Swarm Intelligence

Christian Blum

Artificial Intelligence Research Institute (IIIA)
Spanish National Research Council (CSIC)

© Alex Wild (http://www.myrmecos.net)

Outline

Topics:

- **Swarm intelligence:** Short introduction
- **Topic 1:** Ant colony optimization
  - ⋆ **Inspiration:** Foraging behavior of ant colonies
- **Topic 2:** Particle swarm optimization
  - ⋆ **Inspiration:** Social behavior of flocks of birds and fish schools

Outline

Topics (continued):

- **Topic 3:** Self-synchronized duty cycling in sensor networks
  - ⋆ **Inspiration:** Self-synchronization in ant colonies
- **Topic 4:** Distributed graph coloring
  - ⋆ **Inspiration:** Self-desynchronization of Japanese tree frogs

Swarm Intelligence

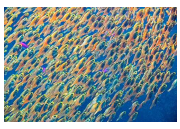
Short introduction

What is swarm intelligence

In a nutshell: AI discipline whose goal is designing intelligent multi-agent systems by taking inspiration from the collective behaviour of animal societies such as ant colonies, flocks of birds, or fish schools

© Ralf Müller

By curtesy of www.aerospaceweb.org

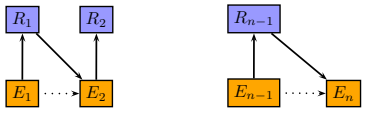


© Su Neko

Swarm intelligence

Properties:

- Consist of a set of simple entities
- Distributedness: No global control
- Self-organization by:
  - ⋆ **Direct communication:** visual, or chemical contact
  - ⋆ **Indirect communication:** Stigmergy (Grassé, 1959)



Result: Complex tasks/behaviors can be accomplished/exhibited in cooperation

Swarm intelligence

Examples of social insects:

- Ants
- Termites
- Some wasps and bees

© Velo Steve

© Alex Wild (http://www.myrmecos.net)

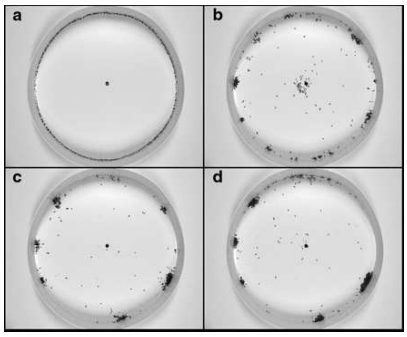
© Alex Wild

Current state: distributed optimization/control and robotics

Examples:

- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

Current state: Cemetery formation (1)



© by the National Academy of Sciences (PNAS)

Current state: Cemetery formation (2)

Note: Models for cemetery formation (brood tending) are used for clustering

- E. D. Lumer and B. Faieta. **Diversity and adaptation in populations of clustering ants**. *3rd International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3 (SAB 94)*, 1994
- D. Merkle, M. Middendorf, A. Scheidler. **Decentralized packet clustering in router-based networks**. *Int. J. Found. Comput. Sci.*, 2005.
- R. Klazar, A. P. Engelbrecht. **Dynamic Load Balancing Inspired by Cemetery Formation in Ant Colonies**. *Swarm Intelligence*, 2012.

Current state: distributed optimization/control and robotics

Examples:

- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

Current state: Division of Labour / Task Allocation (1)

- Problem: in any colony (ants, bees, etc) are a number of tasks to fulfill
- Examples: brood tending, foraging for resources, maintaining the nest
- Requires: dyanamic allocation of individuals to tasks
- Depends on: state of the environment, needs of the colony
- Requires: global assessment of the colonies current state

However: Individuals are unable (as individuals) to make a global assessment

Solution: Response threshold models

Division of Labour / Task Allocation (2)

Assume that:

- We have m tasks to fulfill
- We have n individuals in the colony
- Each individual i has a response threshold δ_{ij} for each task j
- Let $s_j \geq 0$ be the stimulus of task j
- An individual engages in task j with probability

$$p_{ij} = \frac{s_j^2}{s_j^2 + \delta_{ij}^2}$$

This means:

- If $s_j << \delta_{ij}$: p_{ij} is close to 0
- If $s_j >> \delta_{ij}$: p_{ij} is close to 1

Division of Labour / Task Allocation (3)

This means (continued):

- If $s_j = \delta_{ij}$: $p_{ij} = 0.5$
- An individual i with a low δ_{ij} is likely to respond to a lower stimulus s_j

Additional feature: response thresholds are dynamic

- Let Δt be a duration of time.
- Let $x_{ij}\Delta t$ be the fraction of time spent by i on task j within Δt
- Then: $(1 - x_{ij})\Delta t$ is the time spent by i on other tasks

Response threshold update:

$$\delta_{ij} \rightarrow \delta_{ij} - \xi x_{ij}\Delta t + \rho(1 - x_{ij})\Delta t$$

Division of Labour / Task Allocation (4)

where:

- ξ is a reinforcement coefficient
- ρ is a forgetting coefficient

Effects:

- The more an individual engages in a task j, the lower becomes its threshold
- The less an individual engages in a task j, the higher becomes its threshold

Current state: Division of Labour / Task Allocation (4)

References:

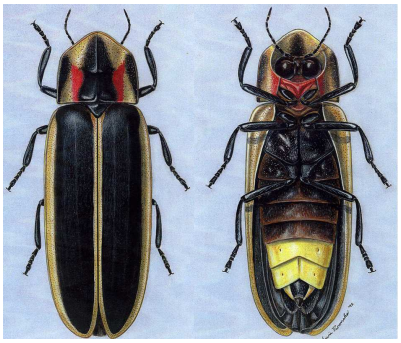
- M. Campos, E. Bonabeau, G. Theraulaz, and J.-L. Deneubourg. **Dynamic scheduling and division of labor in social insects**. *Adaptive Behavior*, 2000.
- D. Merkle, M. Middendorf and A. Scheidler. **Self-Organized Task Allocation for Service Tasks in Computing Systems with Reconfigurable Components**, *Journal of Mathematical Modelling and Algorithms*, 2008.
- A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, M. Dorigo. **Self-organized task allocation to sequentially interdependent tasks in swarm robotics**. *Autonomous Agents and Multi-Agent Systems*, 2014.

Current state: distributed optimization/control and robotics

Examples:

- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

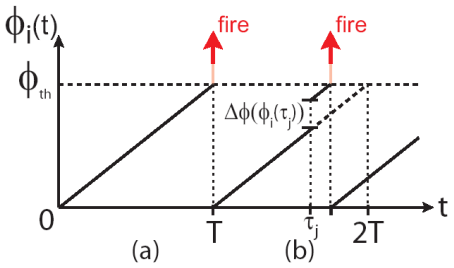
Current state: Self-synchronization of fireflies (1)



By curtesy of www.learner.org

Current state: Self-synchronization of fireflies (2)

Pulse-coupled oscillator model



© A. Tyrrell and G. Auer

Current state: Self-synchronization of fireflies (3)

References:

- A. Rowe, R. Mangharam and R. Rajkumar. **FireFly: A Time Synchronized Real-Time Sensor Networking Platform**, *Wireless Ad Hoc Networking: Personal-Area, Local-Area, and the Sensory-Area Networks*, CRC Press Book Chapter (2006)
- N. Lipa, E. Mannes, A. Santos and M. Nogueira. **Firefly-inspired and robust time synchronization for cognitive radio ad hoc networks**. *Computer Communications*, 2015.
- D. Sutantyo and P. Levi. **Decentralized underwater multi-robot communication using bio-inspired approaches**. *Artificial Life and Robotics*, 2015.

Current state: distributed optimization/control and robotics

Examples:

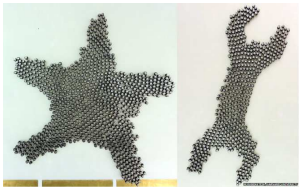
- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

Current state: Nest construction (1)

Termite mound **Ant hill**

Current state: Nest construction (2)

Automated construction **Automated shape forming**



- J. Werfel, K. Petersen, R. Nagpal. **Designing Collective Behavior in a Termite-Inspired Robot Construction Team**, Science, 2014.
- M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, R. Nagpal. **Kilobot: A Low Cost Robot with Scalable Operations Designed for Collective Behaviors.** *Robotics and Autonomous Systems Journal*, 2013.

Current state: distributed optimization/control and robotics

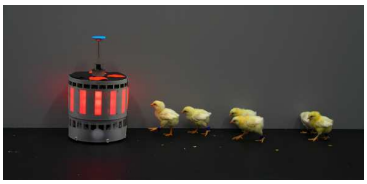
Examples:

- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

Current state: Animal-robot interaction (1)

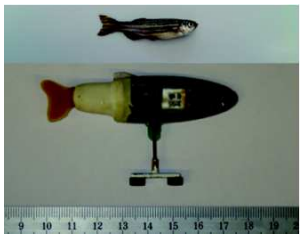
Reasons for controlling animal swarms:

1. Studies of animal behaviour. For example:
   - Studying swarm/herd/group formation
   - Studying collective decision making
2. Guiding a swarm out of a danger zone

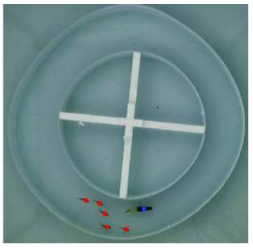


Current state: Animal-robot interaction (2)

Robot fish



Influencing the fish school



- F. Bonnet, Y. Kato, J. Halloy, F. Mondada. **Infiltrating the Zebrafish Swarm: Design, Implementation and Experimental Tests of a Miniature Robotic Fish Lure for Fish-Robot Interaction Studies**. *SWARM 2015: The First International Symposium on Swarm Behavior and Bio-Inspired Robotics*, 2015.

Current state: Animal-robot interaction (3)

- S. Gade, A. A. Paranjape, and S.-J. Chung. **a Flock of Birds Approaching an Airport Using an Unmanned Aerial Vehicle**. **AIAA Guidance, Navigation, and Control Conference**, 2015.

Swarm intelligence: examples

Examples:

- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

Flocking (1)

Definition: The collective motion of a large number of self-propolled entities

Note:

- Commonly used as a demonstration of emergence and self-organization
- Modelled/simulated for the first time by Craig Reynolds (Boids, 1986)

Model: Basic rules

1. Separation: avoid crowding neighbours (short range repulsion)
2. Alignment: steer towards average heading of neighbours
3. Cohesion: steer towards average position of neighbours (long range attraction)

Flocking (2)

References:

- J. Kennedy and R. Eberhart. **Particle Swarm Optimization**, *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995
- G. Folino, A. Forestiero and G. Spezzano. **An adaptive flocking algorithm for performing approximate clustering**, *Information Sciences*, 179(18):3059–3078, 2009
- X. Cui, J. Gao, and E. Potok. **A Flocking based algorithm for document clustering analysis**, *Journal of Systems Architecture*, 52, 505–515, 2006

Swarm intelligence: examples

Examples:

- Cemetery formation (ants)
- Division of labour / Task allocation (ants + bees)
- Self-synchronization of fireflies
- Nest construction (termites + ants)
- Animal-robot interaction
- Flocking (birds + fish)
- Foraging behavior of ants

Foraging behavior of ants (1)

Communication strategies:

- Direct communication: For example, recruitment
- Indirect communication: via chemical pheromone trails

© Alex Wild (http://www.myrmecos.net)

© Christian Blum

Foraging behavior of ants (2)

Communication strategies:

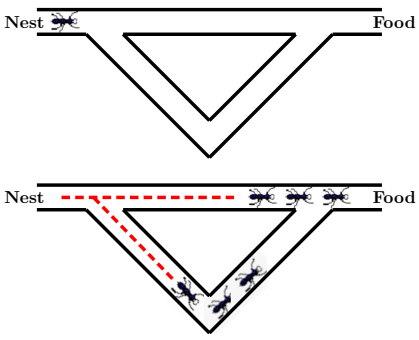
- Direct communication: For example, recruitment
- Indirect communication: via chemical pheromone trails

Basic behaviour:

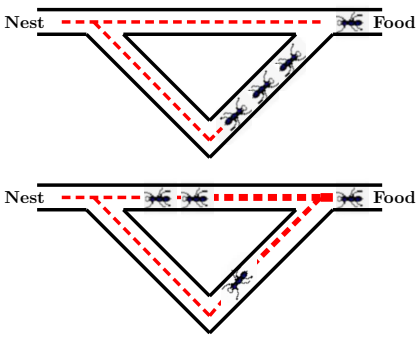
Foraging behavior of ants (3)

© Alex Wild (http://www.myrmecos.net)

Foraging behavior of ants (4)



Foraging behavior of ants (5)



Foraging behavior of ants (6)

References:

- M. Dorigo and T. Stützle. **Ant Colony Optimization**, *MIT press*, 2004.
- C. Blum. **Ant colony optimization: introduction and recent trends**, *Physics of Life Reviews*, 2(4):353–373, 2005.
- P. Korosec, J. Silc and B. Filipic. **The differential ant-stigmergy algorithm**, *Information Sciences*, 192, 82–97, 2012

Topic 1: Ant Colony Optimization

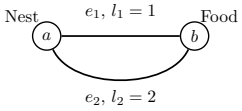
Inspiration: Foraging behavior of ant colonies

Outline (ACO part):

- Simulation of the foraging behaviour
- The ACO metaheuristic
- Example: traveling salesman problem (TSP)
- Example: assembly line balancing
- A closer look at algorithm components
- ACO for continuous optimization

Simulation of the foraging behaviour (1)

Technical simulation:



1. We introduce artificial pheromone parameters:

   $\mathcal{T}_1$ for e_1 and $\mathcal{T}_2$ for e_2

2. W initialize the phermomone values:

   $\tau_1 = \tau_2 = c > 0$

Simulation of the foraging behaviour (2)

Algorithm:

Iterate:

1. Place n_a ants in node a.
2. Each of the n_a ants traverses from a to b either
   - via e_1 with probability $\mathbf{p}_1 = \frac{\tau_1}{\tau_1+\tau_2}$,
   - or via e_2 with probability $\mathbf{p}_2 = 1 - \mathbf{p}_1$.
3. Evaporate the artificial pheromone: $i = 1, 2$

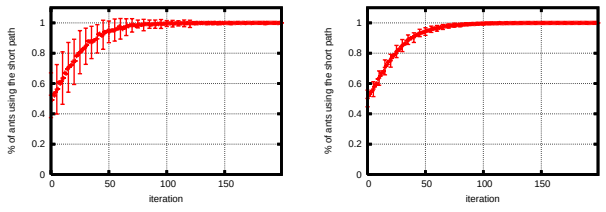
   $$\tau_i \leftarrow (1-\rho)\tau_i \ , \ \rho \in (0, 1]$$

4. Each ant leaves pheromone on its traversed edge e_i:

   $$\tau_i \leftarrow \tau_i + \frac{1}{l_i}$$

Simulation of the foraging behaviour (3)

Simulation results:



Colony size: 10 ants | Colony size 100 ants

Observation: Optimization capability is due to co-operation

Simulation of the foraging behaviour (4)

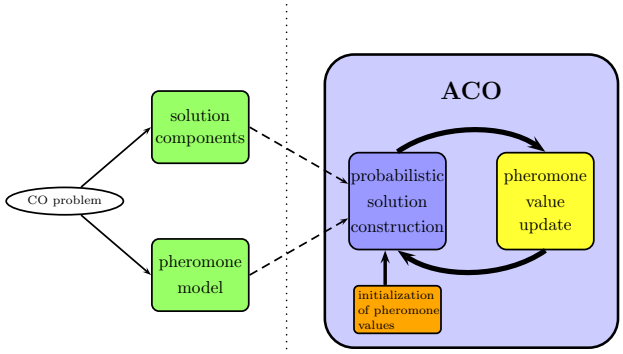
Main differences between model and reality:

	Real ants	Simulated ants
Ants' movement	asynchronous	synchronized
Pheromone laying	while moving	after the trip
Solution evaluation	implicitly	explicit quality measure

Problem: In combinatorial optimization we want to find good solutions

Outline (ACO part):

The ACO framework



The ACO pseudocode

input: An instance P of a combinatorial problem $\mathcal{P}$.
InitializePheromoneValues($\mathcal{T}$)
while termination conditions not met **do**
  $S_{iter} \leftarrow \emptyset$
  for $j = 1, \ldots, n_a$ **do**
    $s \leftarrow$ ConstructSolution($\mathcal{T}$)
    $s \leftarrow$ LocalSearch(s) — optional —
    $S_{iter} \leftarrow S_{iter} \cup \{s\}$
  end for
  ApplyPheromoneUpdate($\mathcal{T}$)
end while
output: The best solution found

Metaheuristics: Timeline of their introduction

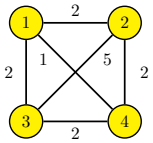
Metaheuristics:

- Simulated Annealing (SA) [Kirkpatrick, 1983]
- Tabu Search (TS) [Glover, 1986]
- Genetic and Evolutionary Computation (EC) [Goldberg, 1989]
- **Ant Colony Optimization (ACO) [Dorigo, 1992]**
- Greedy Randomized Adaptive Search Procedure (GRASP) [Resende, 1995]
- Particle Swarm Optimization (PSO) [Kennedy, 1995]
- Guided Local Search (GLS) [Voudouris, 1997]
- Iterated Local Search (ILS) [Stützle, 1999]
- Variable Neighborhood Search (VNS) [Mladenović, 1999]

Outline (ACO part):

TSP: definition (1)

Example: Traveling salesman problem (TSP). Given a completely connected, undirected graph $G = (V, E)$ with edge-weights.

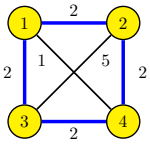


Goal: Find a tour (a Hamiltonian cycle) in G with minimal sum of edge weights.

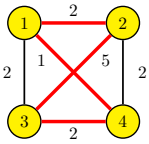
TSP definition (2)

TSP in terms of a combinatorial optimization problem $\mathcal{P} = (\mathcal{S}, f)$:

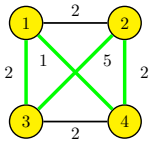
- $\mathcal{S}$ consists of all possible Hamiltonian cycles in G.
- Objetive function $f : \mathcal{S} \mapsto \mathbb{R}^+$: $s \in \mathcal{S}$ is defined as the sum of the edge-weights of the edges that are in s.



obj. function value: 8



obj. function value: 10



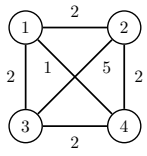
obj. function value: 10

Applying ACO to the TSP

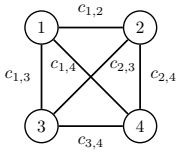
Preliminary step: Definition of the

- solution components
- pheromone model

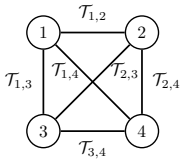
example instance



solution components



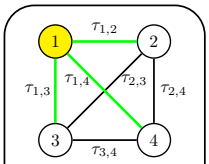
pheromone model



TSP: solution construction

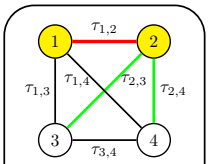
Tour construction:

Step 1



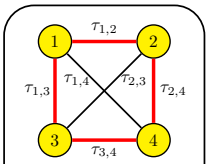
$$\mathbf{p}(c_{i,j}) = \frac{\tau_{i,j}}{\tau_{1,2} + \tau_{1,3} + \tau_{1,4}}$$

Step 2



$$\mathbf{p}(c_{i,j}) = \frac{\tau_{i,j}}{\tau_{2,3} + \tau_{2,4}}$$

Finished



TSP: pheromone update (1)

Pheromone update: For example with the Ant System (AS) update rule

Pheromone evaporation

$$\tau_{i,j} \leftarrow (1 - \rho) \cdot \tau_{i,j}$$

Reinforcement

$$\tau_{i,j} \leftarrow \tau_{i,j} + \rho \cdot \sum_{\{s \in S_{iter} | c_{i,j} \in s\}} F(s)$$

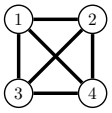
where

- evaporation rate $\rho \in (0, 1]$
- S_{iter} is the set of solutions generated in the current iteration
- quality function $F : S \mapsto \mathbb{R}^+$. We use $F(\cdot) = \frac{1}{f(\cdot)}$

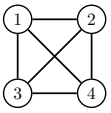
TSP: pheromone update (2)

Pheromone update: For example with the Ant System (AS) update rule

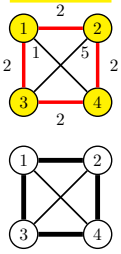
start



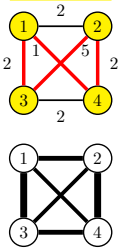
evaporation



solution s_1



solution s_2



Outline (ACO part):

- Simulation of the foraging behaviour
- The ACO metaheuristic
- Example: traveling salesman problem (TSP)
- Example: assembly line balancing
- A closer look at algorithm components
- ACO for continuous optimization

The ant colony optimization metaheuristic

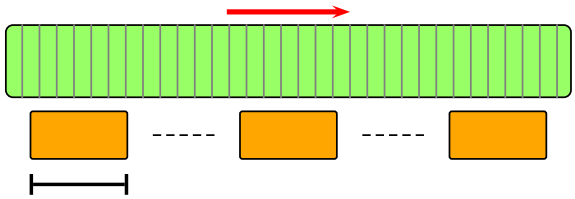
Assembly line balancing

© BMW

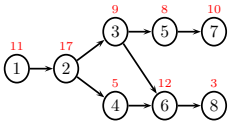
© J. Bautista & NISSAN

Specific problem: Simple assembly line balancing (SALB) [Bautista,Pereira,2004]

The ant colony optimization metaheuristic



Tasks: Each task i has a time requirement t_i



The ant colony optimization metaheuristic

Additionally given: The maximum number UB of possible work stations

Goal: Minimize the number of work stations needed!

1st step of applying ACO: Solution components and pheromone model

1. Solution components: We consider each possible assignment of
   - a task i
   - to a work station j

   to be a solution component $c_{i,j}$
2. Pheromone model: We assign to each solution component $c_{i,j}$ a pheromone trail parameter $\mathcal{T}_{i,j}$ with value $\tau_{i,j}$

The ant colony optimization metaheuristic

Solution construction: Work stations are filled with tasks one after the other

At each iteration:

- j^*: The current work station to be filled
- T: The set of tasks
  1. that are not yet assigned to a work station
  2. whose predecessors are all assigned to work stations
  3. whose time requirement is such that it fits into j^*

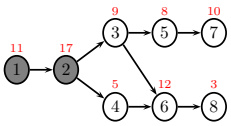
If T is empty: Open a new work station

If all tasks assigned: Stop solution construction

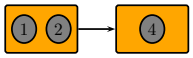
The ant colony optimization metaheuristic

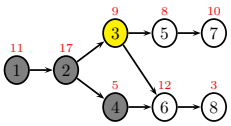
Assumption: Cycle time $C = 30$ seconds

Example situation 1:



Example situation 2:



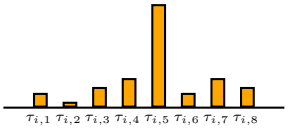


The ant colony optimization metaheuristic

At each iteration: How to choose a task from T?

$$\mathbf{p}(c_{i,j^*}) = \frac{\tau_{i,j^*}}{\sum_{k \in T} \tau_{k,j^*}} \qquad \forall\, i \in T$$

Disadvantage in this case:

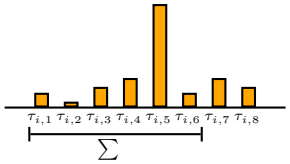


The ant colony optimization metaheuristic

Possible solution: The summation rule [Merkle et al., 2000]

$$\mathbf{p}(c_{i,j^*}) = \frac{\left(\sum_{h=1}^{j^*} \tau_{i,h}\right)}{\sum_{k \in T} \left(\sum_{h=1}^{j^*} \tau_{k,h}\right)} \qquad \forall\, i \in T$$

Graphical example: Current work station: 6



The ant colony optimization metaheuristic

Pheromone update: For example with the iteration-best (IB) update rule

Pheromone evaporation **Reinforcement**

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} \qquad \tau_{i,j} \leftarrow \tau_{i,j} + \rho \cdot F(s_{ib}) \qquad \forall\, c_{i,j} \in s_{ib}$$

where

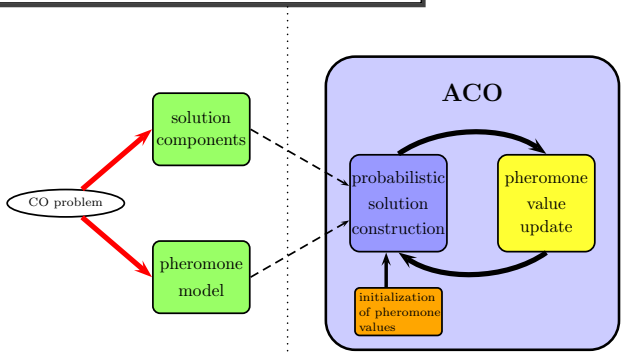
- evaporation rate $\rho \in (0, 1]$
- s_{ib} is the best solution constructed in the current iteration
- quality function $F : S \mapsto \mathbb{R}^+$. We use $F(\cdot) = \frac{1}{f(\cdot)}$

Outline (ACO part):

- Simulation of the foraging behaviour
- The ACO metaheuristic
- Example: traveling salesman problem (TSP)
- Example: assembly line balancing
- A closer look at algorithm components
- ACO for continuous optimization

The ant colony optimization metaheuristic

Definition of solution components and pheromone model



The ant colony optimization metaheuristic

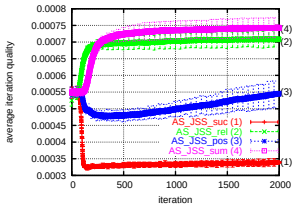
Note: Different pheromone models can be used to solve a problem!

Example: 3 different pheromone models for group shop scheduling

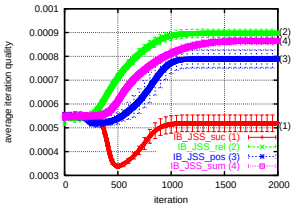
On the board!

The ant colony optimization metaheuristic

Result: Different pheromone models ⇒ different algorithm performance



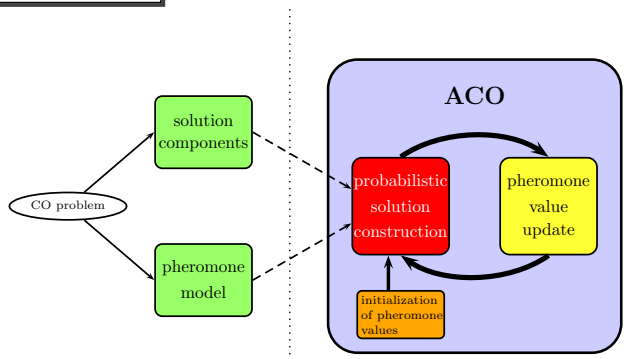
AS-update



IB-update

Solution construction (1)

Solution construction: A closer look



Solution construction (2)

A general constructive heuristic:

- $s^p = \langle\rangle$
- Determine $N(s^p)$
- **while** $N(s^p) \neq \emptyset$
  - $c \leftarrow$ ChooseFrom$(N(s^p))$
  - $s^p \leftarrow$ extend s^p by adding solution component c
  - Determine $N(s^p)$
- **end while**

Problem: How to implement function ChooseFrom$(N(s^p))$?

Solution construction (3)

Possibilities for implementing ChooseFrom$(N(s^p))$:

- Greedy algorithms:

$$c^* = \text{argmax}_{c_{i,j} \in N(s^p)} \eta(c_{i,j}) \ ,$$

where $\eta : C \mapsto \mathbb{R}^+$ is a Greedy function

Examples for Greedy functions:

- TSP: Inverse distance between nodes (i.e., cities)
- SALB: t_i/C

Solution construction (4)

Possibilities for implementing ChooseFrom$(N(s^p))$:

- Ant colony optimization:

$$\mathbf{p}(c_{i,j} \mid s^p) = \frac{[\tau_{i,j}]^\alpha \cdot [\eta(c_{i,j})]^\beta}{\sum_{c_{k,l} \in N(s^p)} [\tau_{k,l}]^\alpha \cdot [\eta(c_{k,l})]^\beta} \ , \quad \forall\, c_{i,j} \in N(s^p) \ ,$$

where α and β are positive values

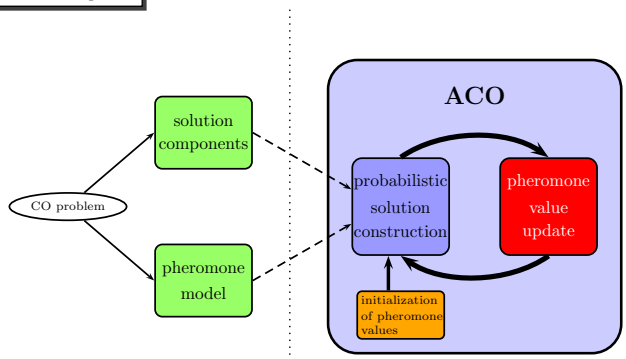
Note: α and β balance between pheromone information and Greedy function

Observations:

- ACO can be applied if a constructive heuristic exists!
- ACO can be seen as an iterative, adaptive Greedy algorithm

Pheromone update (1)

Pheromone update: A closer look



Pheromone update (2)

A general update rule:

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + \rho \cdot \sum_{\{s \in S_{upd} | c_{i,j} \in s\}} w_s \cdot F(s) \ ,$$

where

- evaporation rate $\rho \in (0,1]$
- S_{upd} is the set of solutions used for the update
- quality function $F : S \mapsto \mathbb{R}^+$. We use $F(\cdot) = \frac{1}{f(\cdot)}$
- w_s is the weight of solution s

Question: Which solutions should be used for updating?

Pheromone update (3)

ACO update variants:

AS-update	$S_{upd} \leftarrow S_{iter}$ weights: $w_s = 1 \ \forall \ s \in S_{upd}$
elitist AS-update	$S_{upd} \leftarrow S_{iter} \cup \{s_{bs}\}$ (s_{bs} is best found solution) weights: $w_s = 1 \ \forall \ s \in S_{iter}$, $w_{s_{bs}} = e \geq 1$
rank-based AS-update	$S_{upd} \leftarrow$ best $m-1$ solutions of $S_{iter} \cup \{s_{bs}\}$ (ranked) weights: $w_s = m - r$ for solutions from S_{iter}, $w_{s_{bs}} = m$
IB-update:	$S_{upd} \leftarrow \text{argmax}\{F(s) \mid s \in S_{iter}\}$ weight 1
BS-update:	$S_{upd} \leftarrow \{s_{bs}\}$ weight 1

The ant colony optimization metaheuristic

Successful ACO variant:

- $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System($\mathcal{MM}$AS) [Stützle, Hoos, 2000]

Characteristic properties:

- Use of a pheromone lower bound $\tau_{min} > 0$
- Application of restarts (by re-initializing the phermone values)
- Mix of IB-update and BS-update depending on a convergence measure

The ant colony optimization metaheuristic

Successful ACO variant:

- Ant Colony System(ACS) [Gambardella, Dorigo, 1996]

Characteristic properties:

- Deterministic construction steps with probability q

$$c = \text{argmax}_{c_{i,j} \in N(s^p)} [\tau_{i,j}]^\alpha \cdot [\eta(c_{i,j})]^\beta$$

- Evaporation of pheromone during the construction of solution s:

$$\tau_{i,j} \leftarrow \gamma \tau_{i,j} + (1-\gamma) c \ , \forall \ c_{i,j} \in s \ ,$$

  where $c > 0$ is the initial pheromone value, and $\gamma \in (0,1]$
- Use of the BS-update (evaporation only for used solution components)

The ant colony optimization metaheuristic

Successful ACO variant:

- The hyper-cube framework (HCF) for ACO [Blum, Dorigo, 2004]

Charactersitic properties:

Limits the pheromone values to the interval $[0,1]$ by using the folling update:

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + \rho \cdot \sum_{\{s \in S_{upd} | c_{i,j} \in s\}} \frac{F(s)}{\sum_{s' \in S_{upd}} F(s')}$$

The ant colony optimization metaheuristic

Rewriting the HCF update in vector form:

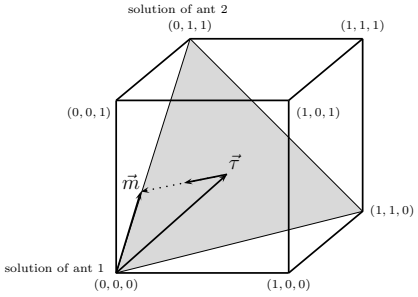
$$\vec{\tau} \leftarrow \vec{\tau} + \rho \cdot (\vec{m} - \vec{\tau}) \ ,$$

where $\vec{m}$ is a $|C|$-dimensional vector with

$$\vec{m} = \sum_{s \in S_{upd}} \gamma_s \cdot \vec{s} \quad \text{and} \quad \gamma_s = \frac{F(s)}{\sum_{s' \in S_{upd}} F(s')} \ .$$

The ant colony optimization metaheuristic

Example: Problem with 3 solutions, 2 ants per iteration



Theoretical studies of ant colony optimization

Search bias in ant colony optimization:

- **Positive (and wanted) bias:** Choice of (in comparison) good solutions for updating
- **Negative bias:**
  1. Modelling of the problem
  2. Solution construction process
  3. Pheromone update

How to detect negative bias? Decreasing algorithm performance over time

Theoretical studies of ant colony optimization

Implicit assumptions in ACO:

Assumption 1:

Good solutions are composed of good solution components.
(A solution component is regarded to be good, if the average quality of the solutions that contain it is high.)

Assumption 2:

The pheromone update is such that good solution components on average are stronger reinforced than others.

Theoretical studies of ant colony optimization

Implicit assumptions in ACO:

Assumption 1:

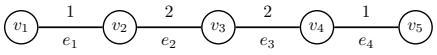
Good solutions are composed of good solution components.
(A solution component is regarded to be good, if the average quality of the solutions that contain it is high.)

Assumption 2:

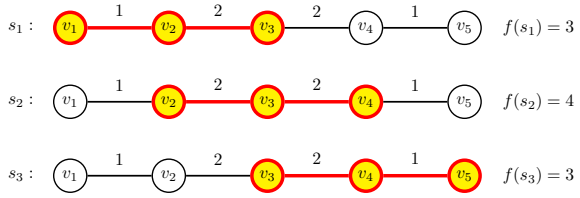
The pheromone update is such that good solution components on average are stronger reinforced than others.

Theoretical studies of ant colony optimization

Example: 2-cardinality tree problem

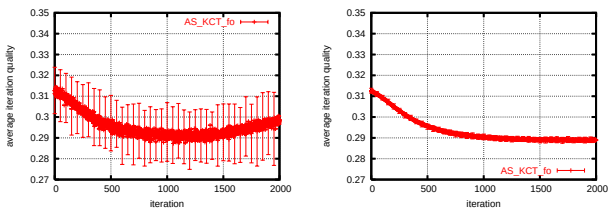


3 different solutions:



Theoretical studies of ant colony optimization

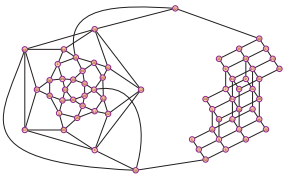
Average iteration quality of Ant System $\rho = 0.01$



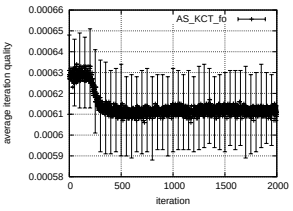
$n_a = 10$ $n_a = 1000$

Theoretical studies of ant colony optimization

Benchmark instances: Ant System applied to an Internet-like instance



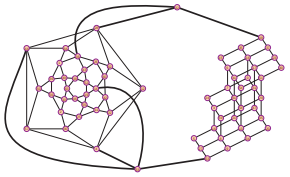
instance gd96c (65 nodes, 125 edges)



10 ants, $\rho = 0.1$, $k = 30$

Theoretical studies of ant colony optimization

Instance statistics:



Theoretical studies of ant colony optimization

Defintion: Competition-balanced system (CBS)

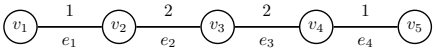
Given:

1. a feasible partial solution s^p;
2. and the set of solution components $N(s^p)$ that can be added to extend the partial solution s^p

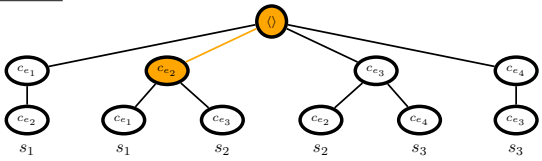
An ACO algorithm applied to $P \in \mathcal{P}$ is called a CBS, if each solution component $c \in N(s^p)$ is a component of the same number of feasible solutions.

Theoretical studies of ant colony optimization

Example: 2-cardinality tree problem



Search tree:



Therefore: This example is **NOT** a competition-balanced system

Theoretical studies of ant colony optimization

What do we know?

1. In case an ACO algorithm applied to a problem instance is **NOT** a competition-balanced system → possibility of negative search bias
2. Existing theoretical result: The Ant System algorithm applied to unconstrained problems does not suffer from negative search bias

Open questions:

1. Can it be shown that a competition-balanced system does not suffer from negative search bias?
2. ...

In general: Research on search bias might lead to better guidelines on how to develop ACO algorithms

Outline (ACO part):

Ant colony optimization for continuous optimization

Continuous optimization

Given:

1. Function $f : \mathbb{R}^n \mapsto \mathbb{R}$
2. Constrains such as, for example, $x_i \in [l_i, u_i]$

Goal: Find

$$\vec{X}^* = (x_1^*, \ldots, x_n^*) \in \mathbb{R}^n$$

such that

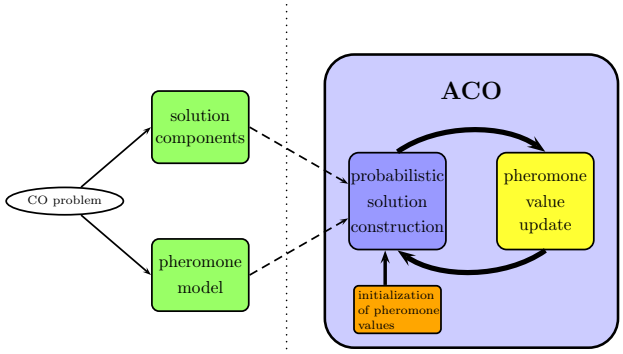
- $\vec{X}^*$ fulfills all constraints
- $f(\vec{X}^*) \leq f(\vec{Y}), \forall\, \vec{Y} \in \mathbb{R}^n$

Ant colony optimization for continuous optimization

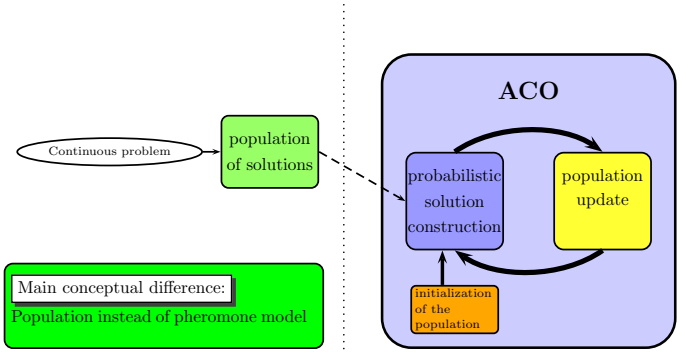
Different approaches:

- N. Monmarché, G. Venturini and M. Slimane. **On how Pachycondyla Apicalis ants suggest a new search algorithm**, *Future Generation Computer Systems*, 16:937–946, 2000.
- K. Socha and M. Dorigo. **Ant colony optimization for continuous domains**, *European Journal of Operational Research*, 185(3):1155–1173, 2008.
- X. M. Hu, J. Zhang and Y. Li. **Orthogonal methods based ant colony search for solving continuous optimization problems**, *Journal of Computer Science & Technology*, 23:2–18, 2008).
- P. Korosec, J. Silc and B. Filipic. **The differential ant-stigmergy algorithm**, *Information Sciences*, 192:82–97, 2012.
- T. Liao et al. **A unified ant colony optimization algorithm for continuous optimization**, *European Journal of Operational Research*, 234(3):597–609, 2014.

Discrete ant colony optimization



Continuous ant colony optimization



Main conceptual difference:
Population instead of pheromone model

Continuous ACO: Probabilistic solution construction

A solution construction: Choose a value $x_i \in \mathbb{R}$ for each variable X_i, $i = 1, \ldots, n$

$\rightarrow n$ solution construction steps

How to choose a value for variable X_i?

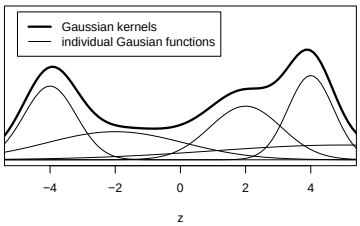
$\rightarrow$ by sampling the following Gaussian kernel probability density function (PDF):

$$G_i(x) = \sum_{j=1}^{k} \omega_j \left(\frac{1}{\sigma_j \sqrt{2\pi}} e^{-\frac{(x-\mu_j)^2}{2\sigma_j{}^2}} \right)$$

where k is the cardinality of the population P.

Continuous ACO: Probabilistic solution construction

A Gaussian kernel PDF:



Continuous ACO: Probabilistic solution construction

Problem: It is quite difficult to sample a Gaussian kernel PDF

Solution: Instead, at the start of each solution construction

1. choose probabilistically one of the Gaussian kernels, denoted by j^*
2. and sample—for all decision variables—the j^*-th Gaussian kernel

Methods for sampling: For example, the Box-Muller method

Continuous ACO: Probabilistic solution construction

Choice of a Gaussian kernel:

$$\mathbf{p}_j = \frac{\omega_j}{\sum_{l=1}^{k} \omega_l} \quad , \forall\, j = 1, \ldots, k$$

Definition of ω_j's:

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} \cdot e^{-\frac{(r_j - 1)^2}{2q^2k^2}}$$

Hereby:

- r_j is the rank of solution j in population P
- q is a parameter of the algorithm: A small q favours high-ranked solutions

Continuous ACO: Probabilistic solution construction

Assumption: Gaussian kernel j^* is chosen for sampling

$$j^*\text{-th Gaussian kernel} = \frac{1}{\sigma_{j^*}\sqrt{2\pi}} e^{-\frac{(x-\mu_{j^*})^2}{2\sigma_{j^*}^2}}$$

What remains? Definition of

1. the mean μ_{j^*}
2. and the standard deviation σ_{j^*}

Continuous ACO: Probabilistic solution construction

Definition of μ_{j^*}:

$$\mu_{j^*} = x_i^{j^*} \quad ,$$

where $x_i^{j^*}$ is the value of the i-th decision variable of solution j^*.

Definition of σ_{j^*}:

$$\sigma_{j^*} = \rho \left(\sqrt{\frac{\sum_{l=1}^{k} \left(x_i^l - x_i^{j^*}\right)^2}{k}} \right)$$

where ρ is a parameter of the algorithm: high ρ means slow convergence speed

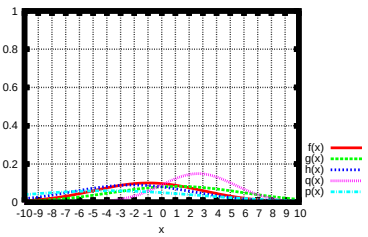
Continuous ACO

Different methods for constraint handling:

1. Repair function: Each unfeasible solution is transformed into a feasible one
2. Penalty function: Unfeasible solutions are penalized by high objective function values

Continuous ACO

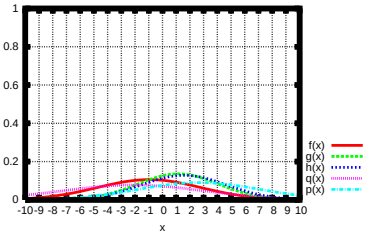
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 1

Continuous ACO

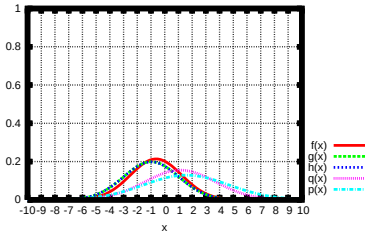
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 2

Continuous ACO

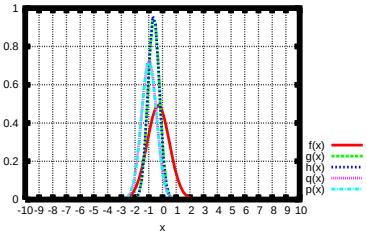
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 3

Continuous ACO

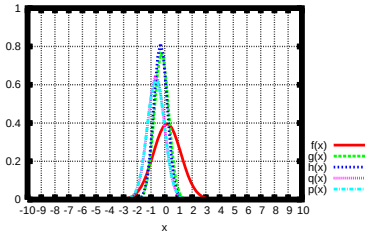
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 4

Continuous ACO

Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 5

Topic 2: Particle Swarm Optimization

Inspiration: Social behavior of flocks of birds and fish schools

PSO: inspiration

Inspiration: Social behaviour observed in animals or insects

Examples:

- Bird flocking
- Fish schooling
- Animal herding

PSO: Facts

Note:

- Invented by J. Kennedy and R. Eberhart in 1995
- Initial intention: Modelling the movements of flocks of birds and fish schools
- PSO deals with a swarm of particles at each iteration
- Particles move in the solution space in the search for good solutions
- The term particles was used because the notion of velocity was adopted
- The initial algorithm is for continuous optimization

PSO: Basic notations

Notations: Each particle $i = 1, \ldots, m$ has a

- current position $\mathbf{x}_i$
- velocity $\mathbf{v}_i$
- personal best position $\mathbf{p}_i$

Furthermore:

- Each particle i has (resp., belongs to) a neighborhood $\mathcal{N}(i) \subseteq \{1, \ldots, m\}$
- $\mathbf{p}_g$ is called the neighborhood best position of i

PSO: Basic algorithm

input: A continuous optimization problem in n dimensions
Generate for each particle i a random initial position $\mathbf{p}_i$, $i = 1, \ldots, m$
$\mathbf{x}_i := \mathbf{p}_i$
while termination conditions not met **do**
  $\mathbf{v}_i := \mathbf{v}_i + \rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i) + \rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$
  $\mathbf{x}_i := \mathbf{x}_i + \mathbf{v}_i$
  if $f(\mathbf{x}_i) > f(\mathbf{p}_i)$ **then** $\mathbf{p}_i := \mathbf{x}_i$
end while
output: The best solution found

Hereby:

- $\rho_k = c_k \cdot \mathbf{r}_k$, where
  1. c_k is a so-called acceleration coefficient
  2. $\mathbf{r}_k$ is a n-dimensional vector of random numbers from $[0, 1]$

PSO: Velocity update

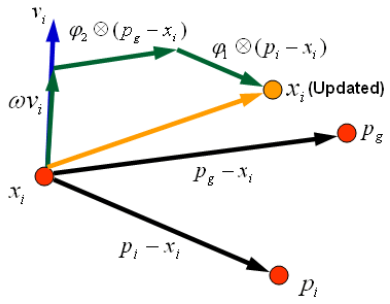
Basic update rule:

$$\mathbf{v}_i := \mathbf{v}_i + \rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i) + \rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$$

Consists of:

1. Momentum term: $\mathbf{v}_i$
   → reinforces the previous direction
2. Cognitive part: $\rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i)$
   → represents the influence of the best solution seen so far by i
3. Social part: $\rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$
   → represents the influence of the best solution seen by the neighborhood of i

PSO: Pictorial view of the update

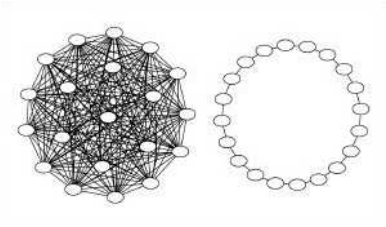


© X. Li

PSO: neighborhoods (1)

Basic division:

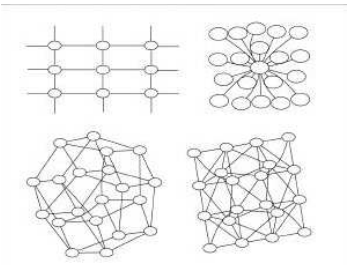
1. gbest PSO: The neighborhood of each particle is the whole swarm
2. lbest PSO: Neighborhoods are restricted



© University of Málaga

PSO: neighborhoods (2)

More sophisticated neighborhoods:



Examples:

- Von Neumann
- Star
- Pyramid
- Dynamic

PSO: neighborhoods (3)

General rules:

- The gbest PSO converges fast, but might miss good solutions
- A lbest PSO converges slower, but usually performs better
- The choice of the right neighborhood is strongly problem dependent
- Dynamic neighborhoods perform usually well, but are computationally more expensive

Systematic evaluation: Two criteria

1. k: degree of connectivity
2. C: amount of clustering

Result: higher k favours exploration, while lower k favours exploitation

PSO: velocity clampling and inertia weight

Problem:

- Observation: Velocities have the tendency to explode to large values
- Result: Particles may leave the boundaries of the search space

Possible solutions:

- Velocity clamping: introducing a maximum velocity v_{max}
- Inertia weight w:

$$\mathbf{v}_i := w \cdot \mathbf{v}_i + \rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i) + \rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$$

Consequences:

1. For $w > 1$: divergence behaviour, for $w < 1$: convergence behaviour
2. Recommendation from literature: Start with $w = 0.9$ and decrease to $w = 0.4$

PSO variants

Main problem: Finding a balance between exploration and exploitation

Some variants:

- Attractive and repulsive PSO (ARPSO)
  Uses different phases of attraction and repulsion between the particles
- Fitness-distance-ratio PSO (FDR-PSO)
  Encourages interaction between particles that are fit and close to each other
- Hierarchical PSO (H-PSO)
  - Organizes the particles in a dynamically changing tree structure
  - Particles are only influenced by their current father

Discrete PSO: binary problems (1)

Note: First discrete PSO introduced in 1997 for binary problems

Changes: with respect to the standard PSO

- The position vectors $\mathbf{x}_i$ are binary
- The position update $\mathbf{x}_i := \mathbf{x}_i + \mathbf{v}_i$ is re-interpreted:

$$\text{if } (r < S(v_{id})) \text{ then } x_{id} = 1, \text{ otherwise } x_{id} = 0$$

where $S()$ is a sigmoidal function, mapping all v_{id} to $[0, 1]$

Note: The velocity update can now be seen as changing the probability that bit x_i will be 1, $i = 1, \ldots, n$

Discrete PSO: binary problems (2)

Extended versions of binary PSO:

- L. Y. Chuang, H. W. Chang, C. J. Tu, et al. **Improved binary PSO for feature selection using gene expression data**. *Computational Biology and Chemistry*, 32, pages 29–37, 2008
- Y. Zhang et al. **Binary PSO with mutation operator for feature selection using decision tree applied to spam detection**, *Knowledge-Based Systems*, 64, pages 22–31, 2014
- J. C.-W. Lin et al. **A binary PSO approach to mine high-utility itemsets**, *Soft Computing*, 2016, in press.

Discrete PSO: TSP example

Basic update rule:

$$\mathbf{v}_i := \mathbf{v}_i + \rho_1 \cdot (\mathbf{p}_i - \mathbf{x}_i) + \rho_2 \cdot (\mathbf{p}_g - \mathbf{x}_i)$$
$$\mathbf{x}_i := \mathbf{x}_i + \mathbf{v}_i$$

Note: In order to apply PSO to any optimziation problem we need to define ...

- Position of a particle
- Velocity of a particle
- Addition of position and velocity. **Result:** Position
- Substraction of positions. **Result:** Velocity
- Addition of velocities. **Result:** Velocity
- Multiplying a velocity with a real. **Result:** Velocity

TSP example: positions

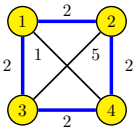
Definition:

- Given: TSP instance with n cities
- Position: Permutation of the n cities + first city added to the end

Example:

$(1, 2, 4, 3, 1)$

Position



Graphically

TSP example: velocity

Definition:

- List of swaps of city identifiers

Example:

$$((1, 4), (3, 4))$$

Note:

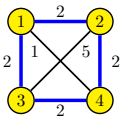
- Null-velocity: empty list
- Oposite velocity: reversed list. Example: $((1, 4), (3, 4)) \rightarrow ((3, 4), (1, 4))$

TSP example: addition of position and velocity

Definition:

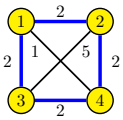
- Given: A position x and a velocity v
- Apply all swaps of city identifiers of v to position x

Example: $x = (1, 2, 4, 3, 1)$, $v = ((1, 4), (3, 4))$



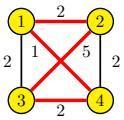
$(1, 2, 4, 3, 1)$

First swap



$(4, 2, 1, 3, 4)$

Second swap



$(3, 2, 1, 4, 3)$

TSP example: substraction and addition

Substraction of positions: Results in a velocity

- Given: Two positions x_1 and x_2. We want $x_1 - x_2$
- Resulting velocity v: sequence of swaps that transforms x_1 into x_2
- Example: $x_1 = (1, 2, 4, 3, 1)$, $x_2 = (1, 4, 2, 3, 1) \rightarrow v = ((2, 4))$

Addition of velocities: Results in a velocity

- Given: Two velocities v_1 and v_2. We want $v = v_1 + v_2$
- Resulting velocity v: merge lists v_1 and v_2 and remove all doubles
- Example: $v_1 = ((1, 3), (2, 4))$, $v_2 = ((1, 4), (2, 4)) \rightarrow v = ((1, 3), (2, 4), (1, 4))$

TSP example: multiplication with a real-number

Multiplication: Multiplying a velocity with a real-number

- Given: A velocity v and a real-number $r \in [0, 1]$
- Result: Reduce v to the first $100 \cdot r\%$ of the swaps

Note:

- This completes the necessary definitions
- This is only an example. This algorithm will not work very well

Discrete PSO: Other examples

Some references:

- D. Y. Sha and C.-Y. Hsu. **A new particle swarm optimization for the open shop scheduling problem**, *Computers and Operations Research*, 35(10):3243–3261, 2008
- Y. Tian et al. **A discrete PSO for two-stage assembly scheduling problem**, *The International Journal of Advanced Manufacturing Technology*, 66 (1-4), pages, 481–499, 2013
- M. Gómez-González, A. López and F. Jurado. **Hybrid discrete PSO and OPF approach for optimization of biomass fueled micro-scale energy system**, *Energy Conversion and Management*, 65, pages 539–545, 2013

Topic 3: Self-Synchronized Duty-Cycling in Sensor Networks

Inspiration: Self-synchronized activity phases of ant colonies

Self-synchronized activity phases in ant colonies (1)

Biologist discovered:

- Colonies of ants show synchronized activity patterns
- Synchronization is achieved in a self-organized way: self-synchronization
- Synchronized activity ...
  1. ... provides a mechanism for information propagation
  2. ... facilitates the sampling of information from other individuals

Model of self-synchronization:

J. Delgado and R.V. Solé. **Self-synchronization and task fulfilment in ant colonies**, *Journal of Theoretical Biology*, 205, 433–441 (2000)

Self-synchronized activity phases in ant colonies (2)

- Each ant is modelled as an automaton
- The state of an automaton i is described by a continuous state variable:

$$S_i(t) \in \mathbf{R} \text{ where } t \text{ is the time step}$$

- Each automaton i can move on a LxL grid with periodic boundary conditions
- At time step t, each automaton i is either active or inactive:

$$a_i(t) = \Phi(S_i(t) - \theta_{act}) \text{ , where}$$

  - θ_{act}: activation threshold
  - $\Phi(x) = 1$ if $x \geq 0$, and $\Phi(x) = 0$ otherwise

Self-synchronized activity phases in ant colonies (3)

Simulation: At each iteration t

1. Activity calculation:
   - Calculate $a_i(t)$
   - If $a_i(t) = 0$: Spontaniously activate i with probability p_a (activity level S_a)
2. Move: Each active automaton i moves (if possible) to one of the free places in its 8-neighborhood
3. State variable update:

$$S_i(t+1) = \tanh(g \cdot (S_i(t) + \sum_{j \in N_i} S_j(t)))$$

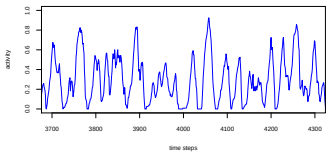
where N_i is the 8-neighborhood of the position of i

Self-synchronized activity phases in ant colonies (4)

What do we measure? Mean activity of the system at time t:

$$A(t) = \frac{1}{N} \sum_{i=1}^{N} a_i(t)$$

where N is the number of automata



Design of a duty-cycling protocol: organization

Organization:

Note: during the duty-cycling phase (DC) ...

- .... all nodes are awake
- Each node executes one duty-cyling event at a **randomly chosen time**. Includes sending of one message.

Design of a duty-cycling protocol: sensor nodes

Note: Automata are now mobile or static sensor nodes

Each sensor i ...

- has a battery with level $0 \leq b_i(t) \geq 1$
- maintains a special message queue Q_i for incoming duty-cycling messages
- is equipped with a radio antenna that allows to choose from a set $\{T^1, \ldots, T^n\}$ of n different transmission power levels

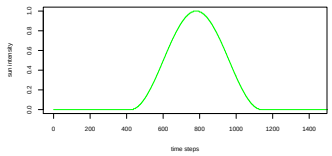
Contents of a message $m \in Q_i$: the sender nodes state variable value

Design of a duty-cycling protocol: energy harvesting

Characteristics:

- Sensor nodes that are equipped with solar panels
- Each sensor i harvests a certain amount of energy per time step

Daily sun intensity:



Design of a duty-cycling protocol: duty-cycling events

1: Calculate a_i
2: **if** $a_i = 0$ **then**
3: Draw a random number $p \in [0, 1]$
4: **if** $p \leq p_a$ **then** $S_i := S_a$ and $a_i := 1$ **endif**
5: **end if**
6: Determine transmission power level T_i
7: Compute new value for state variable S_i
8: Send duty-cycling message m (containing value S_i) with transmission power T_i

Choice of the transmission power level

Ideal transmission power level:

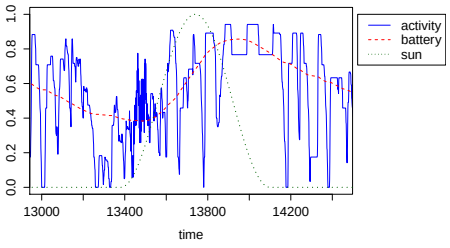
$$T_i := T_{\min} \cdot (1 - b_i) + T_{\max} \cdot b_i$$

where

- $T_{\min}$: minimum transmission power level
- $T_{\max}$: maximum transmission power level

Experimental Results (1): Network simulator Shawn

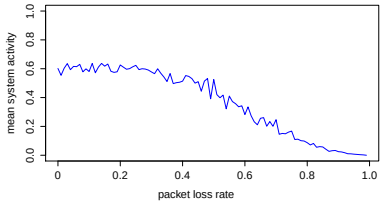
Parameters: 120 static sensor nodes, no packet loss



Note: The average activity of the system is at about 0.6

Experimental Results (2): Packet Loss

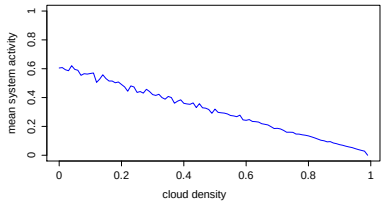
Parameters: 120 static sensor nodes, different packet loss rates



Note: System is very robust up to a packet loss rate of about 0.3

Experimental Results (3): Restricted Energy Harvesting

Parameters: 120 static sensor nodes, different cloud densities



Note: linear relationship between cloud density and system activity

Self-synchronized duty-cycling: literature

Some Papers:

- H. Hernández, C. Blum, M. Middendorf, K. Ramsch and A. Scheidler. **Self-synchronized duty-cycling for mobile sensor networks with energy harvesting capabilities: A swarm intelligence study**. *Proceedings of SIS 2009*, pages 153–159, IEEE press, 2009.
- H. Hernández and C. Blum. **Foundations of ANTCYCLE: Self-synchronized duty-cycling in mobile sensor networks**. *The Computer Journal*, 54(9), 1427–1448, 2011.

Topic 4: Distributed Graph Coloring

Inspiration: Self-desynchronization of Japanese tree frogs

De-synchronization in Japanese Tree Frogs (1)

Biologist discovered:

- Male Japanese Tree Frogs de-couple their calls
- WHY?
  - The purpose of the calls is to attract females
  - Female frogs cannot distinguish between too close calls
  - Result: females cannot determine the correct direction

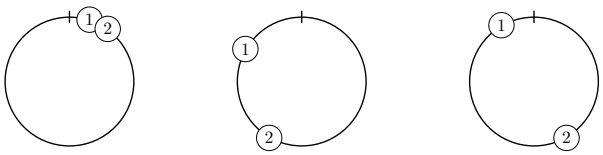
Mathematical model:

I. Aihara, H. Kitahata, K. Yoshikawa and K. Aihara. **Mathematical modeling of frogs' calling behavior and its possible applications to artificial life and robotics.** *Artificial Life and Robotics*, 12(1):29–32, 2008.

De-synchronization in Japanese Tree Frogs (2)

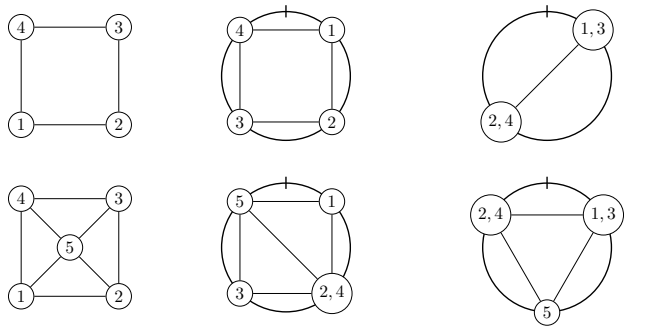
Model components:

- A set of pulse-coupled oscillators.
- Some oscillators are coupled, others are independent of each other
- Each oscillator i has a phase $\theta_i \in [0, 1)$ which changes over time



De-synchronization in Japanese Tree Frogs (3)

Topology | Suboptimal de-synchronization | Optimal de-synchronization



Distributed graph coloring: organization of the algorithm

Implementation:

- In static wireless ad-hoc networks (such as sensor networks)
- Algorithm works with communication rounds (length: 1 time unit)

Each network node i maintains:

- $\theta_i \in [0,1]$: a **graph coloring event** is executed at time θ_i in each communication round
- $c_i \in \{0,1,...\}$: the **current color** of the node

Distributed graph coloring: graph coloring event

1: **PHASE I**
2: Recalculate θ_i
3: Choose a (new) color c_i
4: Send a graph coloring event message m to the neighbors
5:
6: **PHASE II**
7: Execute a kind of distributed local search

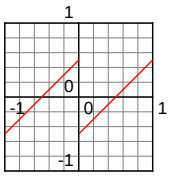
Messages:

- Graph coloring event messages are collected in a separate message queue M_i
- Each message m contains two values:
  - The θ-value theta_m of the sender node
  - The current color color_m of the sender node

Distributed graph coloring: re-calculating θ_i

$$\theta_i := \theta_i + \sum_{m \in M_i} \textbf{inc}(\text{theta}_m - \theta_i)$$

Function **inc**():



Distributed graph coloring: choosing a new color c_i

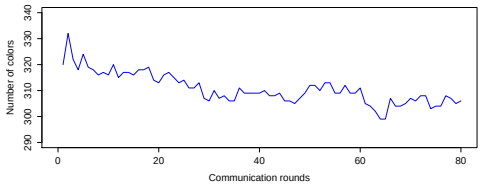
$$c_i := \min\{c \in \mathbb{N} \mid \nexists m \in M_i \text{ with } \text{color}_m = c\}.$$

Use of the θ-values:

- They determine the order in which nodes choose a color
- This is in contrast to an exsiting attempt to use frogs' behavior for graph coloring

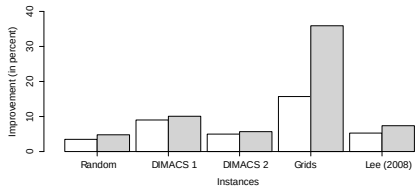
Experiments: quality of the coloring over time

Example: DIMACS graph DSJC1000.9.col



Experiments: comparison

Avg. improvement over: Algorithm by Finocchi et al.



- I. Finocchi, A. Panconesi and R. Silvestri. **An experimental analysis of simple distributed vertex coloring algorithms**, *Algorithmica*, 41(1), 1–23, 2005

Distributed graph coloring: literature

Some Papers:

- H. Hernández and C. Blum. **Distributed Graph Coloring: An Approach Based on the Calling Behavior of Japanese Tree Frogs**. *Swarm Intelligence*, 6(2), 117–150, 2012
- H. Hernández and C. Blum. **FrogSim: distributed graph coloring in wireless ad hoc networks**, *Telecommunication Systems*, 55(2), pages 211–223, 2014
- C. Blum, B. Calvo and M. J. Blesa. **FrogCOL and FrogMIS: new decentralized algorithms for finding large independent sets in graphs**, *Swarm intelligence*, 9(2-3), pages 205–227, 2015

Extension: Finding Large Independent Sets

Independent set | Maximal independent set | Maximum independent set

Complexity in centralized settings:

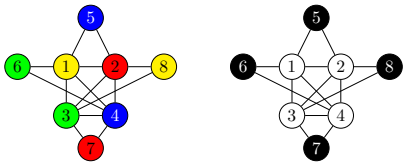
- Problem of finding a maximal independent set is in P
- Problem of finding a maximum independent set is NP-hard

Algorithms from the literature: (for the de-centralized case)

- Simple distributed greedy algorithms
- Iterative self-stabilizing algorithms

Idea: Use Existing FroSim Algorithm

Observe: relation between graph coloring and independent sets



Experiments (1)

Competitors:

- Centralized Greedy: Just for interest
- FruitFly: Newest iterative self-stabilizing algorithm published in the literature

Inspiration of Fruitfly: Development of the fly's nervous system, when sensory organ precursor (SOP) cells are chosen

Paper:

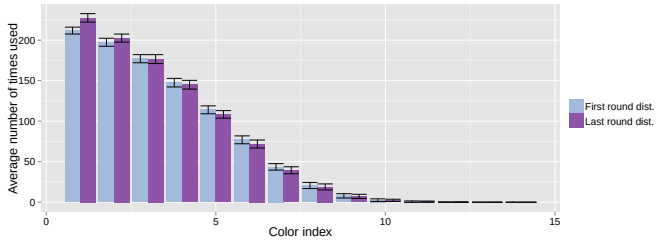
Afek, Y., Alon, N., Barad, O., Hornstein, E., Barkai, N., Bar-Joseph, Z. **A biological solution to a fundamental distributed computing problem.** *Science*, 331:183–185, 2011.

Numerical Results: Random geometric graphs of 1000 nodes

radius (r)	Greedy	FruitFly		FrogSim		
		avg.	**rounds**	**avg.**	**rounds**	**convergence**
0.049	244.88	225.39	66.66	229.76	416.14	734.08
0.0578	190.96	176.50	115.30	180.26	414.50	758.72
0.0666	152.35	142.98	236.02	144.82	325.74	775.16
0.0754	124.53	118.74	480.64	118.82	272.97	770.88
0.0842	103.82	100.91	*1114.90*	99.55	248.03	754.76
0.093	87.82	87.19	*2562.90*	84.93	279.20	755.75
0.1018	75.42	76.72	*7014.74*	73.16	212.85	750.31
0.1106	65.61	67.91	*22063.76*	64.14	205.49	740.14
0.1194	57.84	60.60	*53523.54*	56.55	215.69	684.79
0.1282	51.53	54.60	*165323.04*	50.16	165.77	700.42
0.134	47.83	50.84	*321192.92*	46.95	160.96	678.45

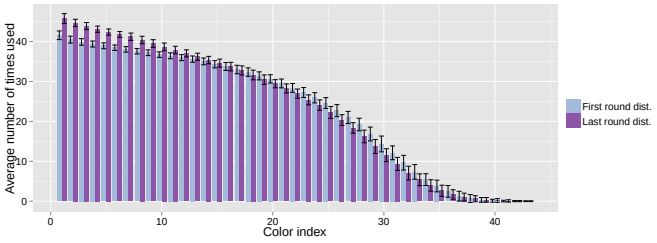
Indication For Convergence To Good Solutions (1)

Example: Sparse graph on 1000 nodes



Indication for Convergence To Good Solutions (2)

Example: Dense graph on 1000 nodes



Summary and Conclusions

Presented Topics:

- Ant colony optimization
- Particle swarm optimization
- Self-synchronized duty-cycling in sensor networks
- Distributed graph coloring in wireless ad-hoc networks

Observation:

- It is still very much possible to find swarm-intelligent behaviors in the biological literature, which are still not used for any technical applications.

Swarm Intelligence: Quo vadis?

- Problem: Swarm intelligence has attracted too many people
- As a consequence:
  1. Experienced researchers were overwhelmed with reviewing
  2. People who should have never been asked to do so did reviewing work
- Therefore: nowadays we find numerous papers in the literature that are either
  1. Non-sense, or
  2. Re-inventing the wheel

First steps against this trend:

- Some journals (J. of Heur. , Comp. & Oper. Res.) ask for algorithms to be described in metahpor-free language
- Colleagues start to expose the problem (G. Rudolph , K. Sörensen)

Questions?