



# Introduction to Artificial Intelligence

Week 11



# Swarms and Ants



# Particle Swarm Optimization



# Artificial Life

- Implementation of Biological Organisms on the Macro Scale
- Two areas where this study focuses:
  - Using computational techniques to further the study of biology
  - Using these biological techniques to advance computational methods
- Karl Sims Creatures
  - Evolutionary Biology – Alife
- PSO
  - Social interactions of birds flocking and fish schooling



# PSO

- Not intended as a search technique originally
- Simulation of Flocking behaviours of birds
- Areas of food and without food
- High and low areas of fitness
- Communication between members of the flock
  - Instant
  - Accurate



# PSO



- Used for real number optimization
  - Single points moving over a space
  - Compare ES
- Parallel Search
  - Flocks of particles over the space
  - All individual search points
- Particles have a location and a velocity
- Velocity based on their momentum, their tendency to fly to the best location they personally have seen, and to the best location of the flock



# PSO Usage Fields



- Neural networks
- Image recognition
- Optimization of non-linear N-dimensional problems
- Designing
- Movies
- Biomechanics
- Biochemistry
- Match Processing
- Data Mining
- etc.



# Controls on Exploration and Exploitation

- Explore
  - More emphases on momentum
  - Wider Circles about known optimums
- Exploitation
  - Local Best/Global Best pull the particles back into the high valued positions



# Particle Update

- Particle
  - $Pos[]$  – current position of the particle
  - $V[]$  – current velocity vector
  - $myBest[]$  – best position found by the particle
- Flock has a best
  - $globalBest[]$  – best position found by any member of the flock
- Particle's new velocity is
  - $V[t+1] = m*V[t] + c_1*rand[0,1]*(myBest[]-Pos[t]) + c_2*rand[0,1]*(globalBest[]-Pos[t])$  **(1)**
  - $M$  – momentum value (usually 1)
- Particle update is
  - $Pos[t+1] = Pos[t] + V[t+1]$  **(2)**

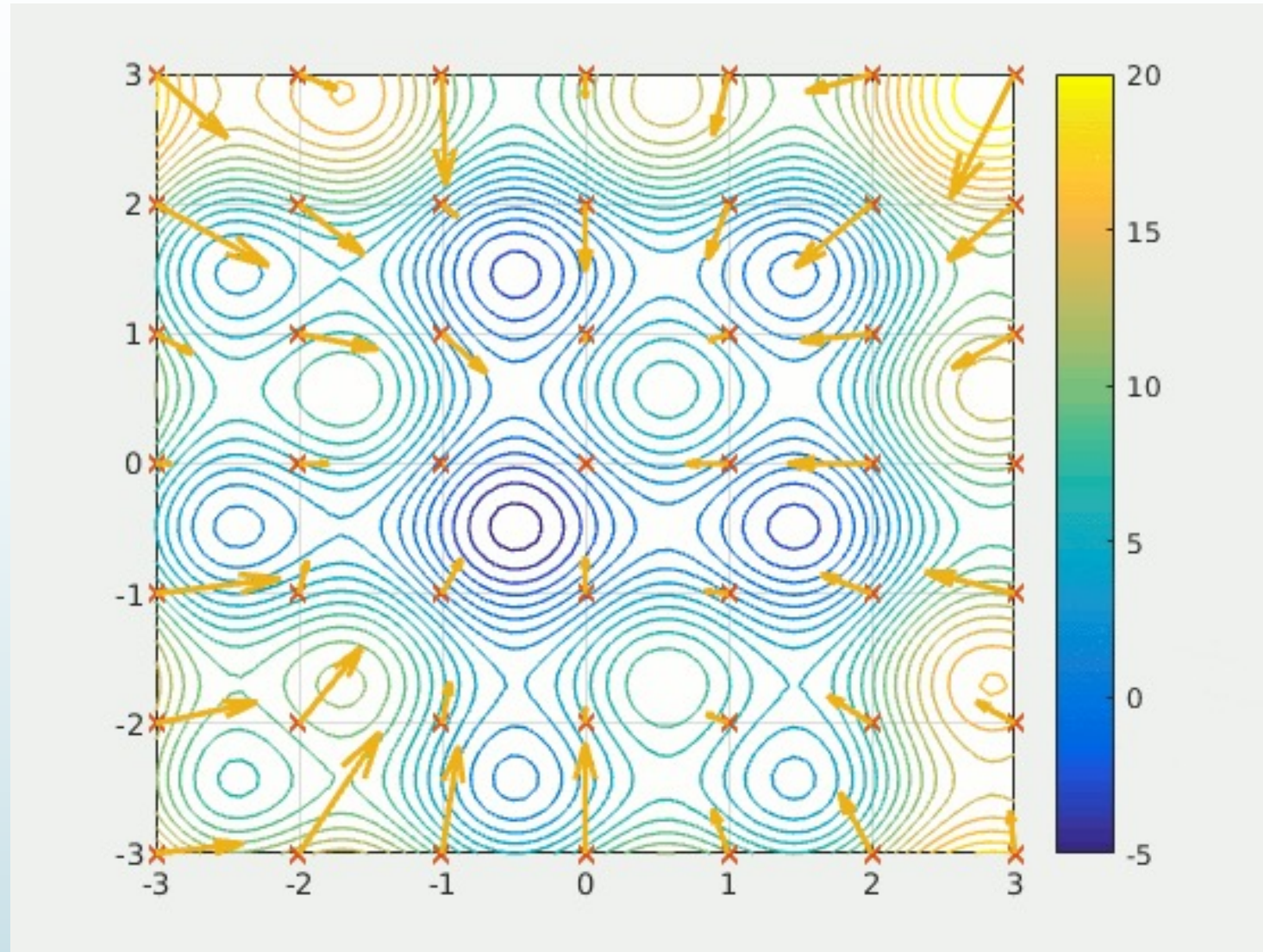


# PSO Loop



- For each particle
  - Initialize particle (usually random position and velocity)
  - END
- Do
  - For each particle
    - Calculate fitness value
    - If the fitness value is better than the best fitness value (pBest) in history, set current value as the new pBest
  - End
  - Choose the particle with the best fitness value of all the particles as the gBest
  - For each particle
    - Calculate particle velocity according equation (1)
    - Update particle position according equation (2)
  - End
- While maximum iterations or minimum error criteria is not attained

# Graphical Simulation





# Comparison to Evolutionary Techniques

- Bio-inspired
- Population Based
- Idea of Fitness
- Optimization via search
  
- Swarms can have physical realities
  - Swarm Robot Intelligence
  - Intelligent Agents
- PSO has memory, EAs don't



# PSO for Discrete Problems

- Quantization of the space
  - Map the values in the real space onto real valued grid
  - Map can be a simple rounding or a more complicated mapping
- Might want to use the same kind of ordered selection mapping based on the real values as we did for ordered gene problems



# Distributing Particles

- Multiple computational elements
  - Cluster-based computing
- Each node has its own flock of particles
- Flocks of Flocks
  - Higher level particles which take on their starting positions to be the global best of one of the flocks – moved to that position as time changes
- Distributed Best
  - The best of all global bests
  - Updated to all the flocks
  - Particles see this as another factor in their equation
- Search space can be divided among the flocks
  - No fly zones
  - Anything outside of the zone has 0 fitness by fiat



# Movies

- Number of hard to make effects shots with large crowds or battles
- Cost of extras is high
  - Casting
  - Administration
  - Wages
  - Catering
- Special effects software for CGI assets



# Movies

- Flocking Agent Simulator for LARGE crowds
- First Developed for Lord of the Rings: Return of the King and has been seen in movies such as:
  - I, Robot – Robots
  - Happy Feet – Dancing Penguins
  - Fast and the Furious: Tokyo Drift – Cars
  - Hugo – Population in a Train Station
  - Avatar - Creatures
  - World War Z - Zombies



# Agents





# Agents

- "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors."
  - Russell, Stuart J. and Peter Norvig (1995), Artificial Intelligence: A Modern Approach
- "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."
  - Maes, Pattie (1995), Artificial Life Meets Entertainment: Life like Autonomous Agents, *Communications of the ACM*, 38, 11, 108-114



# Agents

- *"Let us define an agent as a persistent software entity dedicated to a specific purpose. 'Persistent' distinguishes agents from subroutines; agents have their own ideas about how to accomplish tasks, their own agendas. 'Special purpose' distinguishes them from entire multifunction applications; agents are typically much smaller."*
  - *Smith, D. C., A. Cypher and J. Spohrer (1994), KidSim: Programming Agents Without a Programming Language, Communications of the ACM, 37, 7, 55-67*
- *"Autonomous agents are systems capable of autonomous, purposeful action in the real world."*
  - *Brustoloni, Jose C. (1991), Autonomous Agents: Characterization and Requirements, Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University*

# Parameters

- $V[t+1] = m \cdot V[t] +$  // Inertia term  
+  $c_1 \cdot \text{rand}[0,1] \cdot (\text{myBest}[] - \text{Position}[t]) +$  // Cognitive component  
+  $c_2 \cdot \text{rand}[0,1] \cdot (\text{globalBest}[] - \text{Position}[t])$  // Social component
- $m$  – inertia or momentum
- $c_1$  – personal (cognitive) acceleration coefficient
- $c_2$  – global (social) acceleration coefficient
- $\text{Position}[]$  – current position of the particle
- $V[]$  – current velocity vector
- $\text{myBest}[]$  – best position found by the particle
- $\text{globalBest}[]$  – best position found by any member of the flock
- The PSO parameters can also be tuned by using another overlaying optimizer, a concept known as meta-optimization, or even fine-tuned during the optimization, e.g., by means of fuzzy logic
- **P.S.: population amount and amount of iterations should be chosen too**



# Different PSO Types



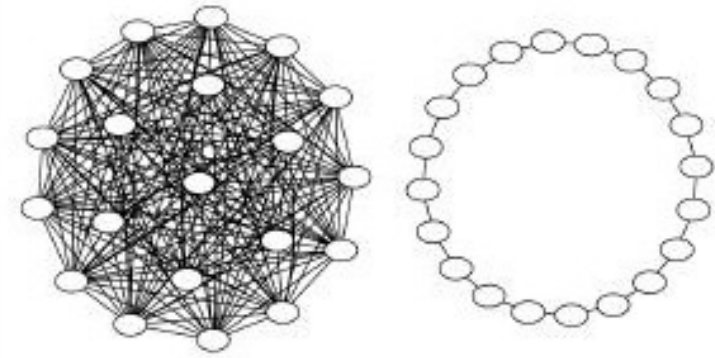
- gbest – global best
- lbest – local best
- Canonical PSO
  - Kenedy and Eberhart developed canonical PSO by altering velocity term into a probability threshold
- Inertia Weighted PSO
- Fully Informed Particle Swarm
- Time-Varying Inertia Weighted PSO
  - According to this PSO type inertia can be not only positive constant, it can be negative, also it can be changed during optimization
- etc.



# GBEST – Global Best

- Originally parameters have next values:
  - $m = 1$
  - $c_1 = 2$
  - $c_2 = 2$
- Parameters can be changed and need to be changed for better results

# LBEST – Local Best

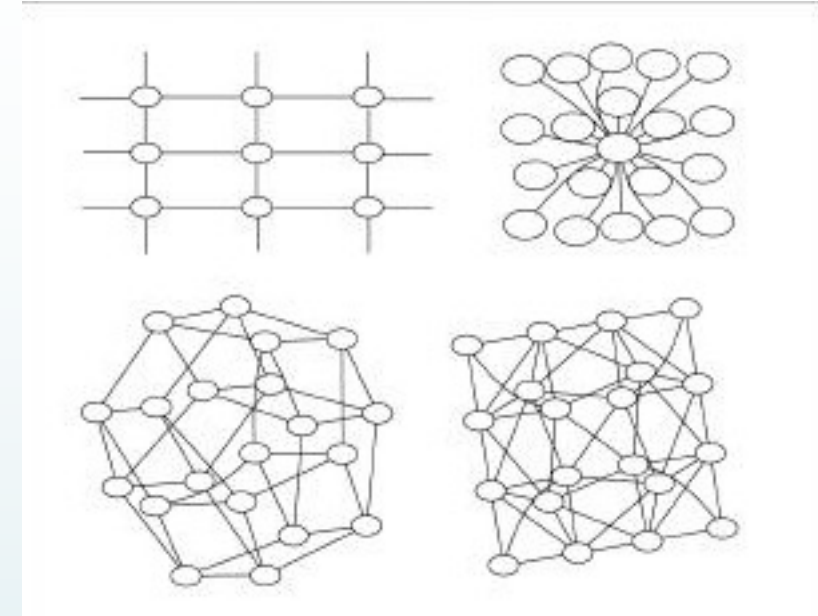


- In the GBEST swarm, all the particles are neighbors of each other. Thus, the position of the best overall particle in the swarm is used in the social term of the velocity update equation. It is assumed that GBEST swarms converge fast, as all the particles are attracted simultaneously to the best part of the search space. However, if the global optimum is not close to the best particle, it may be impossible to the swarm to explore other areas; this means that the swarm can be trapped in local optima
- In the LBEST swarm, only a specific number of particles (neighbor count) can affect the velocity of a given particle. The swarm will converge slower but can locate the global optimum with a greater chance



# Other PSO Topologies (1)

- The following additional topologies were tested:
  - Random
  - Von Neumann, a two dimensional grid with neighbors to the N, E, W and S
  - Pyramid, a three-dimensional triangular grid
  - Star, all the particles connected to a central particle
  - Heterogeneous, particles are grouped in several cliques

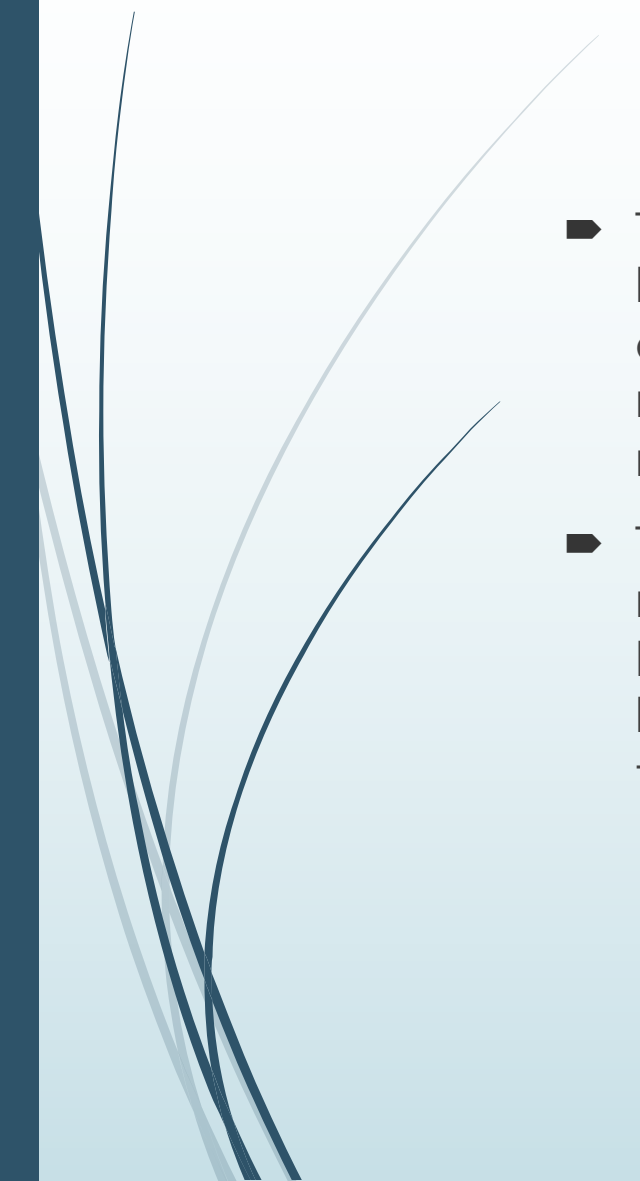


A flattened representation of the Von Neumann neighborhood (top left), three-dimensional representation of the Von Neumann neighborhood (bottom left), the star (top right) and the pyramid (bottom right) neighborhoods





## Other PSO Topologies (2)

- 
- The swarms with the different neighborhoods were used to optimize several known functions and an statistical analysis was carried out over several dependents: best result at 1000 iterations, number of iterations needed to meet the stopping criteria, and whether the stopping criteria was met or not
  - The conclusions depended on the variable selected, but a combined measure selected the Von Neumann and Pyramid neighborhoods as the best, and the star and gbest as the worst. Other conclusions were that higher  $k$  favored performance (good exploration), while lower  $k$  favored the chance of meeting the criteria (good exploitation)



# Ant Colony Optimization


# ACO



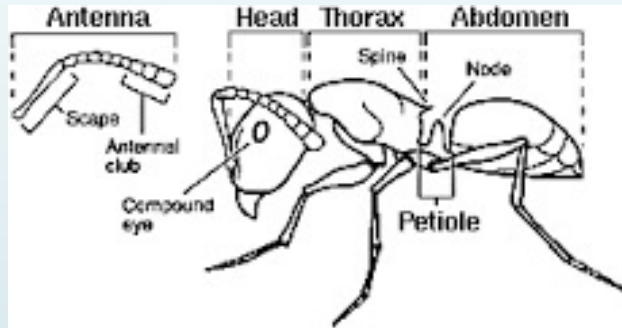
- Ant Colony Optimization (ACO) is a population-based, general search technique for the solution of difficult combinatorial problems, which is inspired by the pheromone trail laying behavior of real ant colonies
- In ACO, a set of software agents called artificial ants search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph
- ACO was initially proposed by Coloni, Dorigo and Maniezzo in 1992
- Ant colony optimization algorithms have been applied to many combinatorial optimization problems



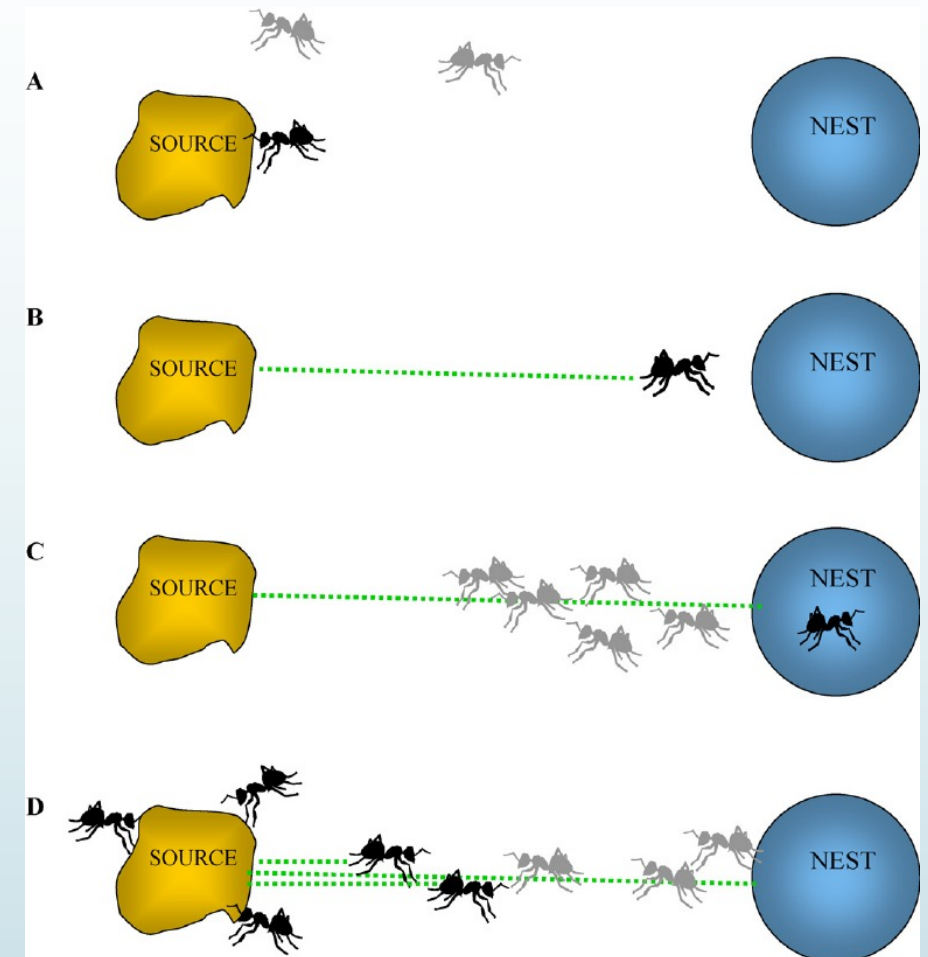
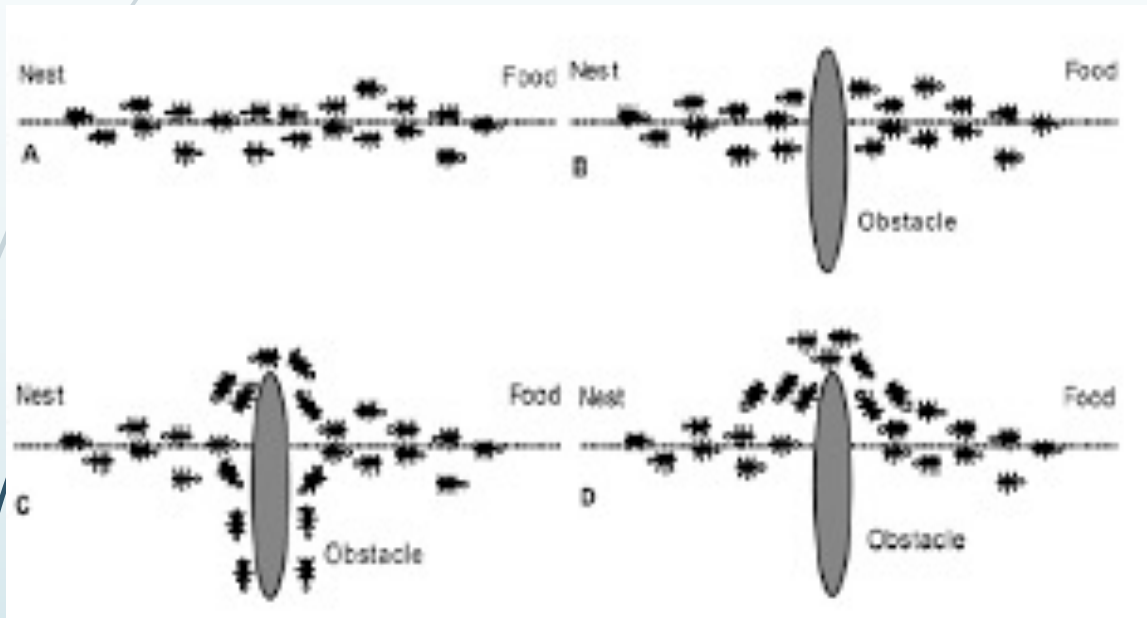
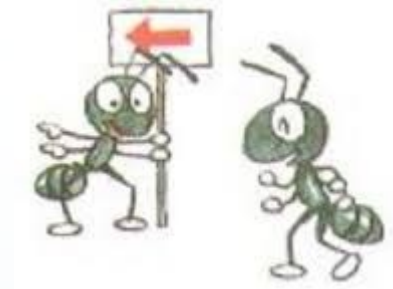
# ACO Advantages

- ▶ They have an advantage over simulated annealing and genetic algorithm approaches of similar problems when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time
- 

# How do ants forage for food?



# Trails of Smell - Pheromones





# Algorithm



- Step 1: Represent the solution space by a construction graph
- Step 2: Initialize ACO parameters
- Step 3: Generate random solutions from each ant's random walk
- Step 4: Update pheromone intensities
- Step 5: Go to Step 3, and repeat until convergence or a stopping condition is satisfied