Swarm Intelligence
(SI)

Swarm intelligence in nature (1/2)

Life in social groups easier:

- facilitates the retrieval of food
- reduces the probability of attack by predators
- allows for a division of labor
- facilitates hunting

Swarm intelligence in nature (2/2)

- Confusing predator by:
  - united erratic maneuvers (e.g. zigzagging)
  - separation into subgroups (e.g., flash expansion & fountain effect)
- Coordinated movements to trap prey: e.g. parabolic formation of tuna
- Fish schooling increases hydrodynamic efficiency:
  - endurance may be increased up to 6×
  - school acts like "group-level vehicle"
- V-formation increases aerodynamic efficiency of geese:
  - Range 70% greater than that of individual
- Lobsters line up in single file by touch:
  - move 40% faster than when isolated
  - decreased hydrodynamic drag

Swarm Intelligence

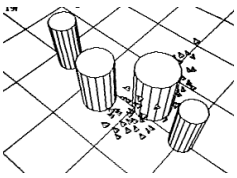
- Swarm intelligence studies the complex global behavior shown by a distributed system that emerges from the self-organized local interactions between its constituent agents
- Swarm intelligence systems are typically made up of a population of simple agents interacting locally with one another and with their environment
- Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global behavior

Particle Swarm Optimization

Early study: Reynolds' model (1/3)

- In 1986, Craig Reynolds made a computer model of coordinated animal motion: the simulated flock is an elaboration of a particle system, with the simulated birds being the particles.

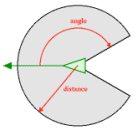
The aggregate motion of the simulated flock is the result of the dense interaction of the relatively simple behaviors of the individual simulated birds.



http://www.red3d.com/cwr/papers/1987/boids.html

Reynolds' model (2/3)

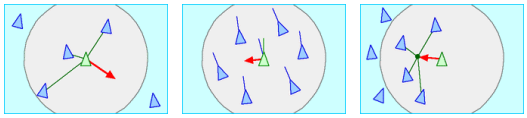
- An individual flocking creature (also called boid) maneuvers are based on the positions and velocities of flockmates within a certain small neighborhood around itself.
- The neighborhood is characterized by a *distance*, measured from the center of the boid, and an *angle*, measured from the boid's direction of flight.



A boyd's neighborhood (grey area)

Reynolds' model (3/3)

Reynolds' model involved 3 simple steering behaviors of boids: separation, alignment and cohesion.



Separation

avoid crowding local flockmates

Alignment

move towards the average heading of local flockmates

Cohesion

move towards the average position of local flockmates

https://www.youtube.com/watch?v=86iQiV3-3IA

Particle Swarm Optimization (PSO)

- In 1995 Kennedy and Eberhart developed a realistic visual simulation of bird flock behaviour and included a point in the environment that attracted the virtual bird agents, e.g. food.
- Characteristics:
  - Simple and quasi-identical units, called particles
  - 'Intermediate' number of units: 10 to 1000 (or more)
  - Each particle is described by a velocity and a position

Kennedy, J.; Eberhart, R.; , "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE International Conference on* , vol.4, no., pp.1942-1948 vol.4, Nov/Dec 1995.

Kennedy, Jim, and R. Eberhart. "Tutorial on particle swarm optimization." *2002 World Congress on Computational Intelligence WCCI*. 2002.

The (canonical) PSO algorithm

- Initialize the swarm (positions and velocities) and the parameters
- While not terminate: For each particle (for all members in the swarm) $i = 1 \ldots n$
  - Create random vectors r_1, r_2 with components in $U[0,1]$
  - update velocities

$$v_i \leftarrow \omega v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{g} - x_i)$$

$\circ$: componentwise multiplication

  - update positions

$$x_i \leftarrow x_i + v_i$$

  - update local bests (for a minimisation problem) $\hat{x}_i \leftarrow x_i$ if $f(x_i) < f(\hat{x}_i)$
  - update global best (for a minimisation problem) $\hat{g} \leftarrow x_i$ if $f(x_i) < f(\hat{g})$

The parameters $\omega, \alpha_1, \alpha_2$ of PSO should be chosen properly in order to balance between exploration and exploitation.

Initialization

- Initialize particle positions

$$X = \text{lower}_{\text{limit}} + (\text{upper}_{\text{limit}} - \text{lower}_{\text{limit}}) \times \text{rand}(n_{\text{particles}}, m_{\text{dimensions}})$$

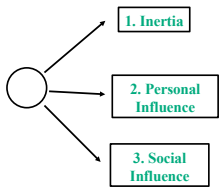
- Initialize velocities (all equal to 0)
- Initialize the global and local fitness to the worst possible
- Initialize parameters $n_{particles}, \omega, \alpha_1, \alpha_2$ (values used in practice: $\omega \sim 0.73, \alpha_1 = \alpha_2 \sim 1.5$)

Velocity update

Velocity of particle i:

$$v_i \leftarrow \omega v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{g} - x_i)$$

Inertia parameter | Local best: personal influence | Global best: 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
- Conservative
- Makes the particle follow the overall best direction

Inertia weight ω

$$v_i \leftarrow \boxed{\omega} v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{g} - x_i)$$

- A large inertia weight favors the diversification of the search process while a small inertia weight favors its intensification
- ω influences convergence of the swarm. $\omega > 1$: velocities increase over time causing divergent behavior, so that particles fail to change direction in order to move back towards promising areas; $\omega < 1$: particles may decelerate until their velocities reach zero

Eberhart, R. C., Shi, Y.: Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, Proceedings of the IEEE Congress on Evolutionary Computation , San Diego, USA, 2000.

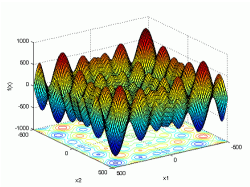
van den Bergh, F.: An Analysis of Particle Swarm Optimizers , Ph.D. Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

Parameters, Conditions and Tweaks

- Initialization methods
- Population size
- Neighborhood topology (used to compute global best)
- Sub-swarms
- Births, deaths, migration

Kennedy, Jim, and R. Eberhart. "Tutorial on particle swarm optimization." *2002 World Congress on Computational Intelligence WCCI*. 2002.

Example: a benchmark function



$n = 2$

$$f(x) = \sum_{i=1}^{n} (-x_i) \cdot \sin(\sqrt{|x_i|})$$

where

$$-500 \le x_i \le 500$$

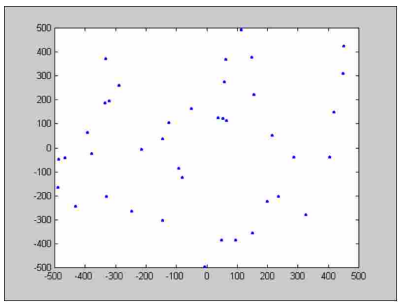
global minimum

$$f(x) = n \cdot 418.9829;$$

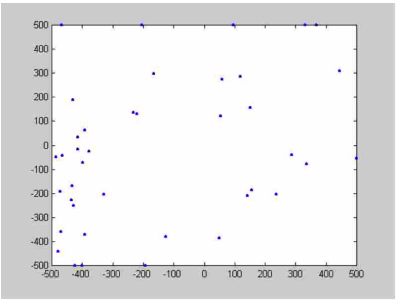
$$x_i = -420.9687, \mathrm{i} = 1:\mathrm{n}$$

Here n denotes the input dimension

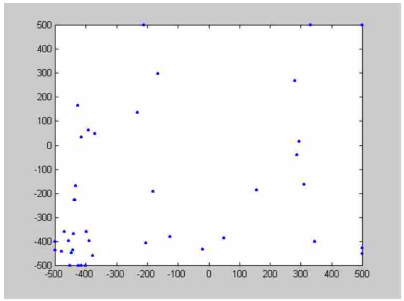
A PSO run (2-dimensional input)



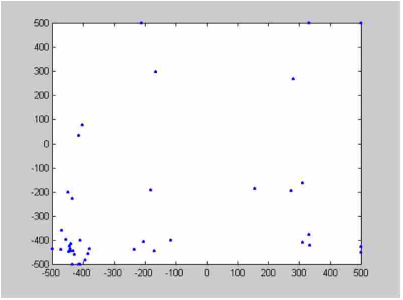
Evolution — 5th iteration



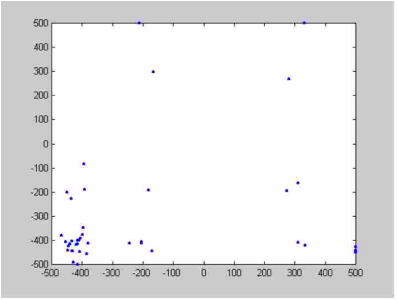
Evolution — 10th iteration



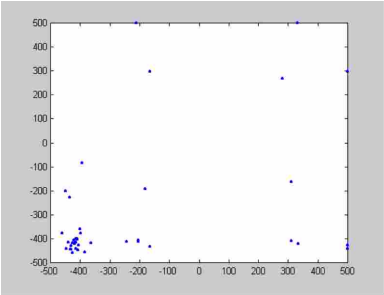
Evolution – 15th iteration



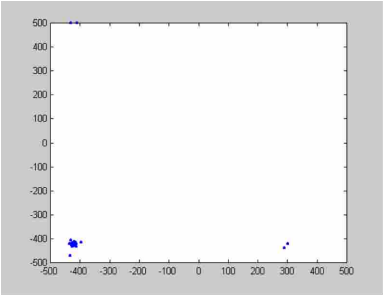
Evolution – 20th iteration



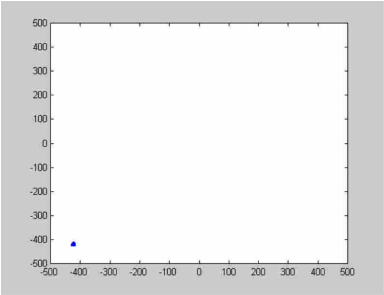
Evolution – 25th iteration



Evolution – 100th iteration

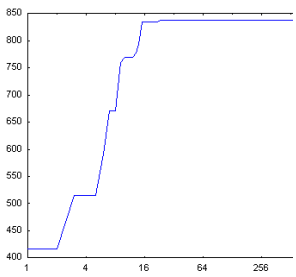


Evolution – 500th iteration



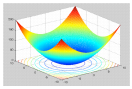
Search result

Iteration	Swarm best
0	416.245599
5	515.748796
10	759.404006
15	793.732019
20	834.813763
100	837.911535
5000	837.965771
Global	837.9658



Other benchmark functions

http://www.cil.pku.edu.cn/resources/benchmark_pso/



1) Sphere Function

$$f(x)=\sum_{i=1}^{n} x_i^2, x\in[-5,5]^n$$

2) Rosenbrock Function

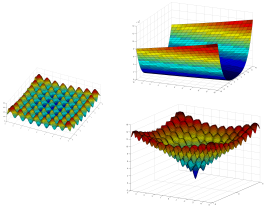
$$f(x)=\sum_{i=1}^{n-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right), x\in[-10,10]^n$$

3) Rastrigin Function

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

4) Ackley Function

$$f(x)=20+e-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right), x\in[-32,32]^n$$



Convergence behavior

- Failure: Swarm diverges or remains itinerant
- Optimally: Global best approaches global optimum (swarm may still oscillate)
- Typically: Global best approaches a local optimum (premature collapse of the swarm)

Mathematical attempts (typically oversimplified): Convergence to global optimum for a 1-particle swarm after infinite time (F.v.d. Bergh, 2001)
Empirically $\omega = 0.7298$ and acceleration coefficients $\alpha_1 = \alpha_2 = 1.49618$ provide good convergence behavior.

van den Bergh, F.: An Analysis of Particle Swarm Optimizers , Ph.D. Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
M. Clerc & J. Kennedy (2002) The particle swarm Explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 7:1, 58-73.

26

PSO: pros and cons

- **Advantages**
  - Simple implementation
  - Easily to parallelize for concurrent processing
  - Derivative free
  - Few parameters
- **Disadvantages**
  - Tendency to fast and premature convergence in mid optimum points
  - Slow convergence in refined search stage (weak local search ability)

PSO vs GA (1/2)

Similar approach:

1. Random generation of an initial population
2. Calculation of a fitness value for each individual
3. Reproduction of the population based on fitness values
4. If requirements are met, then stop. Otherwise go back to 2

PSO vs GA (2/2)

Differences:

Individuals update rule:
- In GA: by genetic operators
- In PSO: particles modify themselves via their own velocity
- Memory of their personal best. Elitism by global best

Sharing of information:
- Mutual in GA. Population moves in groups to optimal areas
- One-way in PSO: Source of information is global best. All particles tend to converge to the best solution quickly

Representation
- GA: usually discrete
- PS: usually continuous

See www.swarmintelligence.org/tutorials.php

Example: clustering

Cluster a given dataset into N clusters using PSO (N is given).

- Which type of candidate solutions?
- Which fitness function?

A solution

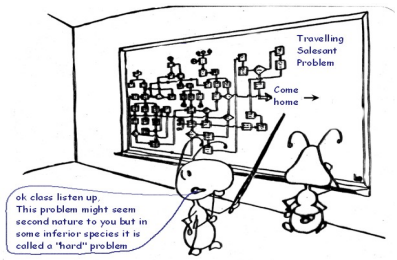
Each particle represents a list of N centroids: $x_i = (c_1^i, \dots, c_N^i)$

- Initialize each particle to contain N randomly selected centroids;
- For iter=1 to maxiter
  - For each particle $x_i = (c_1^i, \dots, c_N^i)$
    - For each data point z
      - Calculate the distance of z to each cluster centroid c_j^i
      - Assign z to the cluster (centroid) with minimal distance;
    - Compute the fitness of x_i (e.g. average weighted distance of data points from their centroid);
    - Update the local best;
  - Update the global best;
  - Update each particle x_i using PSO update rules:

$$v_i \leftarrow \omega v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{g} - x_i) \qquad x_i \leftarrow x_i + v_i$$

Van der Merwe, D. W., and Andries Petrus Engelbrecht. "Data clustering using particle swarm optimization." *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*. Vol. 1. IEEE, 2003.

Ant Colony Optimization (ACO)

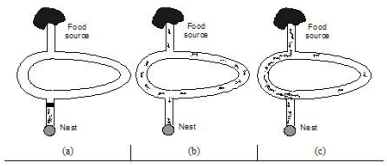


ACO in nature: an experiment (1/3)

(a) The bridge is initially closed

(b) Initial distribution of ants after the bridge is open

(c) Distribution of ants after some time has passed since they were allowed to exploit the food source



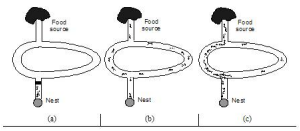
ACO in nature: an experiment (2/3)

- How does it work?
  - Ants place pheromone on their way
  - Routing depends on the concentration of pheromone
- More detailed:
  - Ants that use shorter bridge return faster
  - Pheromone concentration higher on shorter bridge
  - Ants choose shorter bridge more frequently than longer bridge
  - Pheromone concentration of shorter bridge even higher
  - Even more ants choose shorter bridge
  - A.s.f.

Positive feedback loop

ACO in nature: an experiment (3/3)

- Trail-laying by ants modifies the environment and provides indirect communication.
- At the branching point, ants make a `probabilistic' choice biased by the amount of pheromone they smell on the branches.
- This behavior has a self-reinforcing (positive feedback) effect, because choosing a branch will automatically increase the probability that this branch will be chosen again by other ants.



From real to artificial ant colonies

Biology (Ant Foraging)	ACO Algorithms
Ant	Individual (agent) used to build (construct) a solution to the problem
Ant colony	Population (colony) of cooperating individuals known as artificial ants
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 This allows the assessment of the quality of a given edge on a graph
Pheromone evaporation	Reduction in the pheromone level of a given path (edge) due to aging

ACO algorithms

- CO: combinatorial optimization
  S: *search space*, f: *function to optimize*, Ω: *constraints*, $\mathcal{T}$: *pheromone state*

Algorithm 1 The framework of a basic ACO algorithm

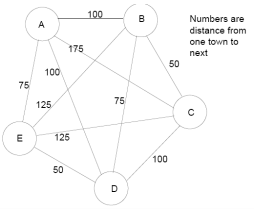
```
input: An instance P of a CO problem model 𝒫 = (𝒮, f, Ω).
InitializePheromoneValues(𝒯)
s_bs ← NULL                                   init best-so-far solution
while termination conditions not met do
  𝔖_iter ← ∅                                  set of valid solutions
  for j = 1, ..., n_a do                      loop over ants
    s ← ConstructSolution(𝒯)
    if s is a valid solution then
      s ← LocalSearch(s)        {optional}
      if (f(s) < f(s_bs)) or (s_bs = NULL) then s_bs ← s   update best-so-far
      𝔖_iter ← 𝔖_iter ∪ {s}                    store valid solutions
    end if
  end for
  ApplyPheromoneUpdate(𝒯, 𝔖_iter, s_bs)
end while
output: The best-so-far solution s_bs
```


ConstructSolutions

- Expand partial solutions by iteratively adding components.
- Selecting a component is a stochastic operation biased by the current pheromone associated with each component.
- Constructing solutions can be viewed as a walk on the so-called construction graph: nodes are solution components, weights are pheromones (they change during execution)

ConstructSolution on TSP

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city once and returns to the origin city?



ConstructSolution on TSP

- Each ant builds its own tour from a starting city
- Each ant chooses a next city to go to with a probability that depends on the city's distance and the amount of pheromone on the connecting edge

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in \mathbf{N}(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_{ij} \in \mathbf{N}(s^p), \\ 0 & \text{otherwise,} \end{cases}$$

- Legal tours are enforced: choosing already visited cities disallowed (keep a tabu list)

Transition probabilities p_{ij}^k

p_{ij}^k: probability of ant k to select component j from component i

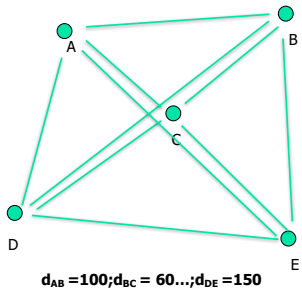
$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in \mathbf{N}(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_{ij} \in \mathbf{N}(s^p), \\ 0 & \text{otherwise,} \end{cases}$$

$$\eta_{ij} = \frac{1}{d_{ij}}$$

distance between cities i and j

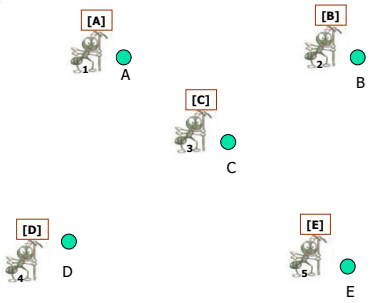
- α=0: choose closest cities more likely (search resembles stochastic greedy search).
- β=0: only pheromone amplification, favors stagnation (all ants follow the same path)

A toy example simulation



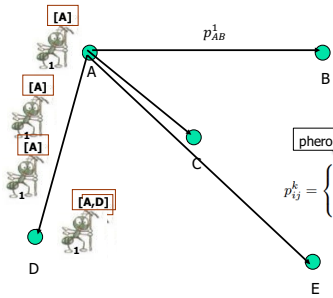
d_{AB} =100;d_{BC} = 60…;d_{DE} =150

Initialization



Select next solution component

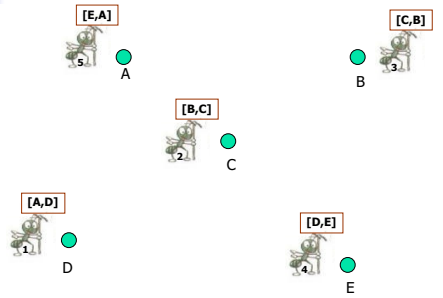
Probabilistic selection



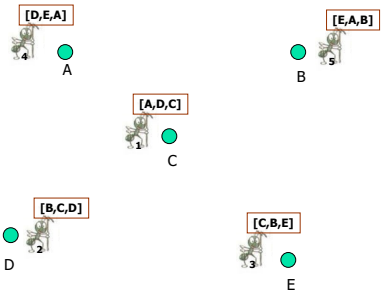
pheromone heuristic

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in \mathbf{N}(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_{ij} \in \mathbf{N}(s^p), \\ 0 & \text{otherwise,} \end{cases}$$

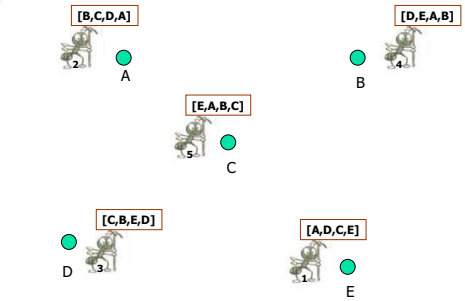
Select next solution component



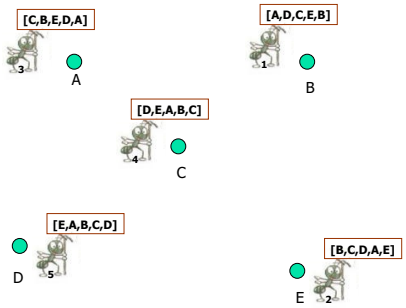
Select next solution component



Select next solution component



ConstructAntSolutions completed



ApplyLocalSearch

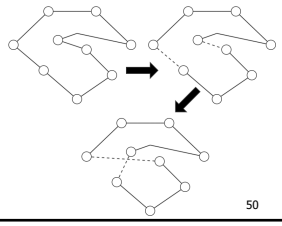
- Once candidate solutions have been constructed, and before updating the pheromone, they can be improved through local search (an optional, problem-dependent, step)

2-opt Local search for TSP

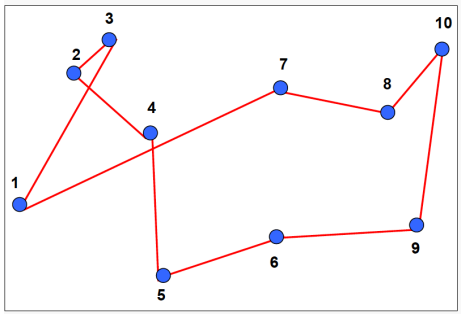
After building a tour ACO applies local search, e.g. the 2-opt:

- Two TSP tours are called 2-adjacent if one can be obtained from the other by deleting two edges and adding two edges.
- A TSP tour T is called 2-optimal if there is no 2-adjacent tour to T with lower cost than T.
- 2-opt local search:

Look for a 2-adjacent tour with lower cost than the current tour. If one is found, then it replaces the current tour. This continues until a 2-optimal tour is obtained.

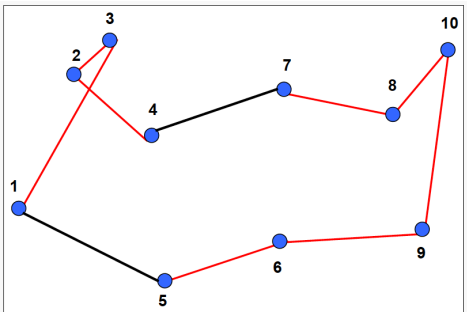


2-opt Example



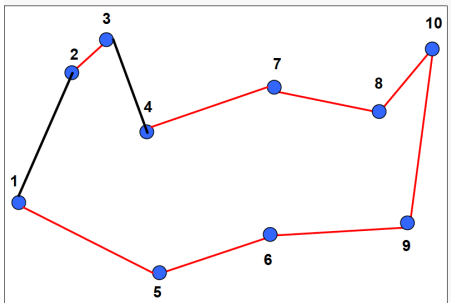
T
1
3
2
4
5
6
9
10
8
7

2-opt



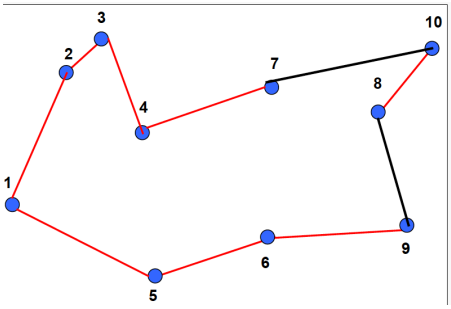
T_1	T_2
1	1
3	3
2	2
4	4
5	7
6	8
9	10
10	9
8	6
7	5

2-opt



T_2	T_3
1	1
3	2
2	3
4	4
7	7
8	8
10	10
9	9
6	6
5	5

2-opt



T_3	T_4
1	1
2	2
3	3
4	4
7	7
8	10
10	8
9	9
6	6
5	5

UpdatePheromones

Pheromone update:

- decrease all the pheromone values through pheromone evaporation, and
- increase the pheromone levels associated with components of good solutions

ACO algorithms mainly differ in the pheromone update rule they use

ACO algorithms

Algorithm	*Authors*	*Year*
Ant System (AS)	Dorigo et al.	1991
Elitist AS	Dorigo et al.	1992
Ant-Q	Gambardella & Dorigo	1995
Ant Colony System	Dorigo & Gambardella	1996
$\mathcal{MAX}$–$\mathcal{MIN}$ AS	Stützle & Hoos	1996
Rank-based AS	Bullnheimer et al.	1997
ANTS	Maniezzo	1999
BWAS	Cordón et al.	2000
Hyper-cube AS	Blum et al.	2001

Dorigo, Marco, and Mauro Birattari. "Ant colony optimization." Encyclopedia of machine learning. Springer, Boston, MA, 2011. 36-39.

Ant System (AS)

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

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L_k} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

- L_k length of tour built by ant k
- ρ: evaporation rate, values in (0,1], used to forget (bad decisions)

Max-Min Ant System (MMAS)

- Values of pheromone are limited τ_{min}, τ_{max} (truncation)

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

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/L_{best} & \textit{if ij in best tour} \\ 0 & \textit{otherwise} \end{cases}$$

L_{best} length of the tour of the best ant (two variants: iteration-best or best-so-far)

Ant Colony System (ACS)

Initial values $\tau_{ij} = \tau_0 \ll 1$

Local and global pheromone update:

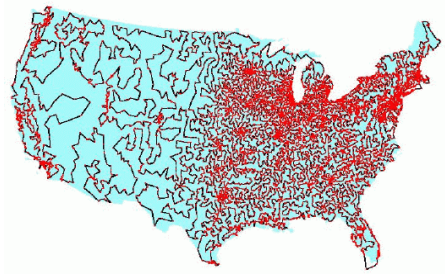
- local: $\tau_{ij} = \varphi\tau_{ij} + (1-\varphi)\tau_0$ each time *ij* traversed by an ant during solution construction (favors exploration for next ants)
- global: $\tau_{ij} = \rho\tau_{ij} + (1-\rho)\Delta\tau_{ij}^{best}$

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

Pseudorandom proportional transition rule: with probability q_0 select component *j* from *i* that maximizes $\tau_{ij}\eta_{ij}$ otherwise use standard probabilitistic rule as in AS.

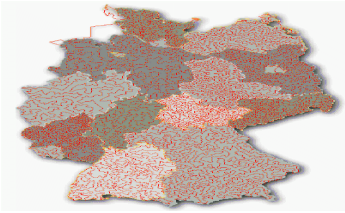
Real life TSP benchmark instances (1/2)

USA Towns of 500 or more population: 13509 cities (1998 Applegate, Bixby, Chvátal and Cook)



Real life TSP benchmark instances (2/2)

Towns in Germany: 15112 cities
(2001Applegate, Bixby, Chvátal and Cook)



TSP benchmark instances
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

61

Convergence

- Convergence in value: probability of finding at least once an optimal solution greater than $(1-\epsilon)$ for any small constant $\epsilon > 0$ if sufficiently large number of iterations. Holds for all ACO's with $\tau_{min} > 0$ (each feasible solution has non-zero probability to be constructed)
- Convergence in solution: the ACO generates the optimal solution over and over. Stronger than convergence in value holds under restrictive assumptions

Stutzle, Thomas, and Marco Dorigo. "A short convergence proof for a class of ant colony optimization algorithms." *IEEE Transactions on evolutionary computation* 6.4 (2002): 358-365.

Gutjahr, Walter J. "ACO algorithms with guaranteed convergence to the optimal solution." *Information processing letters* 82.3 (2002): 145-153.

62

Search bias in ACO

- Induced selection bias: some solution components occur in a larger number of admissible solutions (in terms of sequences built by the algorithm).
- An ACO and a problem instance are Competition-Balanced System (CBS) if each (solution) component occurs in the same number of feasible solutions as any other component.
- CBS implies fair competition between components.
- ACO for TSP is a CBS.

Blum, Christian, and Marco Dorigo. "Search bias in ant colony optimization: On the role of competition-balanced systems." *IEEE Transactions on Evolutionary Computation* 9.2 (2005): 159-174.

63

ACO: pros and cons

- Pros:
  - Inherent parallelism.
  - Positive feedback accounts for rapid discovery of good solutions.
  - Effective for ordering and grouping problems.
  - Can be used also in dynamic optimization applications (it adapts to changes such as new distances, etc).
- Cons:
  - Theoretical analysis is difficult: sequences of random decisions (not independent), probability distribution changes by iteration, time to convergence uncertain.

64

ACO versus PSO

Criteria	ACO	PSO
Communication Mechanism	ACO uses an indirect communication mechanism among ants, called stigmergy, which means interaction through the environment.	The communication among particles in PSO is rather direct without altering the environment.
Problem Types	ACO was originally used to solve combinatorial (discrete) optimization problems, but it was later modified to adapt continuous problems.	PSO was originally used to solve continuous problems, but it was later modified to adapt binary/ discrete optimization problems.
Problem Representation	ACO's solution space is typically represented as a weighted graph, called construction graph.	PSO's solution space is typically represented as a set of n-dimensional points.
Algorithm Applicability	ACO is commonly more applicable to problems where source and destination are predefined and specific.	PSO is commonly more applicable to problems where previous and next particle positions at each point are clear and uniquely defined.
Algorithm Objective	ACO's objective is generally searching for an optimal path in the construction graph.	PSO's objective is generally finding the location of an optimal point in a Cartesian coordinate system.
Examples of Algorithm Applications	Sequential ordering [32], scheduling [33], assembly line balancing [34], probabilistic TSP [35], DNA sequencing [36], 2D-HP protein folding [37], and protein–ligand docking [38].	Track dynamic systems [60], evolve NN weights [61], analyze human tremor [62], register 3D-to-3D biomedical image [63], control reactive power and voltage [64], and even play games [65].

Swarm Intelligence: *Concepts*, *Models* and *Applications*. Technical Report 2012-585. Hazem Ahmed. Janice Glasgow. School of Computing. Queen's University.

65

Ideas for your project

- PSO: DNN architectures, e.g. Junior, Francisco Erivaldo Fernandes, and Gary G. Yen. "Particle swarm optimization of deep neural networks architectures for image classification." *Swarm and Evolutionary Computation* 49 (2019): 62-74.
- ACO: Analyze ACO extensions in the context of a real-life application or over benchmark datastes, e.g. recent survey by Dorigo, M., & Stützle, T. (2019). Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics* (pp. 311-351). Springer, Cham.

- Next time (3/3): Cellular Automata
- Deadline assignment on SI: March 2, 13:30

67