem". IEEE Trans. Evolutionary Computation, vol. 1, no. 1, 1997.

# PHEROMONE AND THE HEURISTIC FUNCTION

The results show that not using pheromone deteriorates the performance,

ACS without heuristics performs better  than ACS without pheromone. This may be due that ACS with pheromone and no heuristics is still guided by the global update rule (reflecting the importance of the solution).

ACS without pheromone reduces to a stochastic multi-greedy algorithm

ACS with both is better confirming the role of cooperation.

M. Dorigo and L. M. Gambardella.”Ant Colony System: a cooperative learning approach to the traveling salesman problem”. IEEE Trans. Evolutionary Computation, vol. 1, no. 1, 1997.

# COMPARISON RESULTS OF ACO SYSTEMS

Number of ants = n

Number of iterations = 10000

| Benchmark | Optimal | AS | MMAS | ACS |
|---|---|---|---|---|
| eil51 | **426** | 437.3 | **427.6** | 428.1 |
| kroa100 | **21282** | 22471.4 | **21320.3** | 21420.0 |
| d198 | **15780** | 16702.1 | **15972.5** | 16054.0 |

T. Stutzle and H. H. Hoos."*MAX-MIN* Ant System". *Future Generation Comput. Syst.*, vol. 16, no 8, 2000.

# COMPARISON RESULTS: ACO AND GENETIC ALGORITHMS

Comparison of ACS with the genetic algorithm (GA), evolutionary programming (EP), simulated annealing (SA), and the annealing-genetic algorithm (AG), a combination of genetic algorithm and simulated annealing.

The best integer tour length, the best real tour length (in parentheses) and the number of tours required to find the best integer tour length (in square brackets).

The best result for each problem is in boldface.

M. Dorigo and L. M. Gambardella."Ant Colonies for the travelling salesman problem", BioSystems, 1997.

# COMPARISON RESULTS: ACO AND GENETIC ALGORITHMS

| Problem name | ACS | GA | EP | SA | AG | Optimum |
|---|---|---|---|---|---|---|
| Oliver30 | **420** | 421 | **420** | 424 | **420** | **420** |
| (30-city problem) | **(423.74)** | (N/A) | **(423.74)** | (N/A) | (N/A) | **(423.74)** |
| | **[830]** | [3,200] | [40,000] | [24,617] | [12,620] | |
| Eil50 | **425** | 428 | 426 | 443 | 436 | **425** |
| (50-city problem) | (427.96) | (N/A) | **(427.86)** | (N/A) | (N/A) | (N/A) |
| | **[1,830]** | [25,000] | [100,000] | [68,512] | [28,111] | |
| Eil75 | **535** | 545 | **542** | 580 | 561 | **535** |
| (75-city problem) | **(542.31)** | (N/A) | (549.18) | (N/A) | (N/A) | (N/A) |
| | **[3,480]** | [80,000] | [325,000] | [173,250] | [95,506] | |
| KroA100 | **21,282** | 21,761 | N/A | N/A | N/A | **21,282** |
| (100-city problem) | **(21,285.44)** | (N/A) | (N/A) | (N/A) | (N/A) | (N/A) |
| | **[4,820]** | [103,000] | [N/A] | [N/A] | [N/A] | |

It is clear that ACS and EP greatly outperform GA, SA, and AG.
N/A means not available in the literature

M. Dorigo and L. M. Gambardella."Ant Colonies for the travelling salesman problem", BioSystems, 1997.

# ACS ON LARGER TSP PROBLEMS

ACS performance for some bigger geometric problems (over 15 trials). We report the integer length of the shortest tour found, the number of tours required to find it, the average integer length, the standard deviation, the optimal solution (for fl1577 we give, in square brackets, the known lower and upper bounds, given that the optimal solution is not known), and the relative error of ACS.

| Problem name | ACS best integer length (1) | ACS number of tours generated to best | ACS average integer length | Standard deviation | Optimum (2) | Relative error $\frac{(1)-(2)}{(2)} * 100$ |
|---|---|---|---|---|---|---|
| d198 (198-city problem) | 15,888 | 585,000 | 16,054 | 71 | 15,780 | 0.68 % |
| pcb442 (442-city problem) | 51,268 | 595,000 | 51,690 | 188 | 50,779 | 0.96 % |
| att532 (532-city problem) | 28,147 | 830,658 | 28,523 | 275 | 27,686 | 1.67 % |
| rat783 (783-city problem) | 9,015 | 991,276 | 9,066 | 28 | 8,806 | 2.37 % |
| fl1577 (1577-city problem) | 22,977 | 942,000 | 23,163 | 116 | [22,204 – 22,249] | 3.27÷3.48 % |

# ACO Applications

## Cell Assignment in PCS Networks

# CELL ASSIGNMENT IN PCS NETWORKS (CA)

The cell assignment problem is a challenging problem in PCS (Personal Communication Services) networks,

In PCS networks, each cell has an antenna that is used to communicate with subscribers over some pre-assigned radio frequencies,

Groups of cells are connected to a switch, through which the cells are then routed to the satellite networks.

S. J. Shyu, B. M. T. Lin and T. S. Hsiao.“An Ant Algorithm for Cell Assignment in PCS Netwroks”. IEEE International Conference on Networking, Sensing and Control, pp. 1081-1086, 2004.
J. R. L. Fournier and S. Pierre.“Assigning Cells to Switches in Mobile Networks using an Ant Colony Optimization Heuristic”.Computer Communications, vol. 28, pp. 65-73, 2005.

# CELL ASSIGNMENT IN PCS NETWORKS (CA)

Assignment of channels to cells

- Assign frequency channels to cells so as to minimize interference (close cells shouldn't have close frequency ranges) and maximize utilization of channels (reuse)

Given a set of n cells to be assigned to m switches, each switch k has a capacity $M_k$. The objective is to have an assignment that minimizes the cost (link between cells and switches and handoff cost)

# ACO FOR CA

In this problem, each ant has to make two different decisions per iteration:

- Choosing the next cell to be assigned,
- Choosing the switch that the chosen cell to be assigned to.
  - A pheromone trail $\tau_{ij}$ is associated with every cell $i$ and switch $j$ possible assignment.

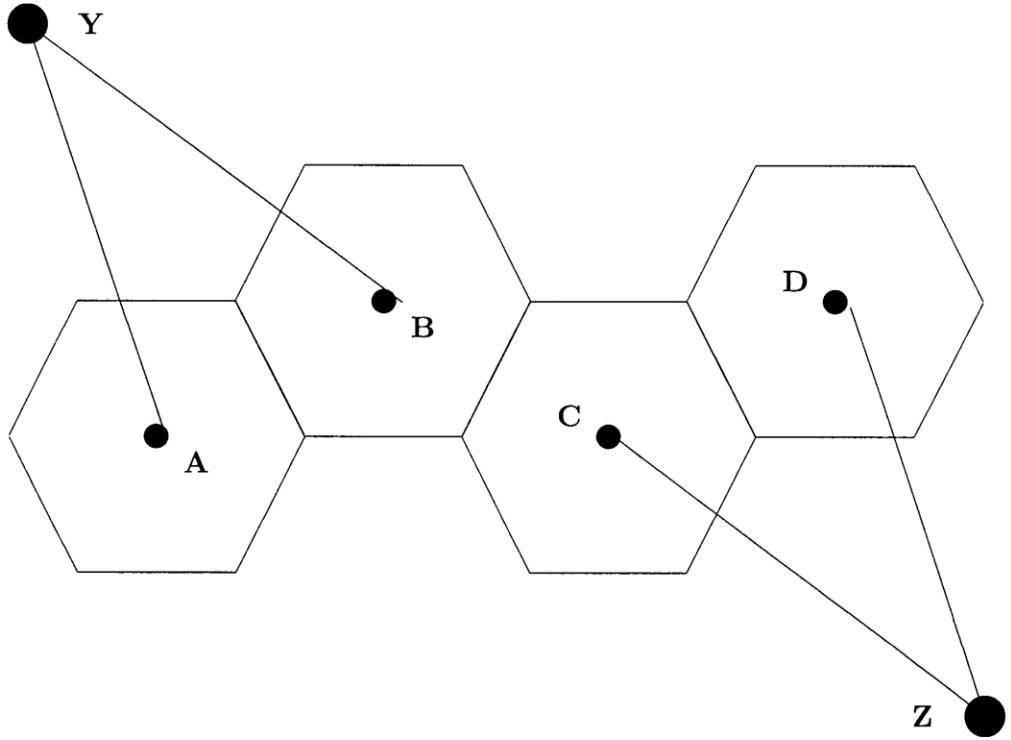
Select max number of iterations

Select number of ants

Decide on transition rule

Decide on pheromone update rules

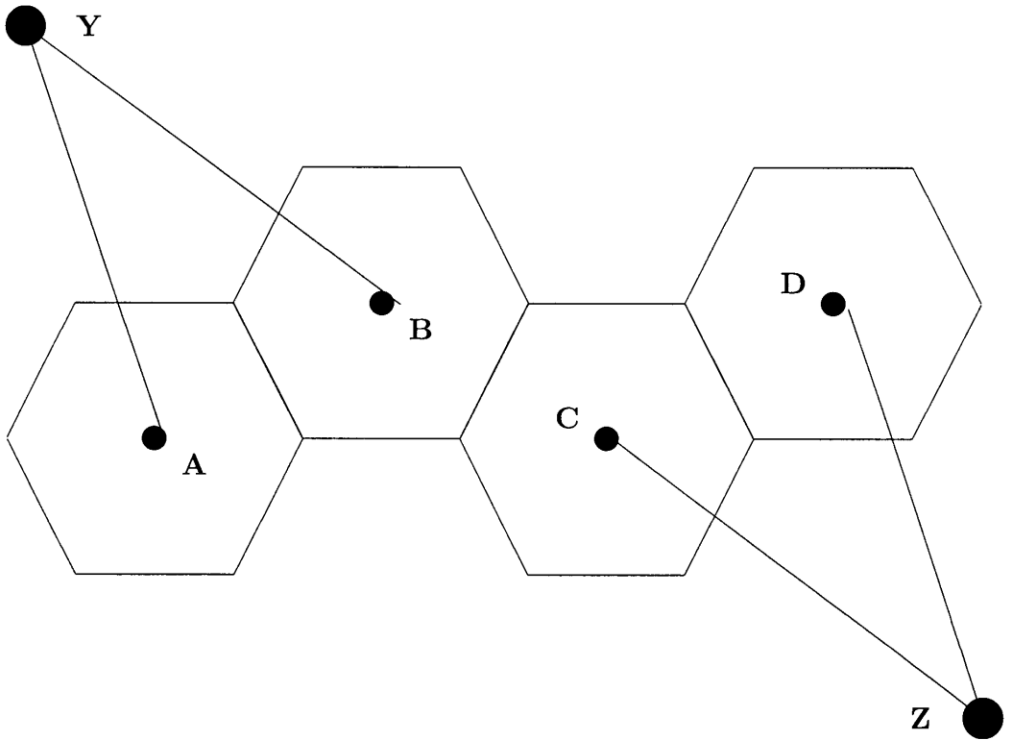
# CELL ASSIGNMENT IN PCS NETWORKS (CA)




For instance:

- Assuming a situation where cells **A** and **B** are assigned to switch **Y** and cells **C** and **D** are assigned to switch **Z**,
- Suppose that a subscriber is currently talking to someone and this call is transmitted through cell **B** and switch **Y**,

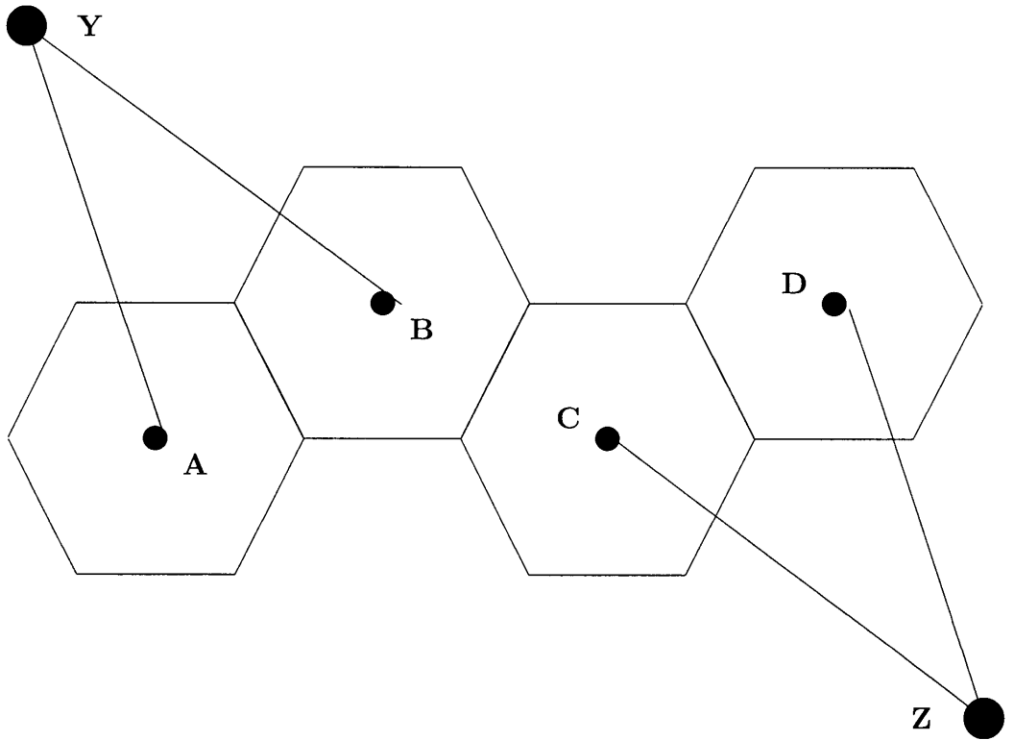
# CELL ASSIGNMENT IN PCS NETWORKS (CA)




If the subscriber moves from cell **B** to cell **A,** switch **Y** will perform a *handoff* for the call:

- This call does not trigger any location update in the database that records the position of the subscriber,
- The handoff does not entail any network entity other than switch **Y**.

# CELL ASSIGNMENT IN PCS NETWORKS (CA)




On the other hand, if the subscriber moves from cell **B** to cell **C**, then the *handoff* involves:

- The modification of the location of the subscriber in the database,
- The execution of a fairly complicated protocol between switches **Y** and **Z**.

# CELL ASSIGNMENT IN PCS NETWORKS (CA)

Costs associated with the assignment include:

- Cable (link) costs between cells and switches,
- Handoff costs between different cells:
  - Simple or no costs (involving only one switch),
  - Complex costs (involving two switches).

Each switch has a certain capacity (in terms of calls volume) that should not be exceeded.

The objective is to have an assignment that minimizes the cost.

NP-hard problem

# MATHEMATICAL FORMULATION

Let $c_{ik}$ be the cost of the link between cell i and switch $k$, $i=1,\ldots,n$ ; $k=1,\ldots,m$

Let $x_{ik}$ is 1 if cell i assigned to switch k, 0 otherwise;

Let $y_{ij}$ = 1 if cell i and cell j are assigned to the same switch.

Let $M_k$ be the capacity of switch k

Let $h_{ij}$ is handoff cost between cell i and j when they are assigned to different switches;

Let $d_i$ is the number of calls allowed for cell i

# MATHEMATICAL FORMULATION

- The objective is

$$\text{Minimize } f = \sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n}\sum_{j=1,i\neq j}^{n} h_{ij}(1-y_{ij})$$

$$\text{subject to } \sum_{i=1}^{n} d_i x_{ik} \leq M_k, k = 1,\ldots,m$$

# ACO-ALGORITHM

Select max no of iterations, itmax

Select number of Ants, antmax

While number of iterations <itmax

- While current ant number< antmax
  - Initialize ant parameters (pheromone, update,..etc)
  - While (number of assigned cells is less than n)
    - Select next cell to be assigned using transition rule.
    - check capacity constraint of the selected switch,
    - Update problem data (capacity, cost,..)
  - End
  - Update Pheromone trail using update rules and evaporation rule
  - Evaluate quality of the Solution using the objective function
- Retain best solution of all ants
- Update pheromone trail based on best solution (if desired)

Return best solution

# ACO FOR CA

- Transition rules

- Let T be the set of switches not assigned yet

- Simple ratio

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}}{\sum_{j \in T} \tau_{ij}^{\alpha}} & if \quad j \in T \\ 0 & if \quad j \notin T \end{cases}$$

- Using heuristics

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in T}[\tau_{lk}]^{\alpha}[\eta_{lk}]^{\beta}} & \text{if} \quad j \in T \\ 0 & \text{otherwise} \end{cases}$$

# ACO FOR CA

- Heuristic can be related to utilization

$$\eta_{ij} = (1/\text{cost of assigning cell i to switch j})$$

- Pheromone update and evaporation can be simple

$$\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij}$$

- Or

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}$$

- Where $\Delta\tau_{ij}$ can be density, quantity (using local cost) or online delayed using solution quality

# ACO FOR CA

- Or as in ACS, two updates

$$\tau_{ij} = (1-\rho_2)\tau_{ij} + \rho_2\tau_0$$

and

$$\tau_{ij} = (1-\rho_1)\tau_{ij} + \rho_1\Delta\tau_{ij}^{best}$$

# REFERENCES

**Text:** Fundamentals of Computational Swarm Intelligence by A. P. Engelbrecht, Wiely, 2005. Chapter 1, 22, and 23.

**Papers:**

ID. Couzin, J. Krause, R. James, GD. Ruxton, NR. Franks.”Collective Memory and Spatial Sorting in Animal Groups”. Journal of Theoretical Biology, 218, pp. 1–11, 2002.

C. Grosan, A. Abraham and C. Monica.”Swarm Intelligence in Data Mining”. In Swarm Intelligence in Data Mining, A. Abraham, C. Grosan and V. Ramos (Eds), Springer, pp. 1–16, 2006.

MM. Millonas .“Swarms, phase transitions, and collective intelligence”. In CG. Langton Ed., Artificial Life III, Addison Wesley, Reading, MA, 1994.

# REFERENCES


P. P. Grasse.“Recherches sur la biologie des termites champignonnistes (*Macrotermitina*)”. Ann. Sc. Nat., Zool. Biol. anim., 6, 97, 1944.

P. P. Grasse.”La reconstruction du nid et les coordinations interindividu- elles chez *bellicositermes natalensis* et *cubitermes* sp. La theorie de la stig- mergie: essai d'interpretation du comportement des termites constructeurs”. Insectes Sociaux, 6, 41, 1959.

J. L. Deneubourg, S. Aron, S. Goss and J. M. Pasteels.”The self- organizing exploratory pattern of the Argentine ant”. Journal of Insect Behaviour, 3, 159, 1990.

S. Goss, S. Aron, J. L. Deneubourg and J. M. Pasteels.”Self-organized shortcuts in the Argentine ant”. Naturwissenschaften, 76, 579, 1989.

M. Dorigo.”Optimization, Learning and Natural Algorithms”. Ph.D. thesis, DEI, Politecnico di Milano, Italy, pp. 140, 1992.

# REFERENCES

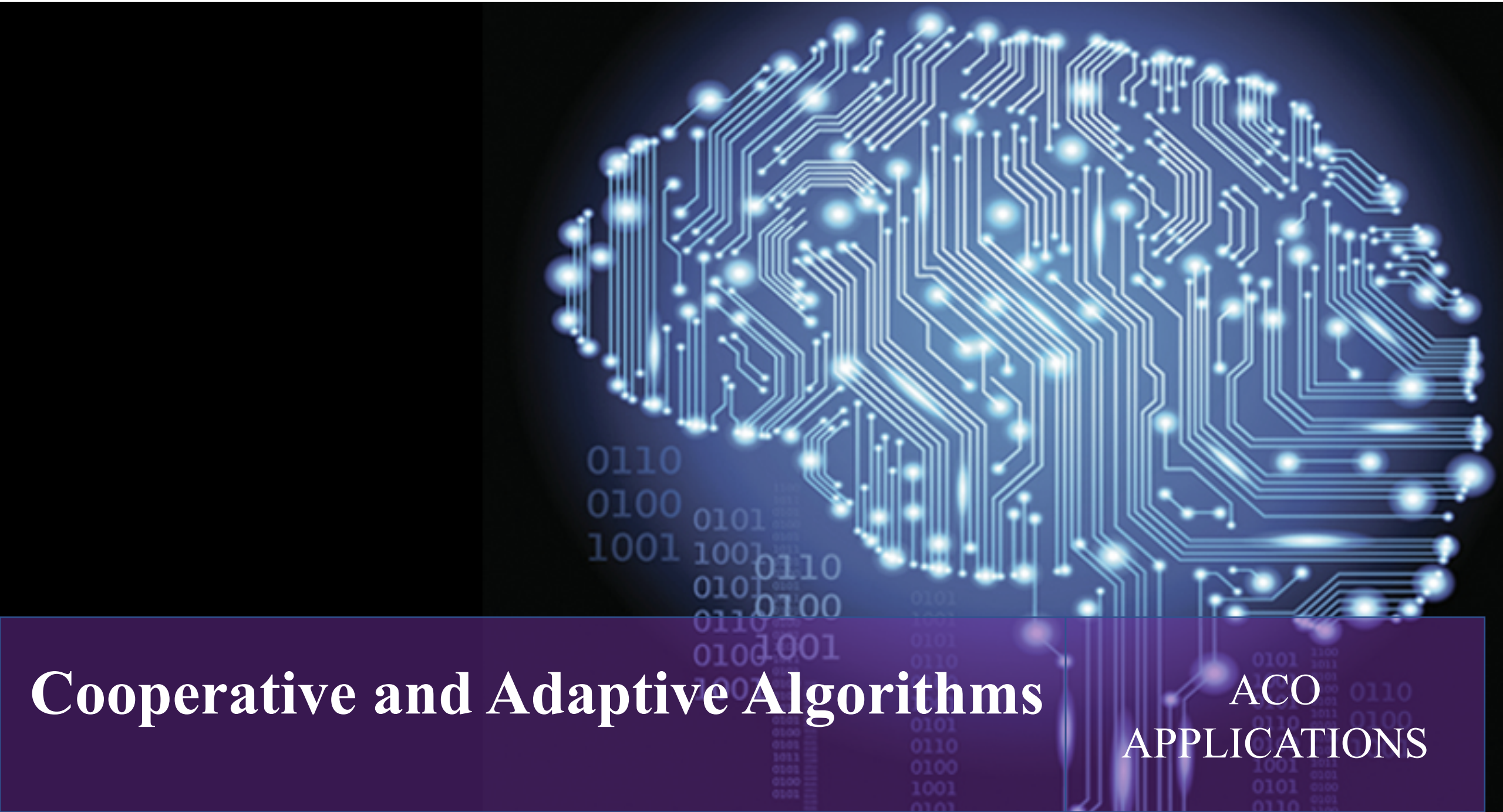

M. Dorigo, V. Maniezzo and A. Colorni.” Ant System: Optimization by a Colony of Cooperating Agents”. IEEE Trans. Syst., Man, and Cybern. Part B, vol. 26, no. 1, 1996.

T. Stutzle and H. H. Hoos.”*MAX-MIN* Ant System”. *Future Generation Comput. Syst.*, vol. 16, no 8, 2000.

M. Dorigo and L. M. Gambardella.”Ant Colony System: a cooperative learning approach to the traveling salesman problem”. IEEE Trans. Evolutionary Computation, vol. 1, no. 1, 1997.

J. Bautista and J. Pereira. “Ant Algorithms for A Time and Space Constrained Assembly Line Balancing Problem”. European Journal of Operation Research, 177, pp. 2016-2032, 2002.

S. J. Shyu, B. M. T. Lin and T. S. Hsiao.“An Ant Algorithm for Cell Assignment in PCS Netwroks”. IEEE International Conference on Networking, Sensing and Control, pp. 1081-1086, 2004.



# Cooperative and Adaptive Algorithms

ACO
APPLICATIONS

# ACO Applications

## Assembly Line Balancing

# ALB

- An assembly line is made of a **m** workstations arranged **in series and in parallel**.
- The production program consists of a **set V of n tasks with precedence relations** that need to be assigned to the workstations.
- Each task **j** requires processing time $t_j > 0$ and space $a_j$
- Each station **k** has a cycle time $c_k$, to carryout a subset of **V**, $S_k$ tasks assigned to it (workload) and space area $A_k$.
- Typically, the **cycle time is the same** for all workstations, **c**, which determines the production rate $r = 1/c$. The cycle time represents an upper limit of the total time required by the tasks assigned to the workstation.
- $I_k = c - t(S_k)$ is the idle time of workstation **k**. Total idle time is sum of all idle times over all stations

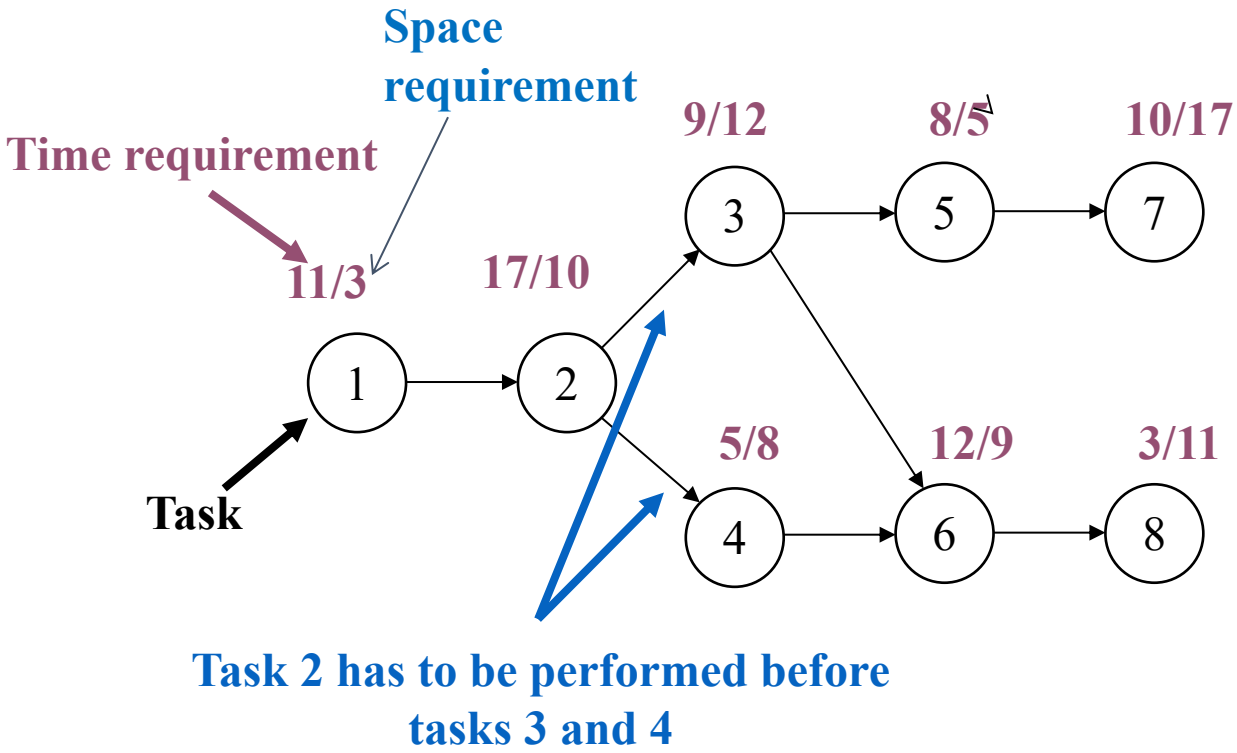
# ALB

- The main objective is to assign the tasks to workstations to satisfy one or more of the following:
  1. Minimize the number of workstations given a fixed value for the cycle time c and space A
  2. Minimize the cycle time c given a fixed number of workstations m and A
  3. Minimize total space required given c and m
  4. Give m, c and A find a feasible solution (assignment)
     - Multi-objectives:
       - 1 and 2 given A
       - 1 and 3 given c
       - 2 and 3 given m
       - 1 and 2 and 3

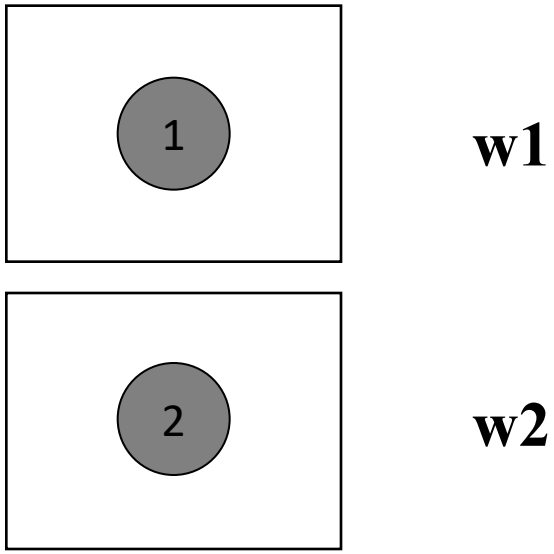
A simplified version of the above is not to consider the space requirement

# ALB

- C=20, A=20


Space requirement
Time requirement
9/12
8/5
10/17
11/3
17/10
3
5
7
1
2
5/8
12/9
3/11
Task
4
6
8
Task 2 has to be performed before tasks 3 and 4

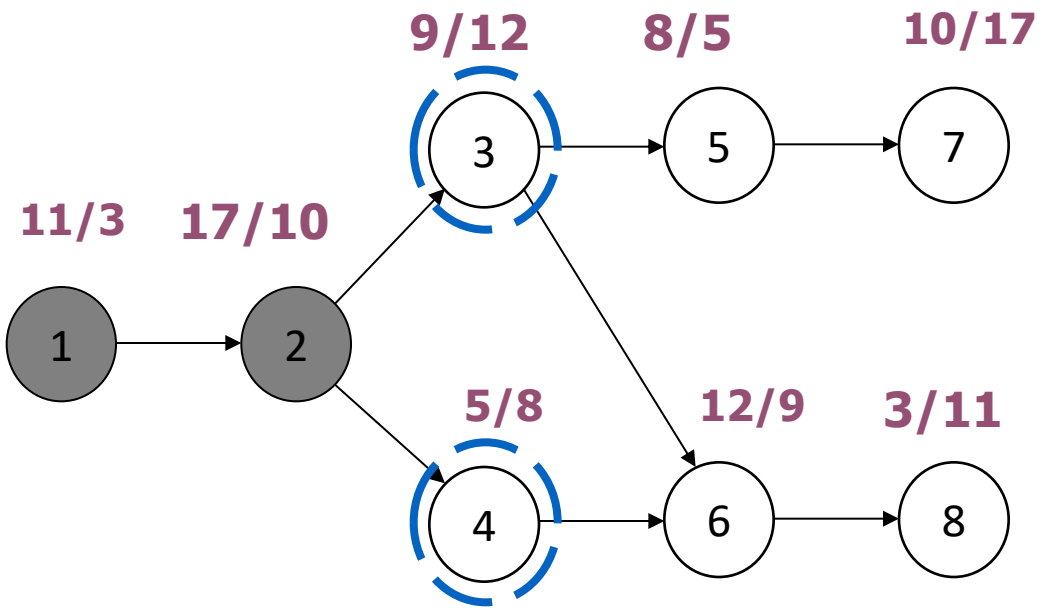
# ALB c=20, A=20




w1

w2

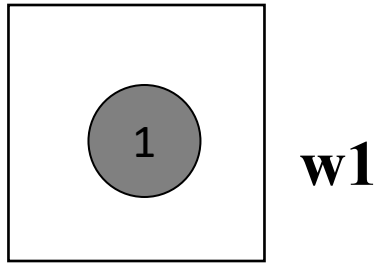
Tasks 1 and 2 get assigned to two workstations



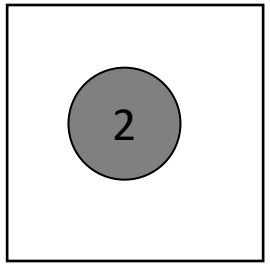

Next tasks to assign

Neither can be assigned to w2 or w1

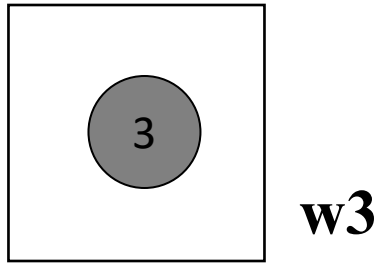
# ALB




w1



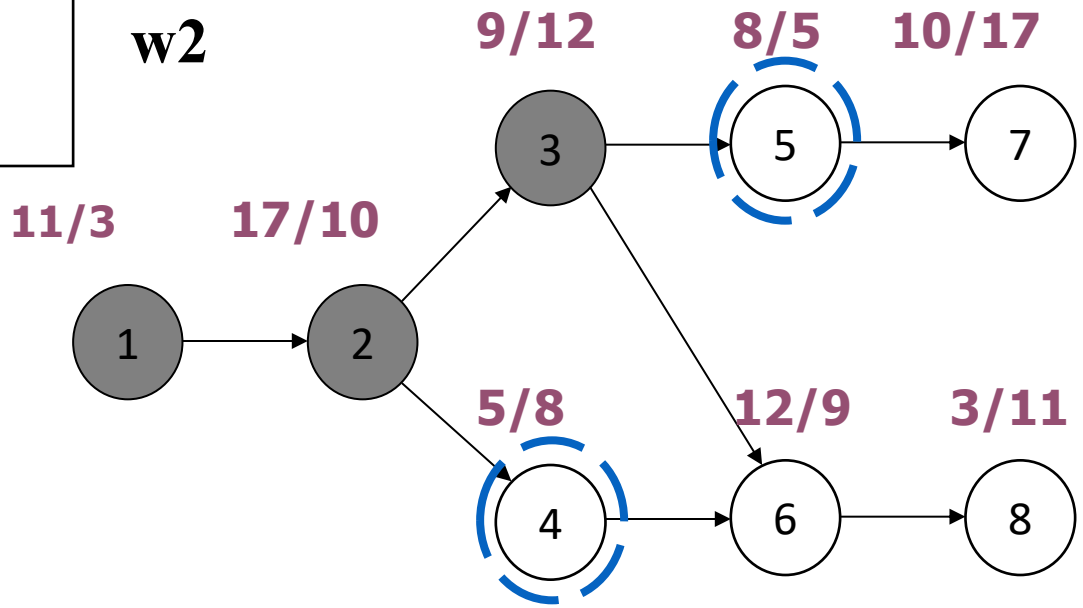

w2




w3

**Assuming task 3 gets assigned to the new workstation w3**

**4 or 5 could be assigned to w3, greedy selects 5**




**Next tasks to assign**

# ALB

- Continuing this way (without checking space constraint) we get the following solution

  w1={1}, w2={2}, w3={3,5}, w4={4,6,8}, w5={7}, m=5 (five machines)

- If we check space w4 doesn't satisfy the space limit.

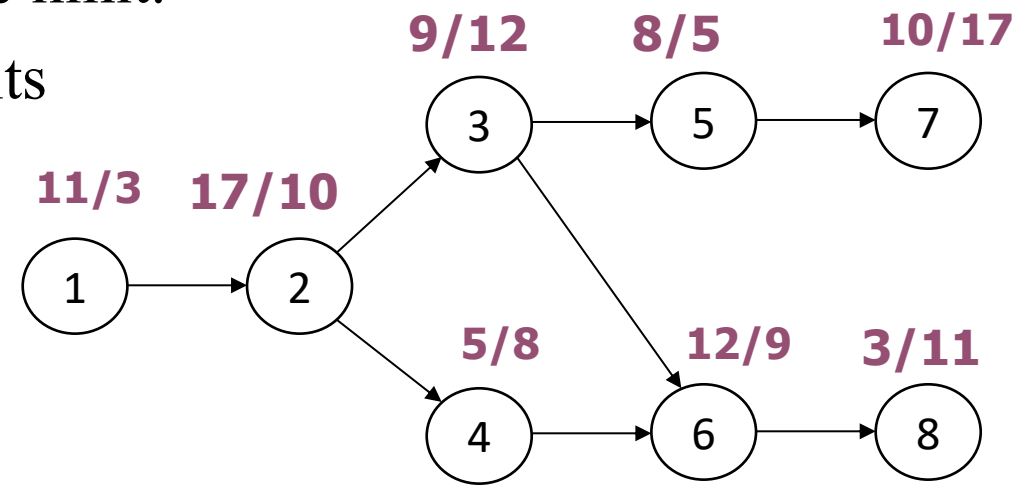
  28 space units, but we are limited by 20 units

- A solution to satisfy space will be

  w1={1}, w2={2}, w3={3,5}, w4={4,6},

  w5={7}, w6={8}

  with m=6 (six machines)



# ALB-ACO

- How to apply ACO to solve the problem:
  - In TSP, each ant determines which city to add next to the solution,
  - In this problem, each ant determines which assignment (task/workstation) to add next to the solution,
  - Each different assignment gets a pheromone trail parameter. A pheromone trail is associated with each assignment,
    - $\tau_{jk}$ denotes the pheromone trail associated with task ***j*** getting assigned to workstation ***k***,
  - Let **T** be the set of tasks that:
    - Are not yet assigned to a work station,
    - Whose predecessors are all assigned to work stations,

# ALB-ACO

- Select max no of iterations, itmax
- Select number of Ants, antmax
- While number of iterations <itmax
  - While current ant number< antmax
    - Initialize ant parameters (pheromone, update,..etc)
    - While (**T** is not empty)
      - Select task j to be assigned to workstation k using transition rule.
      - If none of the tasks can be fit in the current workstation due to time or space capacity) Open a new workstation,
      - Update problem data
    - End (**T** is empty):
    - Update Pheromone trail using update rules and evaporation rule
    - Evaluate Solution
  - Retain best solution of all ants
  - Update pheromone trail based on best solution (if desired)
- Return best solution

# ALB-ACO

- Transition rules
  - Simple ratio

$$p_{jk} = \begin{cases} \dfrac{\tau_{jk}^{\alpha}}{\sum_{n \in T} \tau_{nk}^{\alpha}} & if \quad j \in T \\ 0 & if \quad j \notin T \end{cases}$$

  - Using heuristics

$$p_{jk} = \begin{cases} \dfrac{[\tau_{jk}]^{\alpha}[\eta_j]^{\beta}}{\sum_{l \in T}[\tau_{lk}(t)]^{\alpha}[\eta_j]^{\beta}} & \text{if } j \in T \\ 0 & \text{otherwise} \end{cases}$$

# ALB-ACO

- Heuristic can be related to utilization

$$\eta_j = (1/c)(\sum_{l \in k} t_l + t_j)$$

- Or if space is checked

$$\eta_j = (1/c)(\sum_{l \in k} t_l + t_j) + (1/A)(\sum_{l \in k} a_l + a_j)$$

- Pheromone update and evaporation can be simple

$$\tau_{jk} = (1-\rho)\tau_{jk} + \Delta\tau_{jk}$$

- Where $\Delta\tau_{jk}$ can be density, quantity or online delayed using utilization as solution quality

- An assembly line is made of a **m** workstations arranged **in series and in parallel**.
- The production program consists of a **set V of n tasks with precedence relations** that need to be assigned to the workstations.
- Each task **j** requires processing time $t_j$ >0 and space $a_j$
- Each station **k** has a cycle time $c_k$, to carryout a subset of **V**, $S_k$ tasks assigned to it (workload) and space area $A_k$.
- Typically, the **cycle time is the same** for all workstations, **c**, which determines the production rate **r=1/c**. The cycle time represents an upper limit of the total time required by the tasks assigned to the workstation.
- $I_k = c - t(S_k)$ is the idle time of workstation **k**. Total idle time is sum of all idle times over all stations

# ALB-ACO

- Can use pheromone update based on best solution

$$\tau_{jk} = (1-\rho)\tau_{jk} + \rho\Delta\tau_{jk}^{best}$$

# ACO Applications

## Timetabling Problems

# Timetabling Problem (TTP)

- Scheduling of events in timeslots and rooms, e.g scheduling courses, exams, sports events.

- **Problem consists of:**
  - Set of n events E to be scheduled in a set of time slots $T=\{t_1, t_2,\ldots..,t_k\}$, e.g, k=45 (5 days, 9 hours each)
  - Set of rooms R (each has certain capacity)
  - Set of participants S to attend the events, each is pre-assigned to a set of events

# TTP

- **Problem consists of (continue):**
  - Set of features satisfied by rooms and required by events
  - Set of hard constraints involving the participants such as
    - No participant attends more than one event at the same time
    - Only one event is taking place in each room at a given time
    - The room satisfies the features required by the event
  - Set of soft constraints or preferences

# TTP

- A feasible timetable (schedule) is one in which each of the events is assigned a timeslot and a room and satisfies the hard constraints.

- A best solution is a feasible solution that minimizes violation of soft constraints

- Instances might be
  - Participants= students, players, presenters,
  - Events = lectures, conference sessions, shows, sport events
  - Rooms = rooms, show booths, Gyms, courts
  - Features = seats, projectors, type of equipment, capacity

# TTP

- Soft constraints might be:
  - Students shouldn't have more than 4 classes in a row
  - Room utilization shouldn't exceed certain threshold

- The problem is ogten solved in 2 stages:

  **Stage 1:** solve the assignment of events to time slots

  **Stage 2:** assign rooms to events

ACO can be used for one or both stages

# ACO Characteristics

# Characteristics of problems

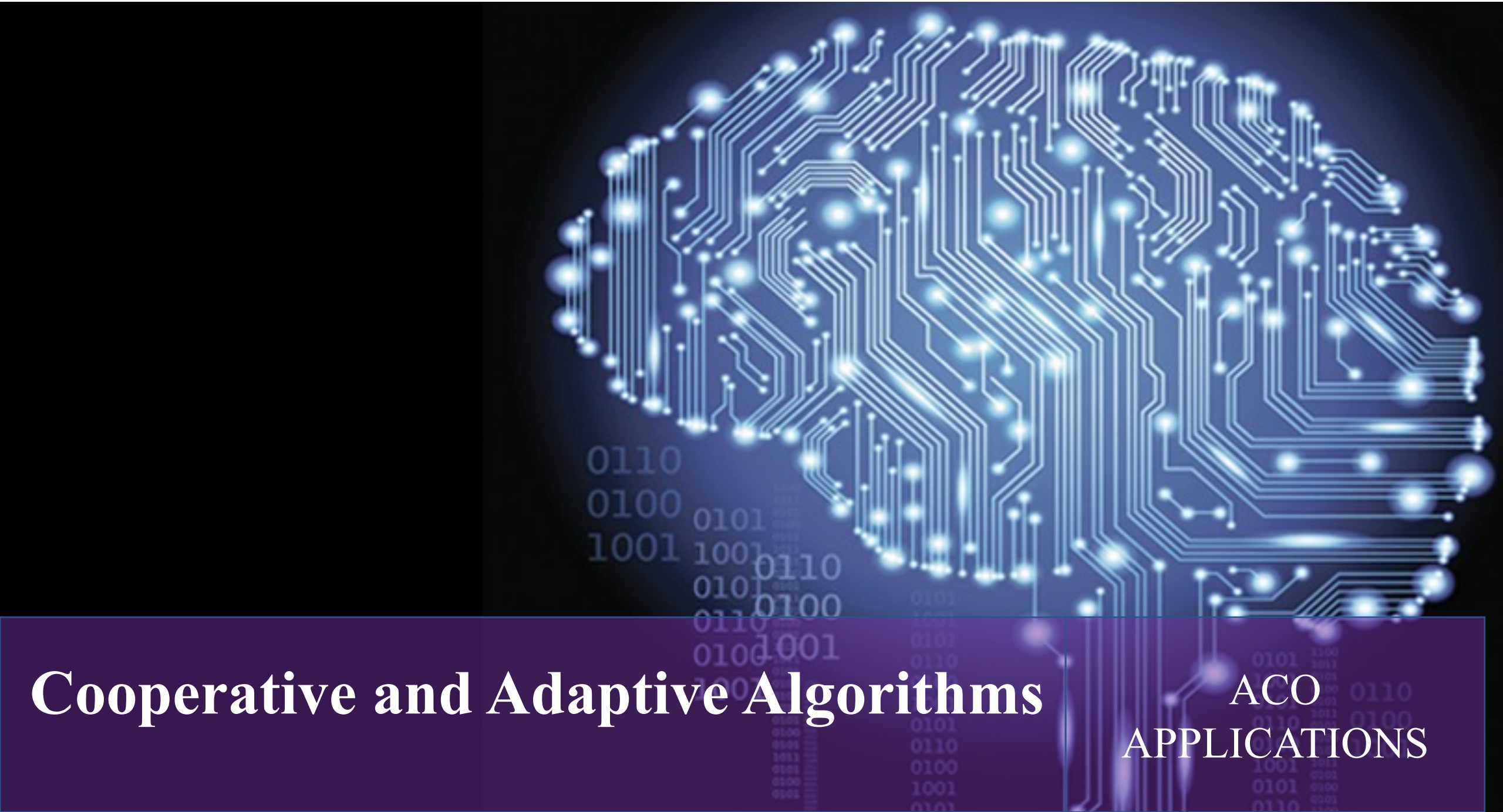
- Combinatorial Optimization problems.
- Not feasible to solve using classical optimization methods if the problem size is large.
- Mainly discrete optimization problems
- There are some variants of ACO to handle continuous optimization problems.

# Advantages and Disadvantages of ACO

- **Advantages:**
  - ACO is stochastic, population based method similar to GA
  - It retains memory of the entire colony instead of just previous generation (as in GA)
  - Less affected by poor initial solutions due to combinations of random path selection.
  - Proved successful in many application
  - Can handle dynamic environments

# Advantages and Disadvantages of ACO

- **Disadvantages:**
  - Mainly experimental,
    - theoretical analysis has been very limited.
    - Some proofs of convergence for some methods under certain conditions exist (MMAS,AS)
  - It uses a lot of parameters, selection of values is experimental.
  - It may take long time to converge.



# Cooperative and Adaptive Algorithms

ACO
APPLICATIONS

# ACO Adaptation

# Adaptation

- Several approaches have been proposed to adapt the parameter values of ACO, which includes:
  - ACSGA-TSP, by Pilat and White,
  - Near parameter free ACS, by Randall.

M. L. Pilat and T. White.“Using Genetic Algorithms to Optimize ACS-TSP”, 2002.

M. Randall. “Near Parameter Free Ant Colony Optimisation”. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. St¨utzle, editors, ANTS Workshop, volume 3172 of Lecture Notes in Computer Science, pp. 374–381. Springer, 2004.

# Adaptation - ACSGA-TSP

- The parameter values in ACS were optimized using a genetic algorithm,
- The idea was to have a GA running on top of the ACS to optimize its parameter values,
- The optimized parameters were:
  - $q_0$, the parameter determining whether the greedy or probabilistic selection is adopted
  - $\rho$, the local pheromone updating factor,
  - $\beta$, the relative importance of the visibility heuristic.

M. L. Pilat and T. White."Using Genetic Algorithms to Optimize ACS-TSP", 2002.

# Adaptation - ACSGA-TSP

- Each ant has its own values of ACS parameters,

- Each GA ant was encoded by a 21-bit string (7 bits/parameter) for ranges:
  - $q_0 \in [0, 1]$, double value,
  - $\rho \in [0, 1]$, double value,
  - $\beta \in [1, 127]$, integer.

# Adaptation - ACSGA-TSP

- The chromosomes were ***randomly initialized***,
- The crossover used was the ***single simple crossover*** to generate two children,
- The mutation used was the ***bit-wise mutation*** (mutate every bit based on a given probability).

# Adaptation - ACSGA-TSP

- **For** each generation **do:**
  - Choose the best 4 individuals,
  - **for** each of the 4 chosen individuals **do**
    - Run ACS-TSP given $\beta$, $\rho$, and $q_o$ value encoded in each individual and record the result as the fitness of the individual
  - **end for**
  - The global pheromone update is done by the ant producing the best tour,
  - Choose the 2 individuals with the best fitness from chosen 4,
  - Produce the 2 children by crossover or copy from the 2 chosen best individuals,
  - Mutate the 2 children,
  - Replace the worst 2 individuals from the chosen 4 in the population with the 2 children
- **end for**

# Adaptation - ACSGA-TSP

- Comparisons are made between the ACSGA and the ACS system using:
  - 20 ants,
  - A fixed number of tour constructions,
  - Crossover probability of 0.9,
  - Mutation probability of 0.01,
  - Averages reported over 10 runs.

# Adaptation - ACSGA-TSP

| Instance | ACS | | ACSGA | |
|---|---|---|---|---|
| | Solution | Iterations | Solution | Iterations |
| eil51 | **428.7** | 2000 | 432.4 | 10000 |
| ft70 | 41144 | 4000 | **40868** | 20000 |
| kroA100 | 21614 | 5000 | **21544** | 25000 |
| Rat783 | 12181 | 100 | **10057** | 500 |

# Adaptation – Near parameter free ACS

- The parameters of ACS were adapted using the same approach for optimizing the problem in hand (an ant approach),

- The optimized parameters are:
  - $q_0$, the parameter determining whether the greedy or probabilistic selection is adopted,
  - $\rho$, the local pheromone updating factor,
  - $\alpha$, the global pheromone updating factor,
  - $\beta$, the relative importance of the visibility heuristic.

M. Randall. “Near Parameter Free Ant Colony Optimisation”. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. St¨utzle, editors, ANTS Workshop, volume 3172 of Lecture Notes in Computer Science, pp. 374–381. Springer, 2004.

# Adaptation - Near parameter free ACS

- Each ant is allowed to:
  - Select the suitable values of its parameters,
  - Select the next solution component.

- Each ant has its own parameter values adaptively selected using an ant approach:
  - A separate pheromone matrix is kept for learning these values,
  - No heuristic information (similar to the distance information in TSP) is used.

# Adaptation - Near parameter free ACS

- Each parameter is given a suitable range to lie in:
  - $q_0 \in [0, 1]$,
  - $\rho \in [0, 1]$,
  - $\alpha \in [0, 1]$,
  - $\beta \in [-5, -1]$.

- The initial value of each parameter is in the middle of its interval.

M. Randall. “Near Parameter Free Ant Colony Optimisation”. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. St¨utzle, editors, ANTS Workshop, volume 3172 of Lecture Notes in Computer Science, pp. 374–381. Springer, 2004.

# Adaptation - Near parameter free ACS

- The pheromone matrix is defined as:

$$v(i, j)$$

Specific parameter $1 \leq i \leq 4$

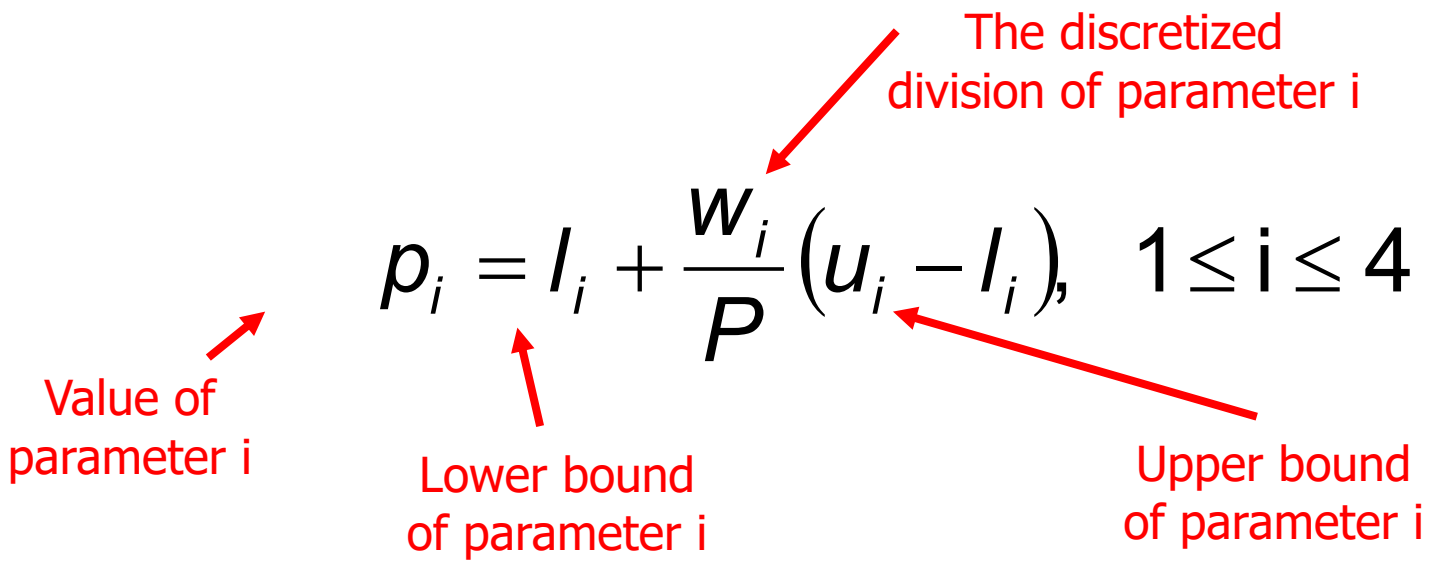
Range of values

- For each parameter, the interval is discretized with a step $P$, thus dividing the interval into $w_i$, $i = 1,.., p$

- The parameter division, $w_i$, is chosen using transition rule based on the pheromone values $v(i, j)$.

# Adaptation - Near parameter free ACS

- The actual value of each parameter is then calculated according:



# Adaptation - Near parameter free ACS

- Comparisons are made between an adaptive version and the ACS system with parameter values from the reference below using:
  - 10 ants,
  - P = 20 divisions,
  - Maximum of 3000 iterations,
  - Applied to TSP with results were reported as ***deviations from the optimal solution***,
  - The results reported are the median values of the relative difference of the best known solution

M. Dorigo and L. M. Gambardella.”Ant Colony System: a cooperative learning approach to the traveling salesman problem”. IEEE Trans. Evolutionary Computation, vol. 1, no. 1, 1997.

# Adaptation - Near parameter free ACS

| Instance | Non-adaptive | | adaptive | |
|---|---|---|---|---|
| | Rel-diff | Time | Rel-diff | Time |
| hk48 | 0.08 | 1.29 | **0.04** | 3.41 |
| eil51 | 2 | 0.49 | **0.23** | 9.75 |
| st70 | 1.33 | 43.48 | **0.89** | 34.9 |
| d198 | **0.33** | 1723.34 | 1.69 | 1975.71 |
| ts225 | **1.15** | 3019.9 | 3.53 | 611.94 |
| pcb442 | **3.53** | 38445.9 | 12.1 | 199.06 |

# ACO Cooperation

# Cooperation

- Two approaches have been studied to investigate the cooperation between different ant colonies:
  - A *heterogeneous* approach, in which the ants in the two colonies have different behaviours (different optimization criteria),
  - A *homogeneous* approach, in which all the ants show a similar behaviour.

# Cooperation

- The **heterogeneous** approach was proposed by Gambardella et al.,
- It was applied to a multi-criteria optimization problem (multi-objective),
- Each colony was used to optimize a different criterion.

L. M. Gambardella, E. D. Taillard and G. Aggazi."MACS-VRPT: A Multiple Ant Colony System for Vehicle Routing Problem with Time Windows". New Ideas in Optimization, McGraw Hill, pp. 63-67, 1999.

# Cooperation

- Different **homogeneous** and parallel approaches were taken, similar to what was done with GAs, including:
  - **A fine-grained implementation**, where each processor holds a single ant,
  - **A coarse-grained implementation**, where each processor holds a complete colony,
  - A complete survey could be found in the reference below.

M. Middendorf, F. Riechle and H. Schmeck. “Information Exchange in Multi Ant Colony Algorithms”. Journal of Heuristics, vol. 8, pp. 305-320, 2002.

# Cooperation

- Middendorf proposed a multi-colony algorithm that have colonies connected in a directed-ring fashion,

- Four information exchange approaches were studied:
  - All colonies get the same global best solution,
  - Circular exchange of locally best solutions,
  - Circular exchange of a number of ants (migrants),
  - A mixture of the two previous approaches.

# Cooperation

- The multi-colony approach was applied to TSP and compared to the performance of a single colony,
- The best exchange method was found to be the circular exchange of locally best solutions,
- A multi-colony approach with a moderate number of colonies is better than a single colony.

# References


- M. Dorigo, V. Maniezzo and A. Colorni.” Ant System: Optimization by a Colony of Cooperating Agents”. IEEE Trans. Syst., Man, and Cybern. Part B, vol. 26, no. 1, 1996.

- T. Stutzle and H. H. Hoos.”*MAX-MIN* Ant System”. *Future Generation Comput. Syst.*, vol. 16, no 8, 2000.

- M. Dorigo and L. M. Gambardella.”Ant Colony System: a cooperative learning approach to the traveling salesman problem”. IEEE Trans. Evolutionary Computation, vol. 1, no. 1, 1997.

- J. Bautista and J. Pereira. “Ant Algorithms for A Time and Space Constrained Assembly Line Balancing Problem”. European Journal of Operation Research, 177, pp. 2016-2032, 2002.

- S. J. Shyu, B. M. T. Lin and T. S. Hsiao.“An Ant Algorithm for Cell Assignment in PCS Netwroks”. IEEE International Conference on Networking, Sensing and Control, pp. 1081-1086, 2004.

# References


- J. R. L. Fournier and S. Pierre.“Assigning Cells to Switches in Mobile Networks using an Ant Colony Optimization Heuristic”.Computer Communications, vol. 28, pp. 65-73, 2005.

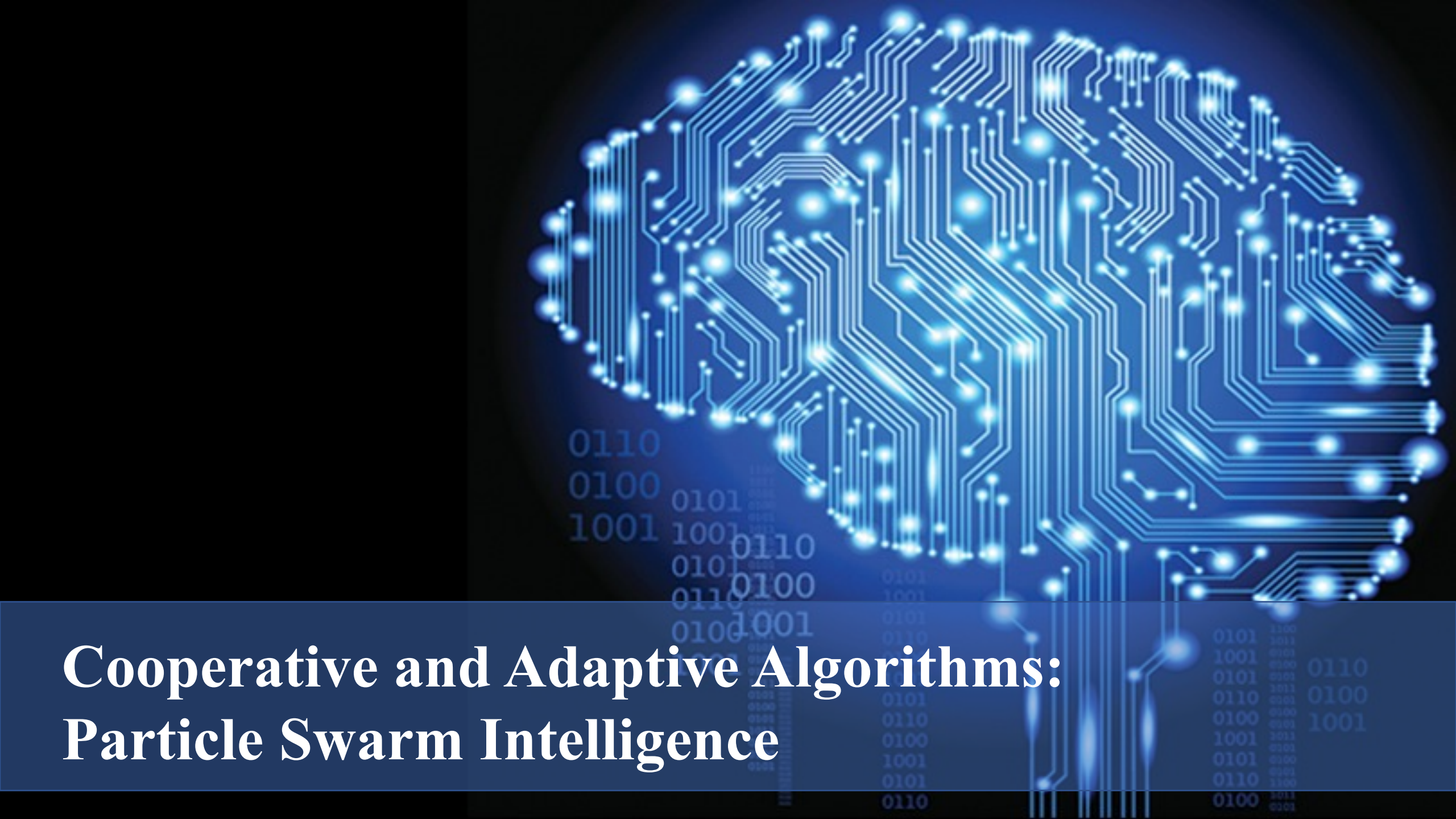
- M. L. Pilat and T. White.“Using Genetic Algorithms to Optimize ACS-TSP”, 2002.

- M. Randall. “Near Parameter Free Ant Colony Optimisation”. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. St¨utzle, editors, ANTS Workshop, volume 3172 of Lecture Notes in Computer Science, pp. 374–381. Springer, 2004.

- L. M. Gambardella, E. D. Taillard and G. Aggazi.”MACS-VRPT: A Multiple Ant Colony System for Vehicle Routing Problem with Time Windows”. New Ideas in Optimization, McGraw Hill, pp. 63-67, 1999.

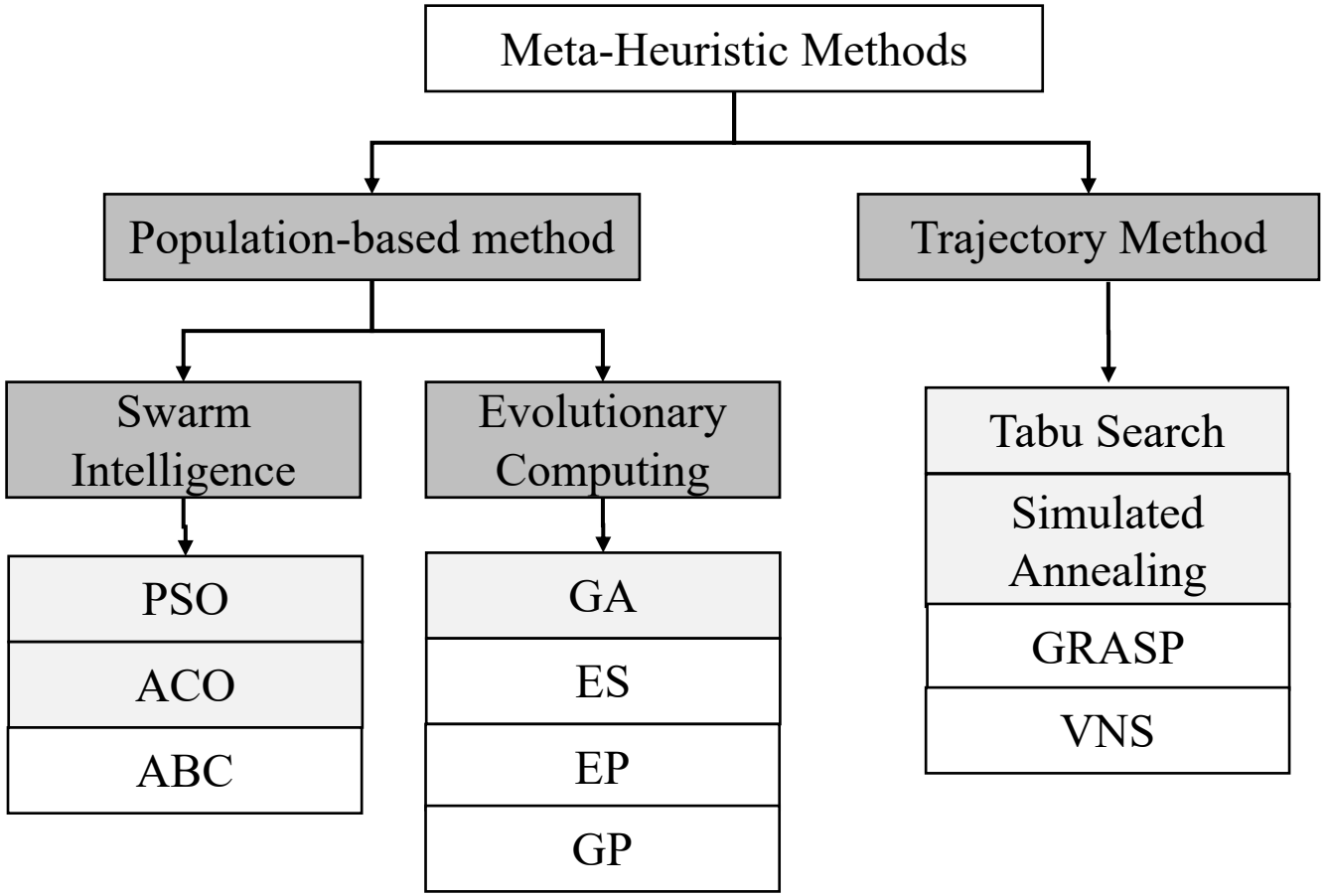
- M. Middendorf, F. Riechle and H. Schmeck. “Information Exchange in Multi Ant Colony Algorithms”. Journal of Heuristics, vol. 8, pp. 305-320, 2002.

- M. Dorigo and L. M. Gambardella.”Ant Colonies for the travelling salesman problem”, BioSystems, 1997.



# Cooperative and Adaptive Algorithms: Particle Swarm Intelligence

# Meta-Heuristics

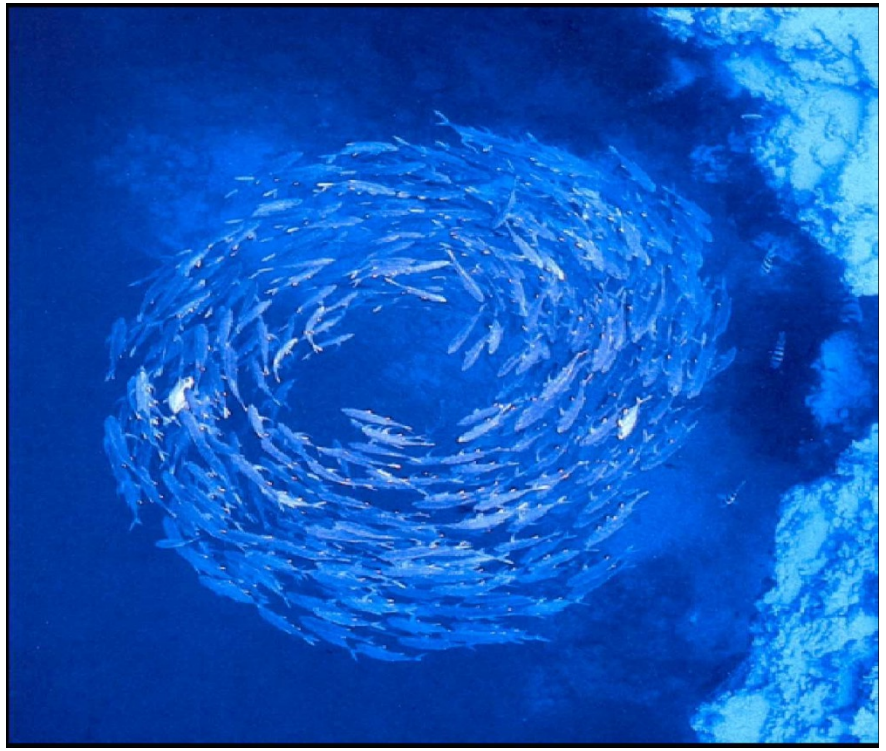


# Particle Swarm Intelligence

## Introduction

# Introduction


Bird flocking – V formation (© Soren Breitling)



Fish schooling (© CORO, CalTech)

# Introduction

- The main idea is to ***simulate*** the collective behavior of social animals
- In particular, bird flocking and fish schooling behaviors
- Unlike some animal teams where there is a leader (e.g a pride of lions or a troop of baboons), the interest here in teams that has ***no leader***
- Individuals have **no knowledge of the global behavior** of the group
- They have the ability to move together based on **social interaction** between neighbours

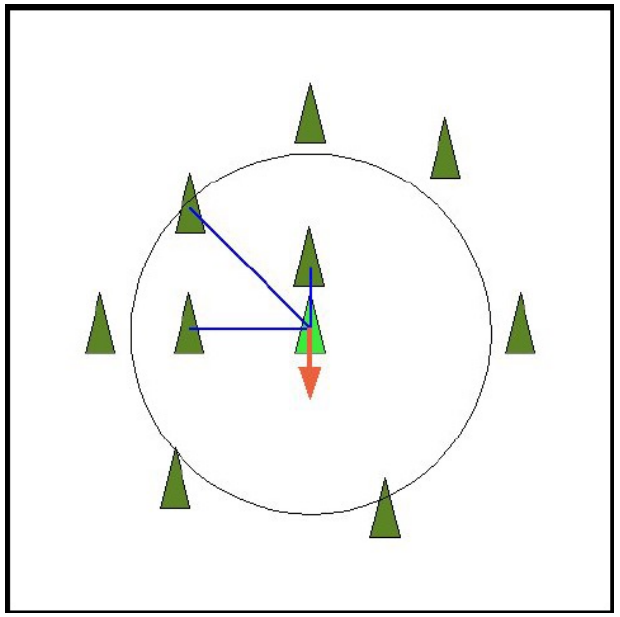
# Introduction

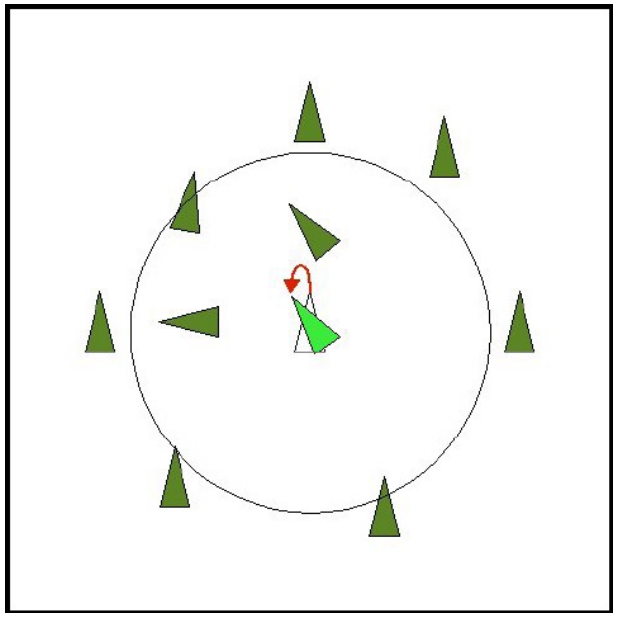
- The first computer program was written by Reynolds in 1986 [1] to simulate swarms for computer graphics and movies,

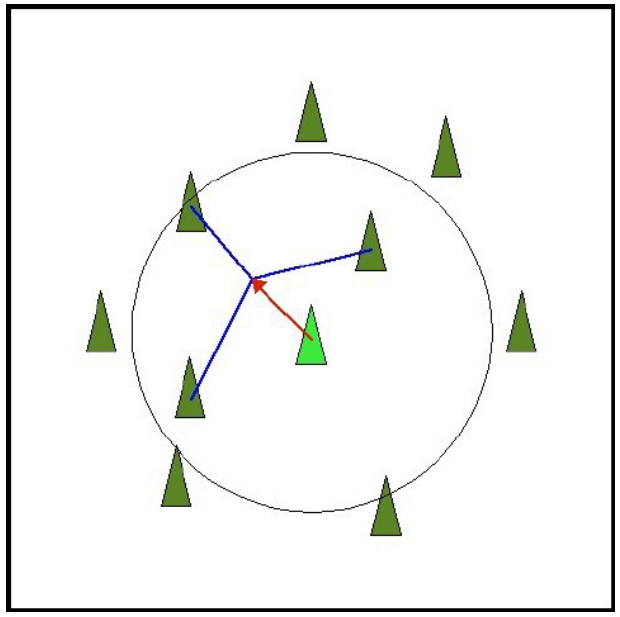
- The work took account of three behaviours:
  - Separation,
  - Alignment,
  - Cohesion.

- For online simulations, refer to http://www.red3d.com/cwr/boids/

# Introduction







**Separation:** Each agent tries to move away from its nearby mates if they are too close (**Collision Avoidance**).

**Alignment:** Each agent steers towards the average heading of its nearby mates (**Velocity matching**).

**Cohesion:** Each agent tries to go towards the average position of its nearby mates (**Centering or position control**).

From [3]

# Introduction

- Heppner and Grenander [4] used a similar flocking model but added a roost (place for birds to rest) as an attractor for the birds.

- The intent was to provide a computer simulation of a flock of birds to understand the underlying rules that enable synchronous flocking

- Kennedy and Eberhart in 1995 [5, 6], introduced an optimization method based on the simple behavior of emulating the success of neighboring individuals.
  - Followed the same steps taken by Reynolds and adding a ***Roost*** as proposed by Heppner and Grenander

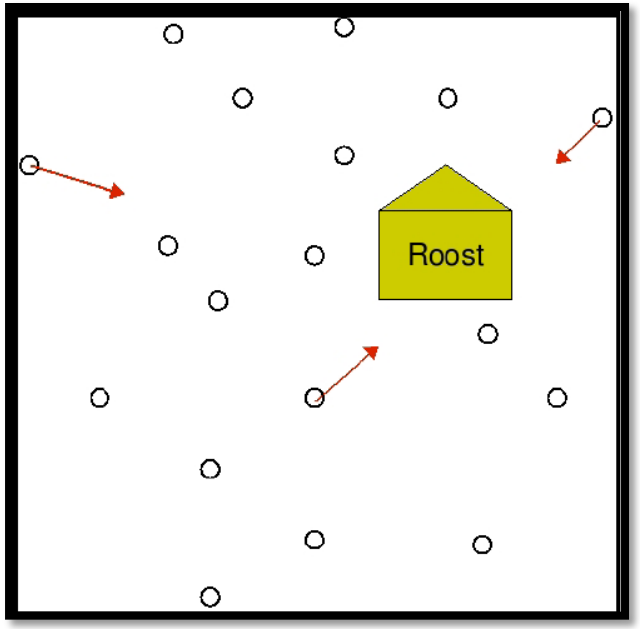
# Introduction

- The ***roost*** is in the form of a memory of previous **own best and neighborhood best** positions (referred to ***cornfield***)

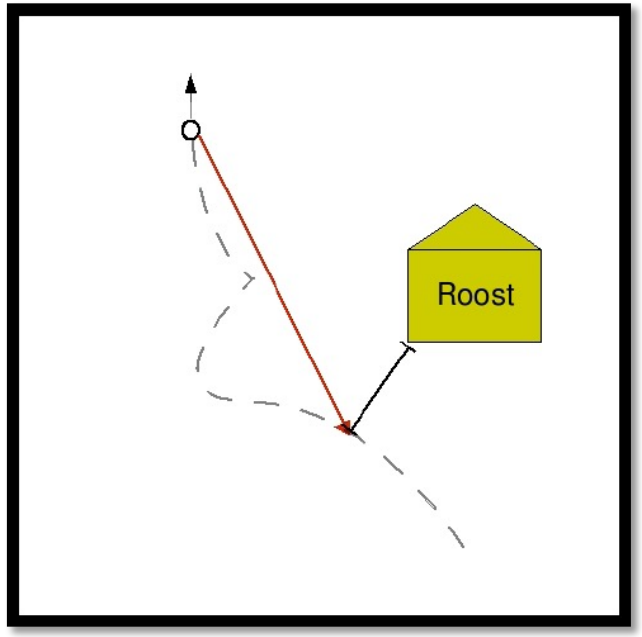
- These two best positions serve as attractor

- By adjusting the positions of the flock proportion to the distance from the best positions, they converge to the goal

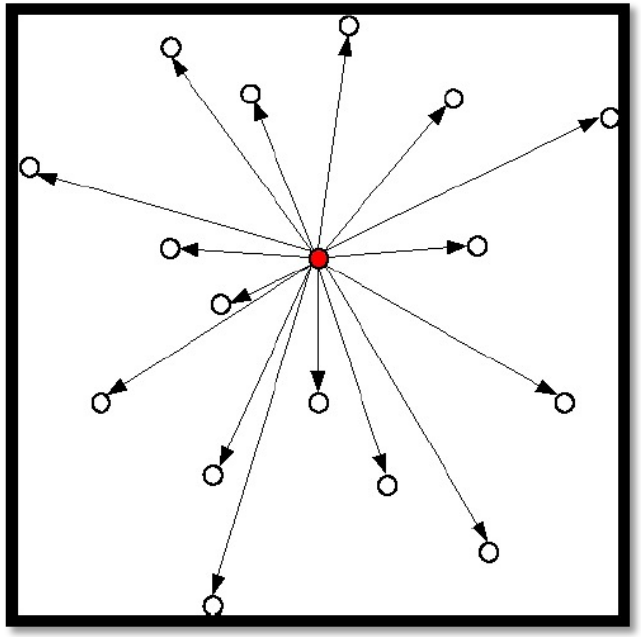
# Introduction




All the individuals are attracted to the roost.




Each memorizes the position in which it was closest to the roost.



Each shares its information with all the others.

From [3]

# Introduction

- At the end of the simulation, all the individuals landed on the roost,
- It was realized, this could be used to solve optimization problems,
- If the distance to the roost was changed by some unknown function, the individuals land on the minimum

# Introduction

- Kennedy and Eberhart called their model **P**article **S**warm **O**ptimization (***PSO***)
- They choose the word ***particle*** to mean individual or candidate solution (in optimization terms) as they felt it is more appropriate for the use with velocity and acceleration
- As their paradigm is a simplified version of bird flocking, they preferred the use of the word ***swarm*** to indicate the population.
- PSO is a population based approach similar to GA and other EC approaches

# PSO vs GA

| PSO | | GA |
|---|---|---|
| Swarm | ⟷ | Population |
| Particle | ⟷ | Individual |
| Fitness | ⟷ | Fitness |
| Less fit don't die | ⟷ | Survival of the fittest |
| Uses past experience and relationship to neighbours | ⟷ | Uses crossover and mutations |

# Particle Swarm Intelligence

## Motion

# PSO - Introduction

- A stochastic optimization approach that manipulates a number of candidate solutions at once,
- A solution is referred to as a ***particle***, the whole population is referred to as a ***swarm***,
- Each particle holds information essential for its movement.

# PSO - Motion

- Each particle holds:
  - Its current position $x_i$,
  - Its current velocity $v_i$,
  - The best position it achieved so far, personal best, $pbest_i$ (sometimes $p_i$ for short),
  - The best position achieved by particles in its neighbourhood *Nbest*
  - If the neighbourhood is the whole swarm, the best achieved by the whole swarm is called *global best*, $gbest_i$ (sometimes $p_g$ for short).
  - If the neighbourhood is restricted to few particles, the best is called *local best, lbest (or* $p_l$*)*

# PSO - Motion

- Each particle adjusts its velocity to move towards its personal best and the swarm neighbourhood best,

- After the velocity is updated, the particle adjusts its position.

# PSO - Motion

- **Equations of motion:**

$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id}\left(pbest_t^{id} - x_t^{id}\right) + c_2 r_2^{id}\left(Nbest_t^{id} - x_t^{id}\right)$$

$$x_{t+1}^{id} = x_t^{id} + v_{t+1}^{id}$$

- **Where**
  - $v$ is the velocity of particle *id,*
  - $w$ is the inertia weight,
  - $c_1$, $c_2$ are the acceleration coefficients,
  - $r_1$,$r_2$ are randomly generated numbers in [0, 1],
  - $x$ is the position of the particle
  - $t$ is the iteration number,
  - $i$ and $d$ are the particle number and the dimension.

# PSO - Motion

$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id} \left( pbest_t^{id} - x_t^{id} \right) + c_2 r_2^{id} \left( Nbest_t^{id} - x_t^{id} \right)$$

→ **Inertia**

→ **Cognitive component**

→ **Social component**

- The inertia component accommodates the fact that a bird (particle) cannot suddenly change its direction of movement,

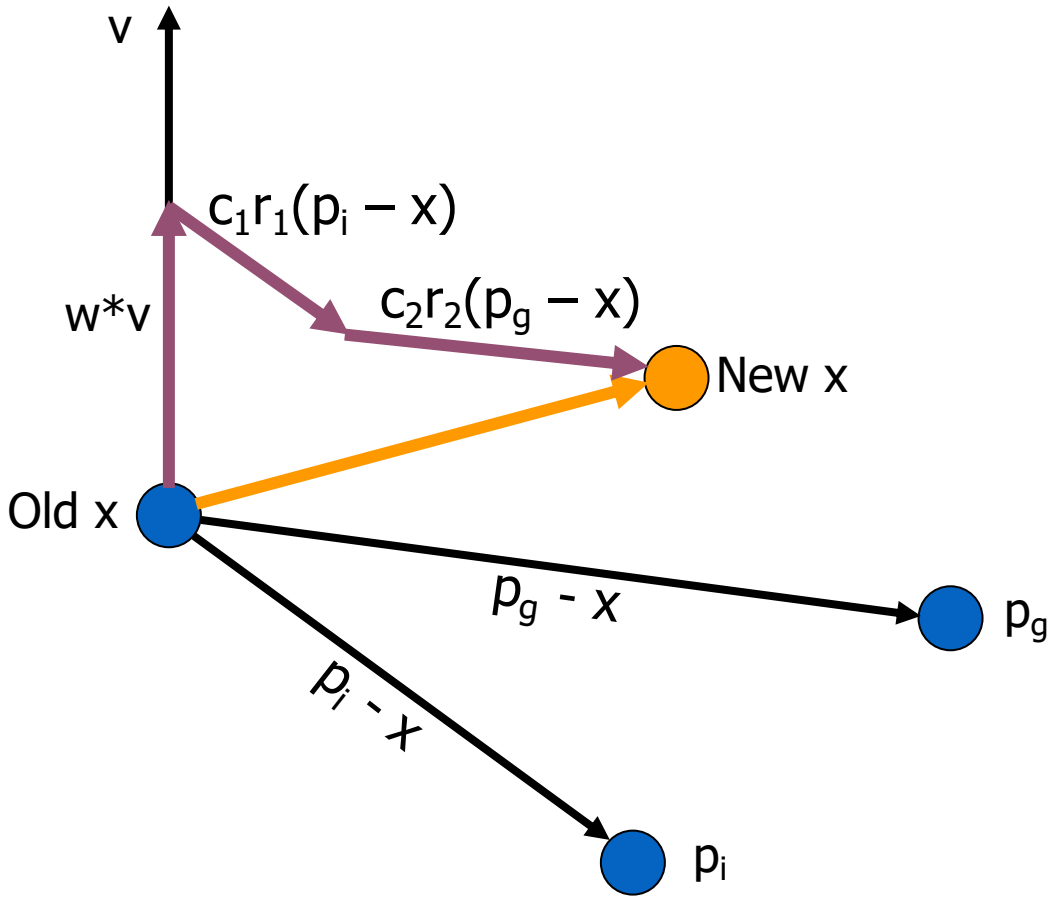
- The $c_1$ and $c_2$ factors balance the weights in which each particle:
  - Trusts its own experience, *cognitive component*,
  - Trusts the swarm experience, *social component*.

# PSO - Motion

- Note that the random numbers are generated for each dimension and not for each particle,
  - if the function you are optimizing has 3 variables, the particle will have 3 dimensions

- If the numbers are generated for each particle, the algorithm is referred to as ***linear PSO***, which usually produces sub-optimal solutions in comparison with PSO.

# PSO - Motion




- An important factor to set is the maximum velocity allowed for the particles $V_{max}$ :
  - If too high, particles can fly past optimal solutions,
  - If too low, particles can get stuck in local optima.

- Usually set according to the domain of the search space.

# PSO - Motion

- After that, each particle updates its own personal best (assuming a minimization problem):

$$pbest_{t+1}^{i} = \begin{cases} x_{t+1}^{i} & ,if\ f\left(x_{t+1}^{i}\right) \le f\left(pbest_{t}^{i}\right) \\ pbest_{t}^{i} & ,\text{otherwise} \end{cases}$$

- After that, each swarm updates its global best (assuming a minimization problem):

$$\text{Nbest}_{t+1}^{i} = \arg \min_{pbest_{t+1}^{i} \in N} \text{f}\left(\text{pbest}_{\text{t+1}}^{\text{i}}\right)$$

# Particle Swarm Intelligence

## Algorithms

# PSO – A simple algorithm (Synchronous update)

- Initialize the swarm,
- While *termination criteria* is not met
  - For each particle
    - Update the particle's velocity,
    - Update the particle's position,
    - Update the particle's personal best,

  end for
  - Update the Nbest,

end while

# PSO – A different algorithm (Asynchronous update)

- Initialize the swarm,
- While *termination criteria* is not met
  - For each particle
    - Update the particle's velocity,
    - Update the particle's position,
    - Update the particle's personal best,
    - Update the Nbest,

  end for

end while

The neighbourhood best update is moved into the particles update loop

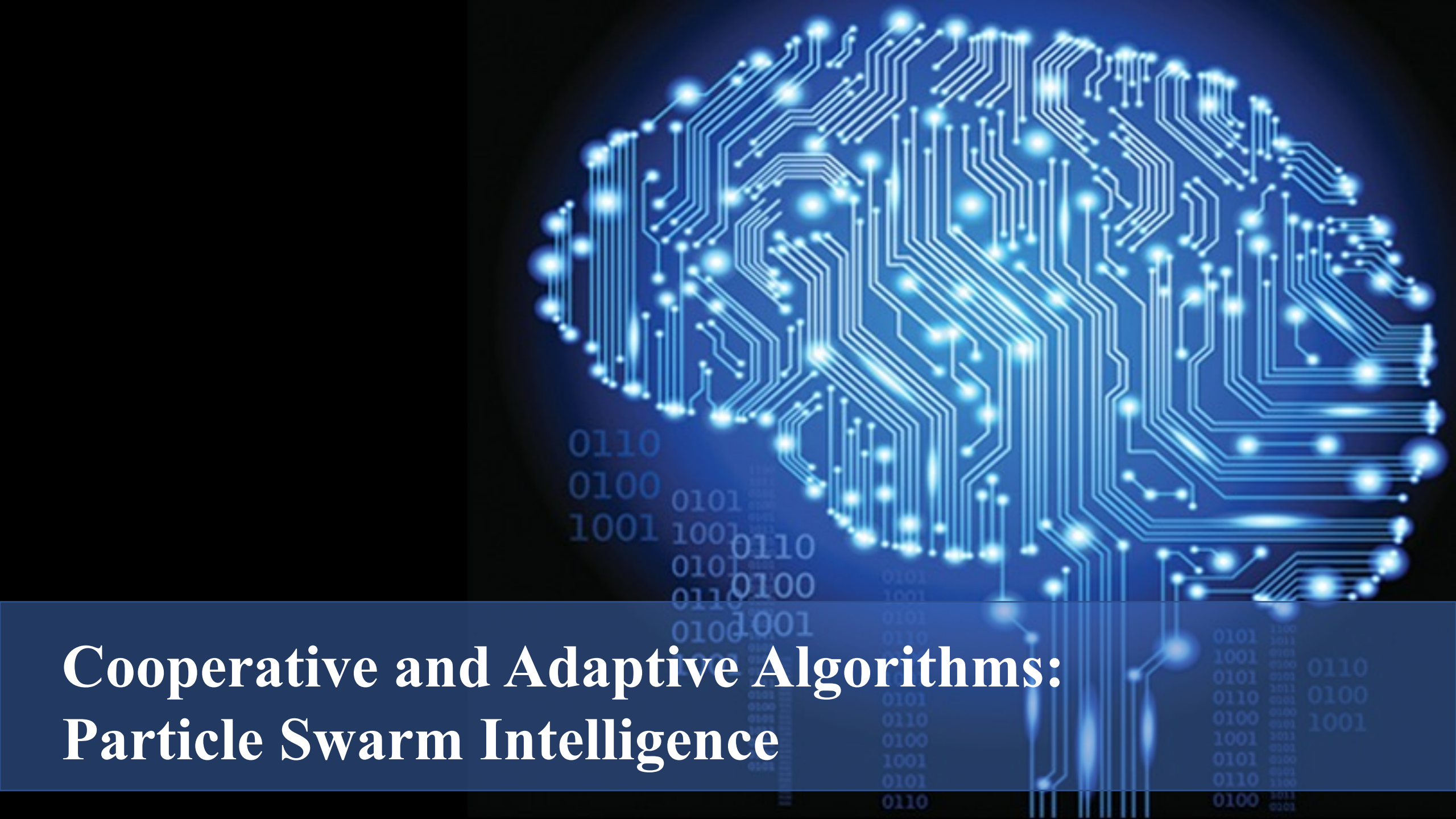
# PSO Algorithms

- ***Synchronous*** version, if the neighbourhood best is updated after all the population has been updated as well,

- ***Asynchronous*** version, if the neighbourhood best is updated after every particle,

- Asynchronous version usually produces better results as it causes the particles to use a more up-to-date information.

- Termination Criteria can be:
  - Max number of iterations
  - Max number of function evaluations
  - Acceptable solution has been found
  - No improvement over a number of iterations.



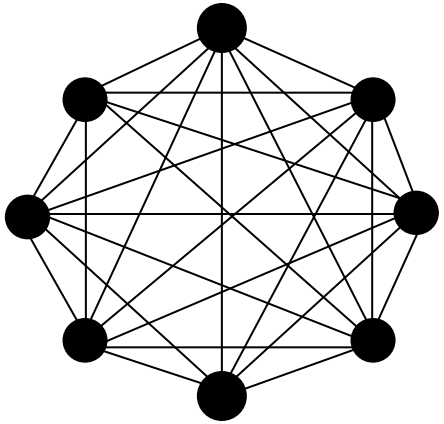
# Cooperative and Adaptive Algorithms: Particle Swarm Intelligence

# Particle Swarm Intelligence

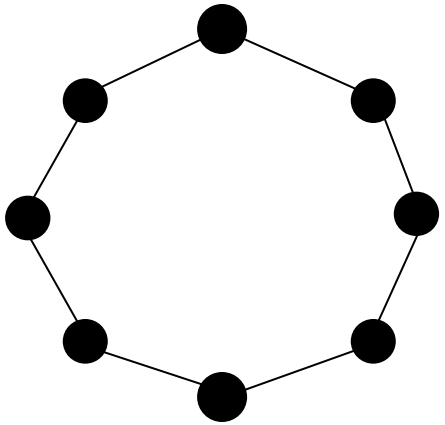
## Neighborhood

# PSO - Neighbourhoods

- In PSO, each particle shares its personal best information with other particles,
- Selecting a ***proper neighbourhood*** affects the ***convergence*** and also helps in avoiding getting ***stuck at local minima***,
- Different neighbourhood topologies have been introduced [9].



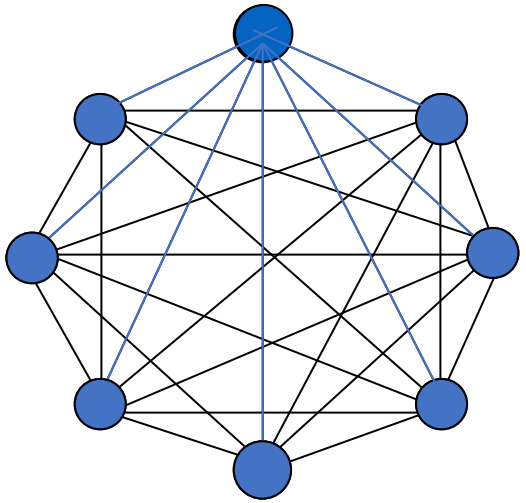
Star topology – global best



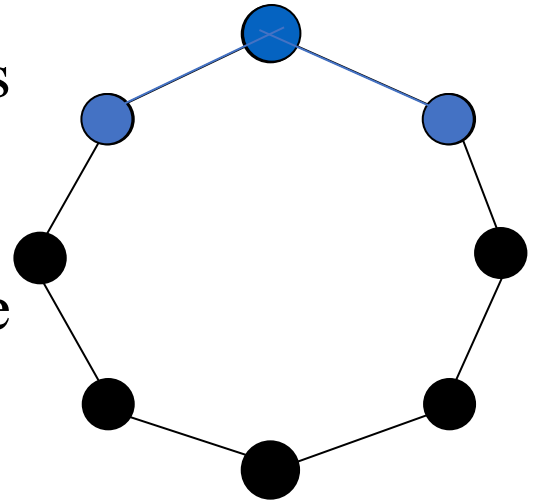
Ring topology – local best

# PSO - Neighbourhoods

- *Star Topology:*
  - *gbest model:* each particle is influenced by all the other particles,
  - The fastest Propagation of information in a population,
  - Particles can get stuck easily in local minima.



- Ring Topology:
  - *lbest model:* each particle is influenced only by particles in its own neighbourhood,
  - The propagation of information is the slowest,
  - Doesn't get stuck easily in local minima but might increase the computational cost.



# PSO - Neighbourhoods

- The most obvious way to select the neighbours for a certain particle *m* is to choose the particles that are closest to it in the search space (physical neighbors)

- The notion of closest is based on the ***distance in the Cartesian space***,
  - This approach might be computationally expensive as distances has to be computed each time the particle changes its position.

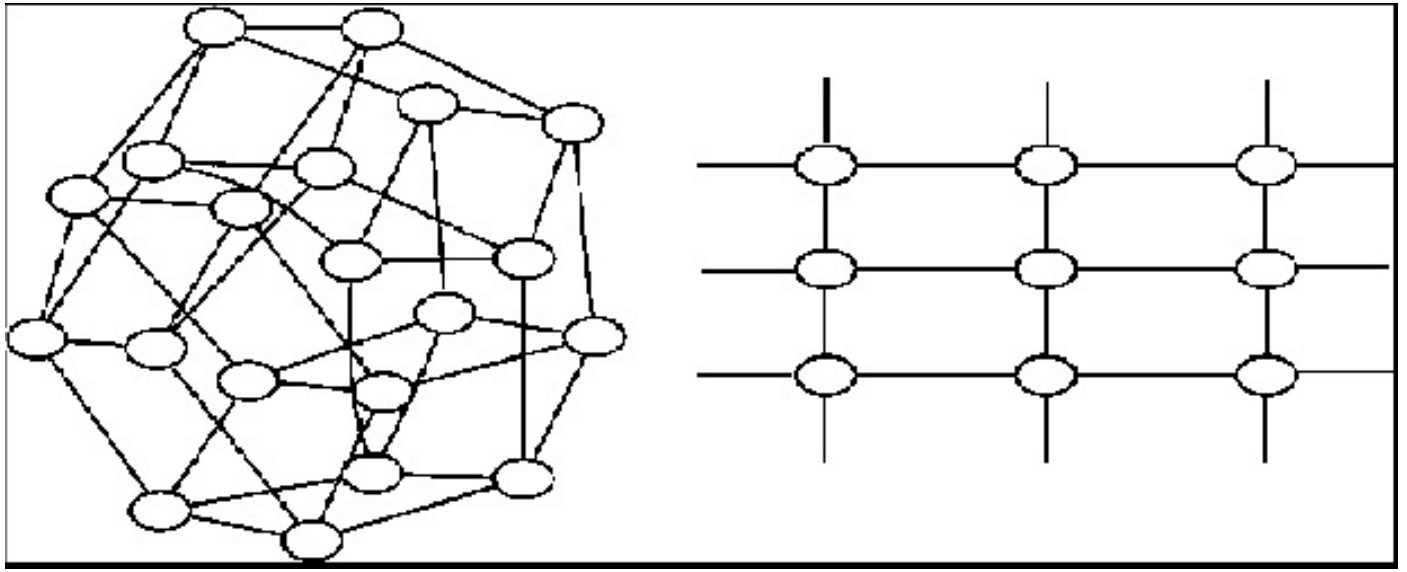
- The particles can be kept in a matrix data structure for example,
  - The most commonly used approach is to pick the particles that are stored next to *m* in the matrix (social neighbors)

- The performance is affected by the size of the neighbourhood selected (2, 4, 6, …).

# PSO - Neighbourhoods

- The most successful neighbourhood structure was the square topology (Von Neumann model),

- Formed by arranging the particles in a grid and connecting the neighbours above , below and to the right and left.



The Von Neumann model [10]

The complete population

A local region

# Particle Swarm Intelligence

## Initialization

# Initialization

- For each particle, the particle position and velocity need to be initialize.
- Particles positions can be initialized randomly in the range.
- Particles velocities can be initialized to zero or small values,
  - large values will result in large updates which may lead to divergence
- Personal best position is initialized to the particle's initial position

# Initialization

- Parameters $w, c_1, c_2, r_1, r_2$

$$v_{t+1}^{id} = w * v_t^{id} + c_1 r_1^{id} \left( pbest_t^{id} - x_t^{id} \right) + c_2 r_2^{id} \left( Nbest_t^{id} - x_t^{id} \right)$$

- Using $c_1 = 0$ reduces the velocity model to Social-only model (particles are all attracted to *Nbest*)

- Using $c_2 = 0$ reduces to cognition-only model (particles are independent hill climbers)

# Initialization

- In most applications $c_1 = c_2$
- Small values will result in smooth trajectories
- High values cause more acceleration with abrupt movement towards or past good regions
- The value of $w$ is important to balance exploration and exploitation
  - Large values promote exploration and small promote exploitation (allowing more control to cognitive and social components)

# Particle Swarm Intelligence

## Convergence

# PSO - Convergence

- An important question is: What are the suitable parameter values that guarantee the swarm convergence?

- The wrong settings can cause the particles to explode out of the search space,

- One of the simplest theoretical studies for PSO was proposed by Trelea in 2003 [7],
  - The study followed the same steps taken in dynamic systems theory,
  - It was carried using a deterministic PSO algorithm, randomness was removed.
  - The random numbers were replaced by their expected values

# PSO - Convergence

- For a one-dimensional one-particle system:

$$r_1 = r_2 = \frac{1}{2}$$

$$v_{t+1} = w * v_t + \frac{c_1}{2}(pbest_t - x_t) + \frac{c_2}{2}(gbest_t - x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

- Let

$$c = \frac{c_1 + c_2}{2}$$

$$p = \frac{c_1}{c_1 + c_2} pbest + \frac{c_2}{c_1 + c_2} gbest$$

# PSO - Convergence

- Hence, the equations of motion would be:

$$v_{t+1} = w * v_t + c(p - x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

- The equations could be rewritten in matrix form:

$$y_{t+1} = Ay_t + Bp$$

where

$$y_t = \begin{bmatrix} x_t \\ v_t \end{bmatrix}, A = \begin{bmatrix} 1-c & w \\ -c & w \end{bmatrix}, B = \begin{bmatrix} c \\ c \end{bmatrix}$$

# PSO - Convergence

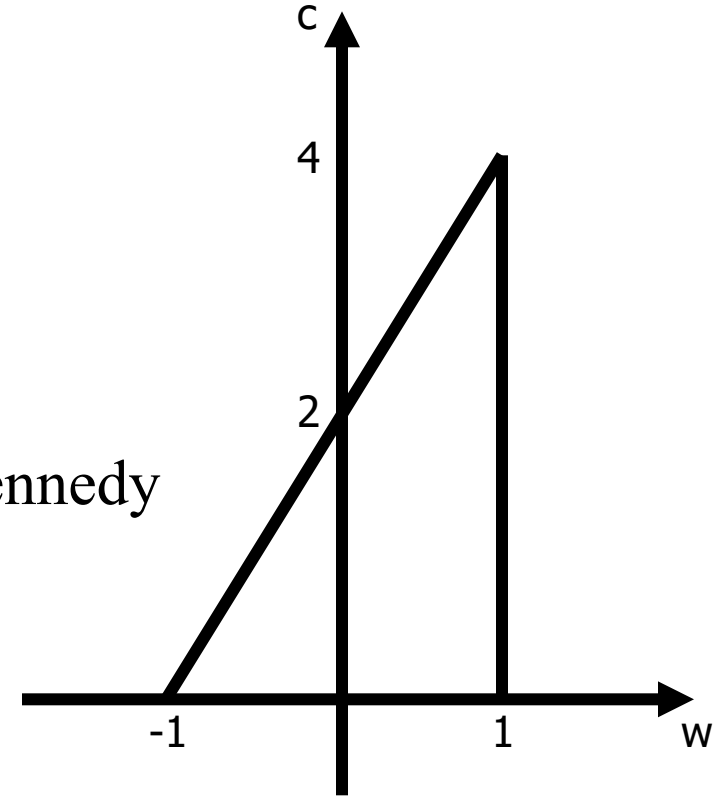
- By analyzing the characteristic equation, we can find the conditions necessary for stability:

$$w < 1,$$

$$c > 0,$$

$$2 * w - c + 2 > 0$$

- The convergence domain would be inside the triangle shown.




- The widely used set of parameters is proposed by Clerc and Kennedy in 2003 [8]:

$$w = 0.792,$$

$$c1 = c2 = 1.4944$$

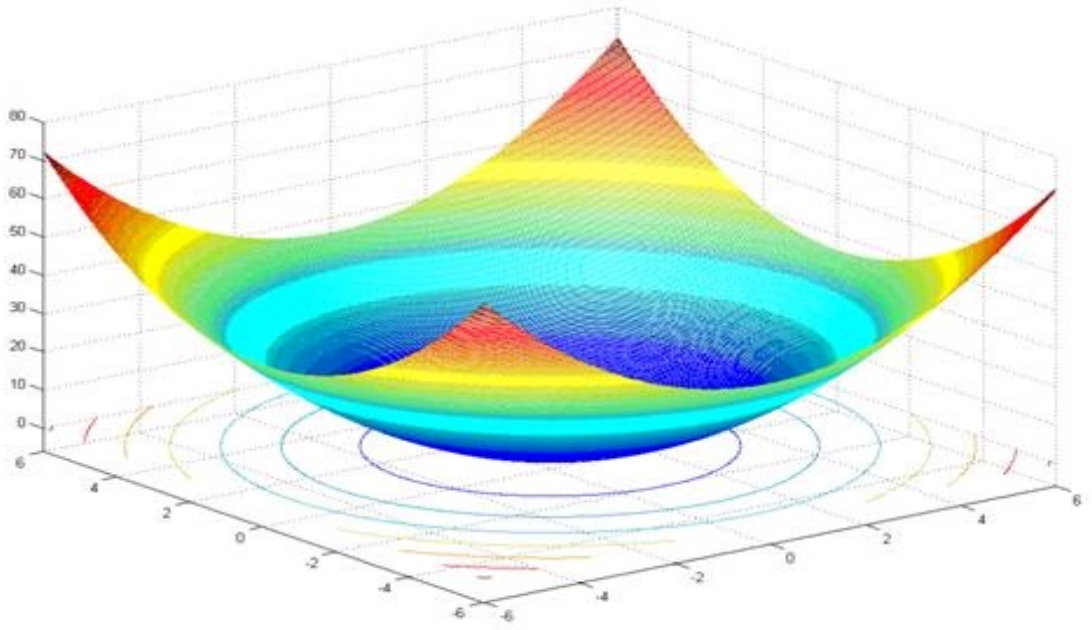
# Particle Swarm Intelligence

## PSO vs GA

# PSO – A Comparison with GAs

- **Spherical Function:**

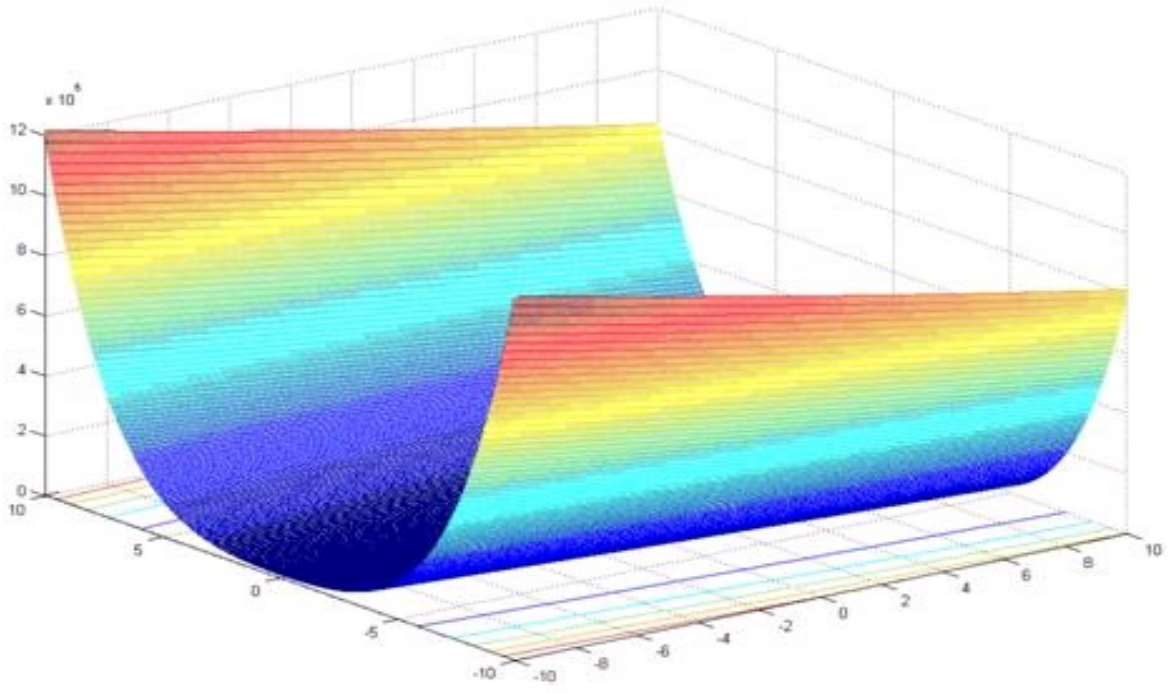
$$f(x) = \sum_{i=1}^{d} x_i^2$$



# PSO – A Comparison with GAs

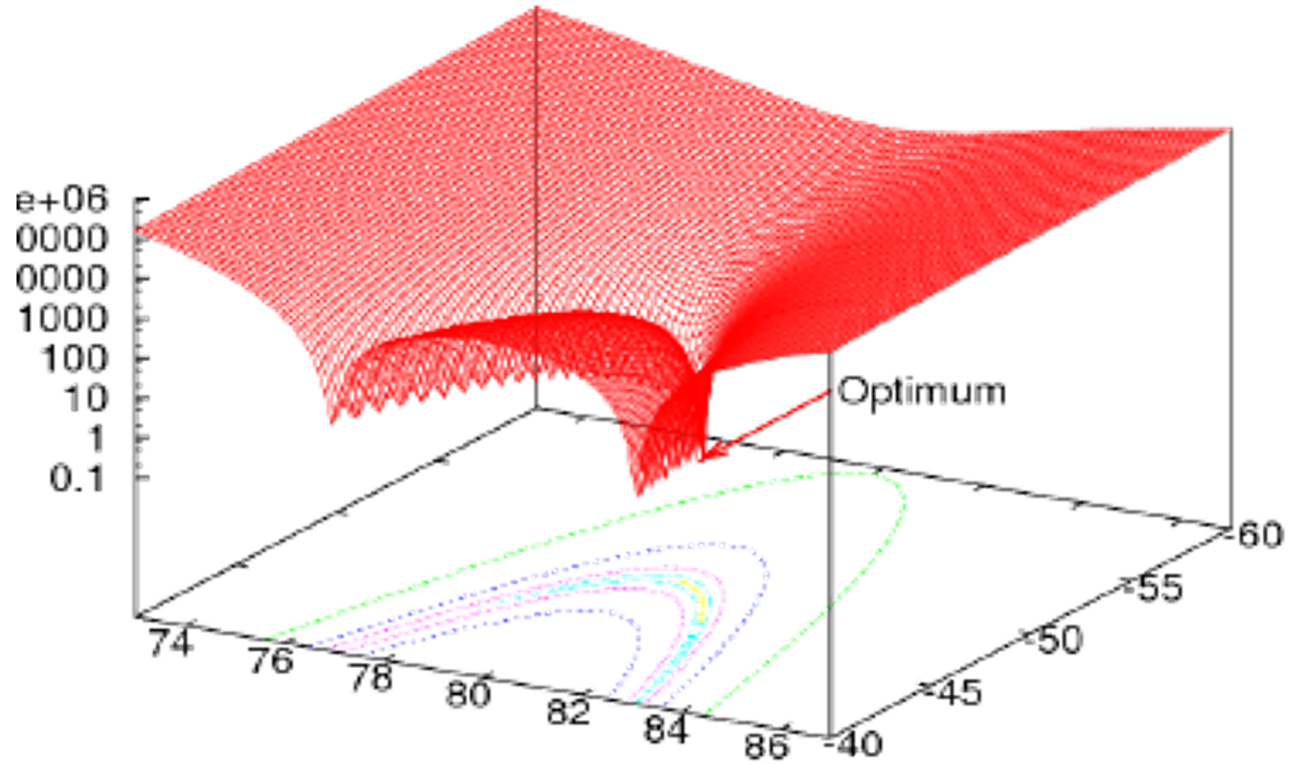
- **Rosenbrock Function:**

$$f(x) = \sum_{i=1}^{d-1} \left[ (1 - x_i)^2 + 100 \left( x_{i+1} - x_i^2 \right)^2 \right]$$



# PSO – A Comparison with GAs

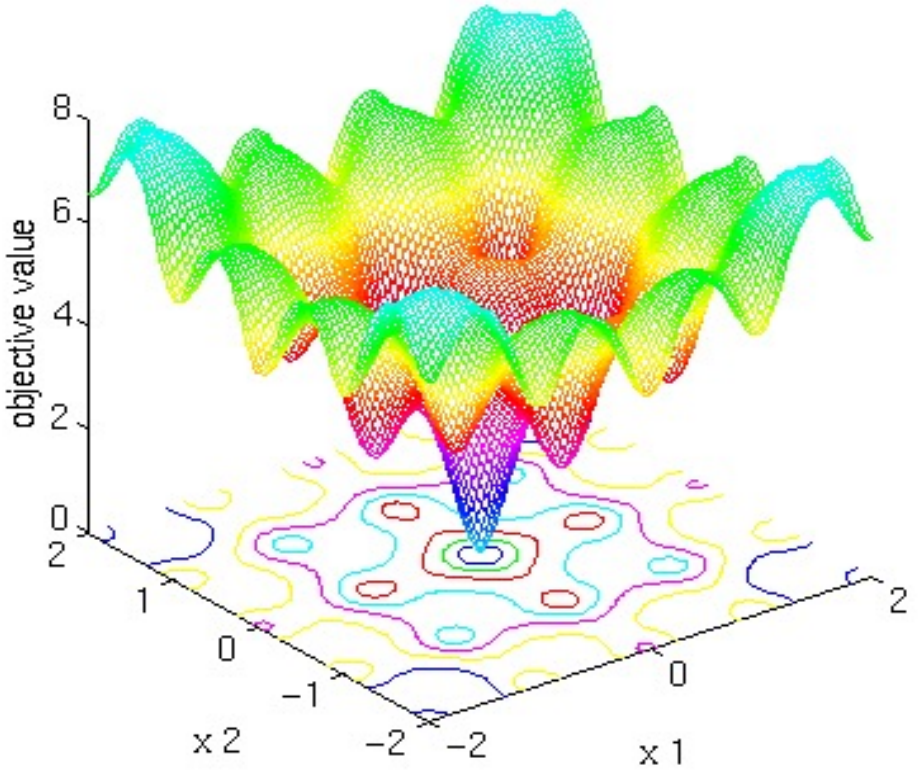
- **Rosenbrock Function:**



# PSO – A Comparison with GAs

- **Ackley Function:**

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$$

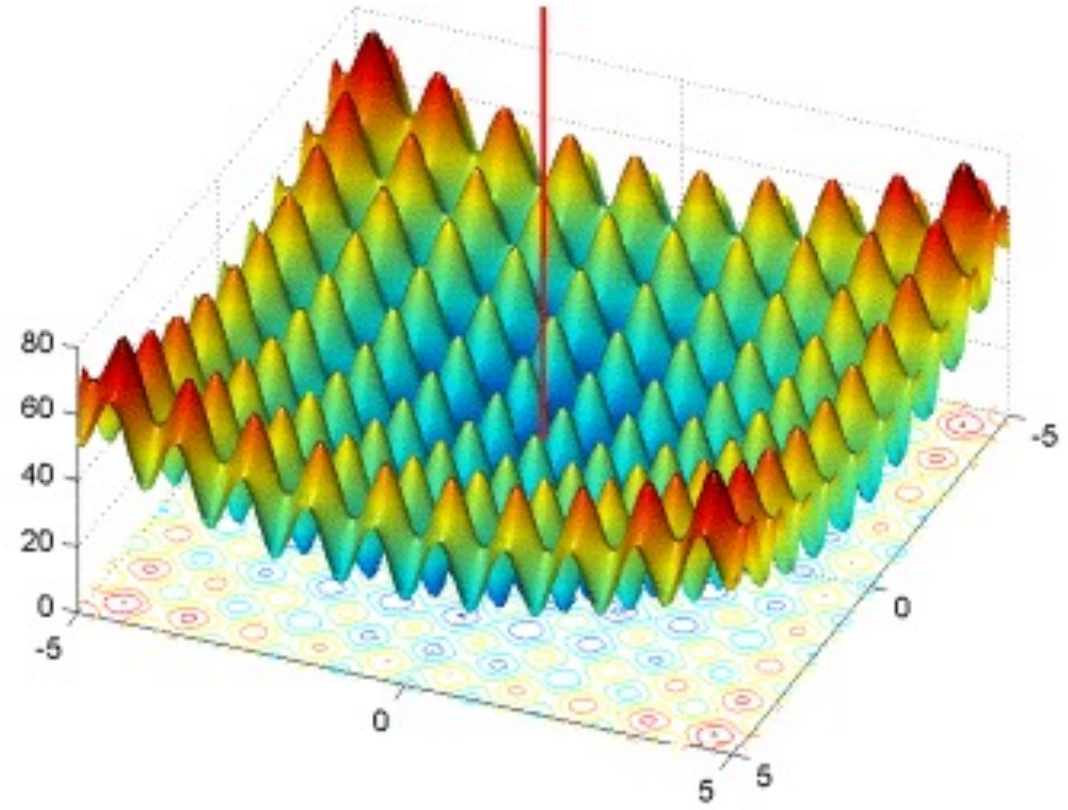


# PSO – A Comparison with GAs

- **Rastrigin Function:**

$$f(x) = \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i) + 10$$

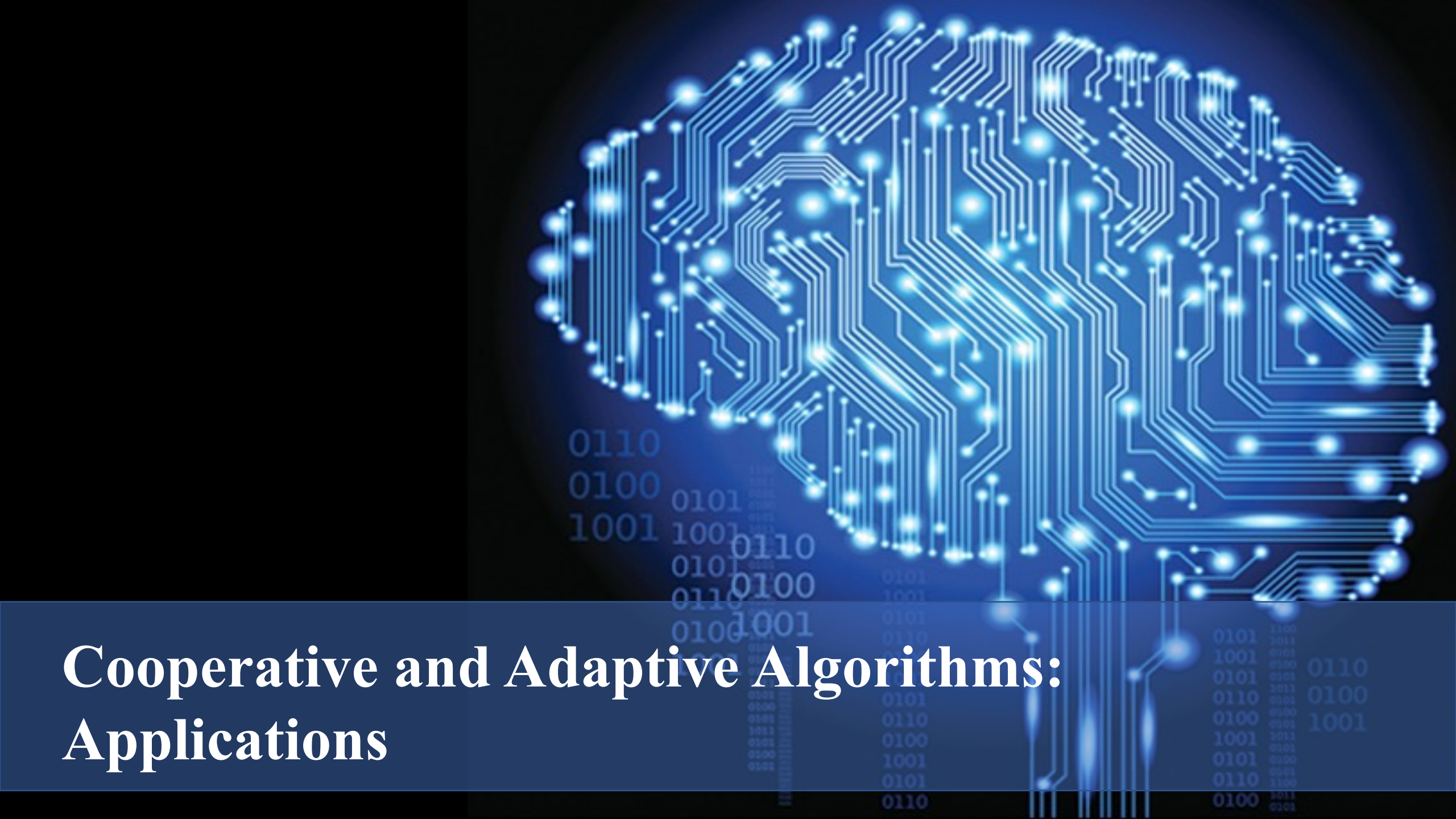
Global minimum at [0 0]



# PSO - A comparison with GAs

- A comparison is made between PSO and GAs using the four functions (Spherical, Rosenbrock, Ackley and Rastrigin):
  - Both algorithms use 10 particles (individuals) and run for 1000 iterations, using Clerc and Kennedy parameters.
  - For a dimensionality of 10,
  - The results are the averages reported over 20 runs.

| Benchmark | GA - Elitism | PSO - *gbest* |
|---|---|---|
| Spherical | 0.0099 | **2.3273e-17** |
| Rosenbrock | **5.5760** | 9.6467 |
| Ackley | 0.9451 | **0.3047** |
| Rastrigin | 38.2186 | **18.7730** |



# Cooperative and Adaptive Algorithms: Applications