

# Genetic Algorithms and Genetic Programming


Lecture 13: (6/11/09)

## Ant Colony Optimization IIa



Michael Herrmann

# Overview: Remainder of the course

- I. GA (1-7)
- II. GP (8-10)
- III. ACO (11-13): Ant colony optimization 
- IV. PSO (14-15): Particle swarm optimization and differential evolution
- V. NC (16): Overview on DNA computing, Membrane computing, Molecular computing, Amorphous computing, Organic computing, ....
- VI. Wrapping up: Metaheuristic search (17)

Not included:

artificial neural networks, quantum computing, cellular automata, artificial immune systems

# ACO algorithm

---

**Algorithm 1** The framework of a basic ACO algorithm

---

**input:** An instance  $P$  of a CO problem model  $\mathcal{P} = (\mathcal{S}, f, \Omega)$ .

InitializePheromoneValues( $\mathcal{T}$ )

$s_{bs} \leftarrow \text{NULL}$

init best-so-far solution

**while** termination conditions not met **do**

$\mathcal{S}_{iter} \leftarrow \emptyset$

**for**  $j = 1, \dots, n_a$  **do**

loop over ants

$s \leftarrow \text{ConstructSolution}(\mathcal{T})$

set of valid solutions

**if**  $s$  is a valid solution **then**

$s \leftarrow \text{LocalSearch}(s)$       {optional}

**if**  $(f(s) < f(s_{bs}))$  or  $(s_{bs} = \text{NULL})$  **then**  $s_{bs} \leftarrow s$

update best-so-far  
store valid solutions

$\mathcal{S}_{iter} \leftarrow \mathcal{S}_{iter} \cup \{s\}$

**end if**

**end for**

    ApplyPheromoneUpdate( $\mathcal{T}, \mathcal{S}_{iter}, s_{bs}$ )

**end while**

**output:** The best-so-far solution  $s_{bs}$

---

# Some general considerations

- Best ant laying pheromone (global-best ant or, in some versions of ACO, iteration-best ant) encourage ants to follow the best tour or to search in the neighbourhood of this tour (make sure that  $\tau_{\min} > 0$ ).
- Local updating (the ants lay pheromone as they go along without waiting till end of tour). Can set up the evaporation rate so that local updating “eats away” pheromone, and thus visited edges are seen as less desirable, encourages exploration. (Because the pheromone added is quite small compared with the amount that evaporates.)
- Heuristic improvements like 3-opt – not really “ant”-style
- “Guided parallel stochastic search in region of best tour” [Dorigo and Gambardella], i.e. assuming a non-deceptive problem.

# Max-Min Ant System (MMAS)

- (i) Only the best ant adds pheromone trails (iteration best or best so far)

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

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

- (ii) Minimum and maximum values of the pheromone are explicitly limited (by truncation):  $\tau_{\min}$ ,  $\tau_{\max}$

Pseudorandom proportional rule:

$$p(c_{ij} | s_k^p) = \begin{cases} \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{c_{im} \in N(s_k^p)} \tau_{im}^\alpha \eta_{im}^\beta} & \text{if } j \in N(s_k^p) \\ 0 & \text{otherwise} \end{cases}$$

Initialize by maximum (minimum empirical)  $\tau_{\max} = \frac{1}{\rho L^*}$ ,  $L^*$  best so far or optimum (if known)

# Theoretical results: Overview

- Convergence in probability of an ACO algorithm (Gutjahr 2000)  
(Theoretical bounds, but not very practical)
- Run-time analysis
- Understanding ACO: Search biases
- Relations to other optimization algorithms

# Search biases

- A desirable search bias towards good zones of the search space is given by the pheromones
- *Negative search bias caused by selection fix-points*
- *Negative search bias caused by an unfair competition*
- Note: For these theoretical considerations local heuristic information is ignored (e.g. by setting  $\beta=0$ ), i.e. the question is: What can ACO do beyond local search?

# Selection fix-points: Ant number

- Why to use more than one ant per iteration? Wouldn't the algorithm work with only one ant?

$$\tau_i^j(t+1) = (1 - \rho) \cdot \tau_i^j(t) + \rho \cdot \Delta_i^j$$

$$\Delta_i^j = \mathbf{p}(c_i^j \mid \mathcal{T}) = \tau_i^j(t) \bigg/ \sum_{k=1}^{|D_i|} \tau_i^k(t)$$

Peromones tend towards  $\tau_i^j(t) = \Delta_i^j$

Effect increases with problem size. (Merkle & Middendorf 2004)

- Several ants building a common pheromone matrix may contribute “building blocks” to a solution that is likely to be followed by the ants of later generations.
- The competition between the ants is a driving force of ACO algorithms.
- This is analogous to but not generally the same as in GA



# Selection fix-points: Constraints

- The adaptation of the pheromone matrix depends also on the number of ants having passed a solution component
- In unconstrained problem all nodes of the underlying graph have the same degree
- In constrained problems the degree may differ such that poor regions with low degrees become more attractive than good regions with high degree (cf. 2. exercise of set 6)
- One can construct examples where the increased exploration of the bad regions lead to a fixed point of the algorithm or a local minimum of the search problem

# Bias by an unfair competition

- Unconstrained ACOs always improve the iteration quality (expected value of the ants' performance) or are stationary

- Constrained problem: minimal  $k$ -cardinality tree

- For example the trivial case

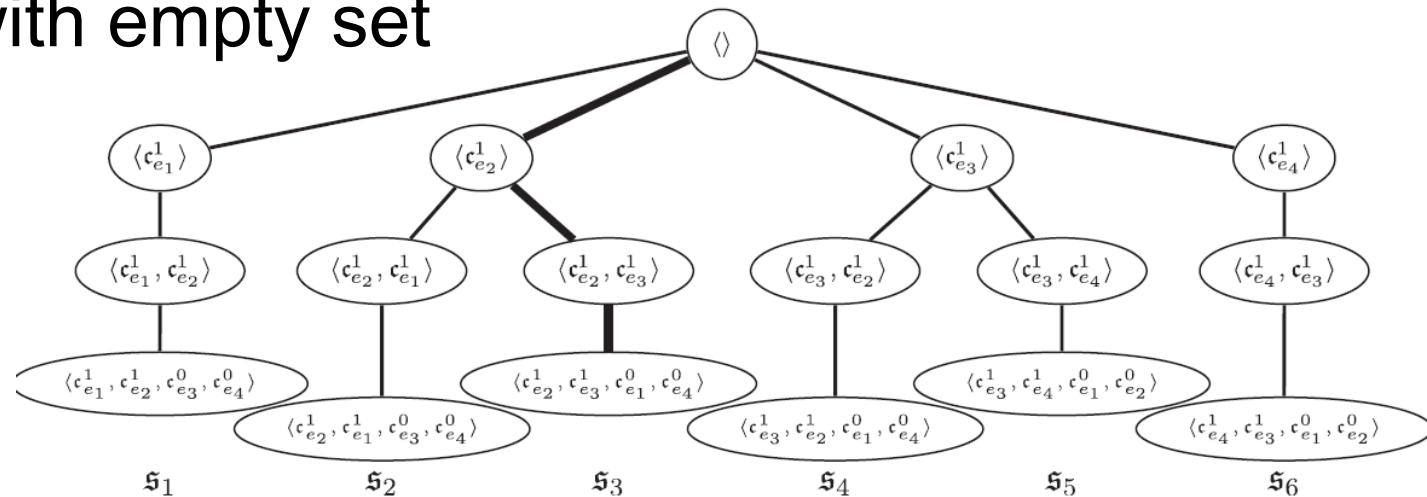


- Start solution with empty set

- add one node

- etc.

- finally



- obj. function

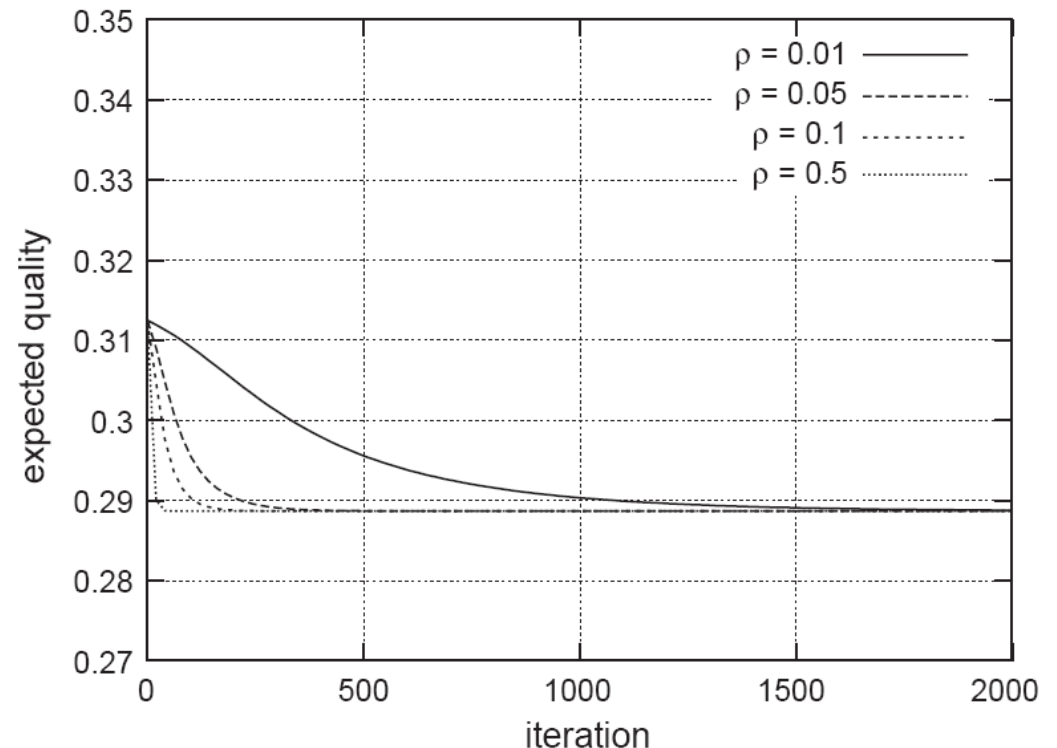
$$f(s_1) = f(s_2) = f(s_5) = f(s_6) = 3 \text{ and } f(s_3) = f(s_4) = 4$$

- since

$$w(e_1) = w(e_4) = 1 \text{ and } w(e_2) = w(e_3) = 2$$

# Bias by an unfair competition

- Quality decreases (!) when starting from a homogeneous initial pheromone matrix
- The branching paths get twice as much update although they lead to a larger expected cost
- The impact of the pheromone value update increases and the expected iteration quality decreases faster
- Note: Quality relates to inverse costs



# Hyper-cube framework (HC-ACO)

[Not an algorithm, but a framework which applies for several variants]

## Pheromone update

$o_j$  is a component of solution  $i$

$$\tau_j \leftarrow \rho \cdot \tau_j + \sum_{i=1}^k \Delta\tau_j^i \quad \text{where} \quad \Delta\tau_j^i = \begin{cases} \frac{1}{f(s^i)} & \text{if } o_j \in s^i \\ 0 & \text{otherwise} \end{cases}$$

$$\lim_{t \rightarrow \infty} \tau_i(t) \leq \frac{1}{1 - \rho} \cdot \frac{k}{f(s^{opt})}$$

Maximum if all ant follow forever the optimal solution:  $\tau_i = \rho \tau_i + k / f(s^{opt})$

$\tau = (\tau_1, \dots, \tau_k)$  is a  $k$ -dimensional vector in  $[\tau_{\min}, \tau_{\max}]^k$

$$\tau = \sum_{j=1}^M \alpha_j s_j, \quad \alpha_j \in [\tau_{\min}, \tau_{\max}], \quad s_j \in \{0,1\} \quad \text{w.l.o.g.: } \alpha_j \in [0,1]$$

$M$ : number of solution components (e.g. edges of a graph)

# Search space: “Hyper-cube framework”

- A binarised solution  $s=(s_1, \dots, s_M)$  is a subset of the edges  $E$  of a graph  $G=(N, E)$  indicated by  $s_i$  being 1 or 0.

- Pheromone normalisation

$$\tau_j \leftarrow \rho \cdot \tau_j + (1 - \rho) \cdot \sum_{i=1}^k \Delta\tau_j^i \quad \text{where} \quad \Delta\tau_j^i = \begin{cases} \frac{\frac{1}{f(s^i)}}{\sum_{l=1}^k \frac{1}{f(s^l)}} & \text{if } o_j \in s^i \\ 0 & \text{otherwise} \end{cases}$$

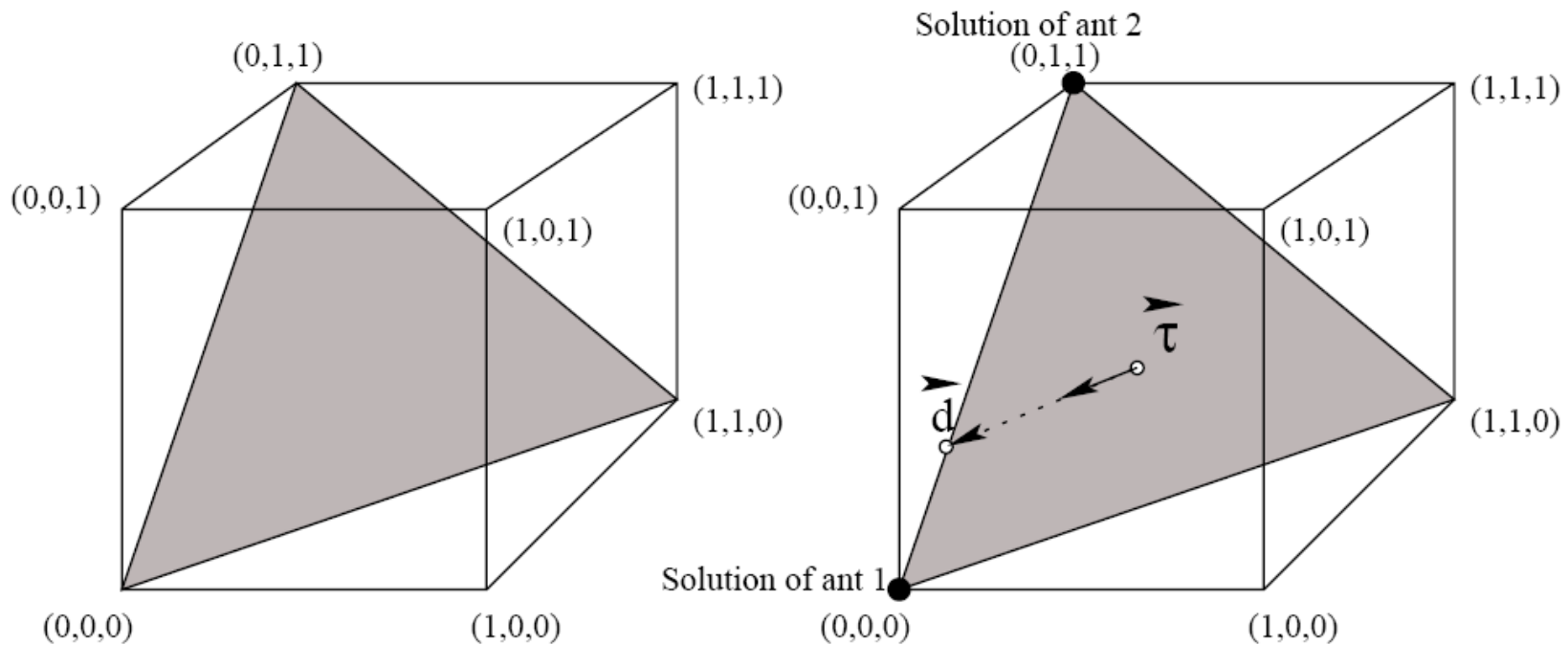
- Hyper-cube update rule ( $\mu=1-\rho$ )

$$\vec{\tau} \leftarrow \vec{\tau} + \mu \cdot (\vec{d} - \vec{\tau})$$
$$\vec{d} = (d_1, \dots, d_n) \quad \text{where} \quad d_j = \sum_{i=1}^k \frac{\frac{1}{f(s^i)} \cdot s_j^i}{\sum_{l=1}^k \frac{1}{f(s^l)}}, \quad j = 1, \dots, n$$

- Pheromones are updated in the span of the solutions

# Search space: “Hyper-cube framework”

$$\vec{\tau} \longleftarrow \vec{\tau} + \mu \cdot (\vec{d} - \vec{\tau})$$



The pheromone vector moves a bit towards the weighted mean of the solutions produced by the current iteration.

# Benefits of the Hyper-cube framework

- A diversification scheme

global desirability: 
$$v_j^{des} \leftarrow \max\left\{\frac{1}{f(s)} : s \in \mathcal{S}_{ants}, s_j = 1\right\}$$

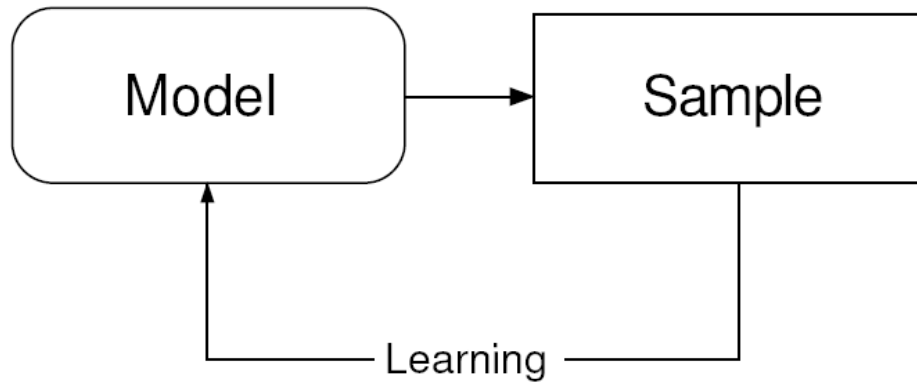
global frequency: 
$$v_j^{fr} \leftarrow \sum_{s \in \mathcal{S}_{ants}} s_j$$

$\mathcal{S}_{ants}$  all solutions generated since the start

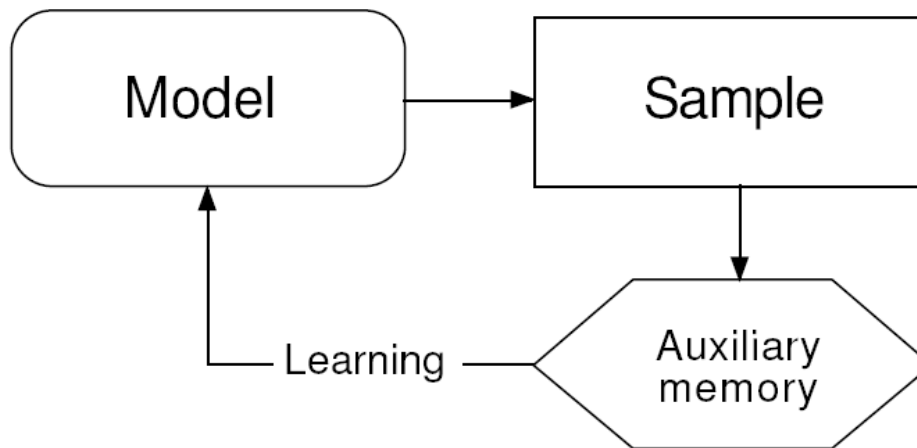
At stagnation the algorithm may be restarted with a pheromone ( $n \times n$ ) matrix (or vector in  $n(n-1)/2$  dimensions) constructed from  $v^{des}$  or the inverse of  $v^{fr}$  in order to keep good solutions, but also to favour regions where few ants have been before.

(More generally, the benefit is theoretical convenience!)

# Relation to other algorithms: Model-Based Search



Scheme of the MBS approach



MBS approach with memory

E.g. in ACO:

- Model:  
pheromone matrix
- Sample:  
ants following  
pheromone traces
- Learning:  
pheromone update
  
- Auxiliary memory:  
best-so-far solution



# Model Based Search

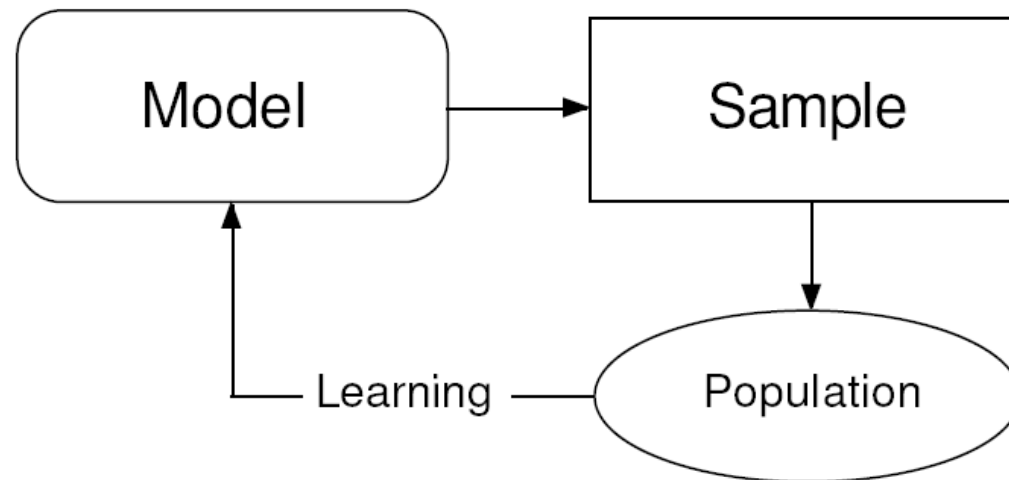
- Candidate solutions are constructed using some parameterized probabilistic model, that is, a parameterized probability distribution over the solution space.
- The candidate solutions are used to modify the model in a way that is deemed to bias future sampling toward low cost solutions.

# ACO as MBO

- ✿ A finite set  $C = \{c_1, c_2, \dots, c_{N_C}\}$  of *components*, where  $N_C$  is the number of components
- ✿ A finite set  $X$  of *states* of the problem, where a state is a sequence  $x = \{c_i, c_j, \dots, c_k, \dots\}$  over the elements of  $C$ . The length of a sequence  $x$ , that is, the number of components in the sequence, is expressed by  $|x|$ . The set of (candidate) solutions  $S$  is a subset of  $X$  (i.e.  $S \subseteq X$ ).
- ✿ A set of feasible states  $X_f$ , with  $X_f \subseteq X$ , defined via a set of *constraints*  $\Omega$
- ✿ A non-empty set  $S^*$  of optimal solutions, with  $S^* \subseteq X_f$  and  $S^* \subseteq S$
- ✿ Formulation of the update in the hyper-cube framework
- ✿ Result is a fully-connected weighted graph

# GA as MBS

- Generate new solutions using the current probabilistic model
- Replace (some of) the old solutions by the new ones.
- Modify the model using the new population.



# GA as MBS

compact Genetic Algorithm  
(cGA) (Harik et al., 1999)

- Probabilistic simulation of a genetic algorithm with tournament selection
- Probabilistic model of the population: individuals are generated by biased draws based on a probability vector. E.g. if the vector entry  $p_i$  is 0.9 it is likely to have a 1 at position  $i$  in this individual's string.
- Tournament selection: Choose two individuals  $a$  and  $b$

if  $a_i \neq b_i$  then

if  $a_i = 1$  then  $p_i \leftarrow p_i + 1/n$

else  $p_i \leftarrow p_i - 1/n$

- The model is updated by

$$p_i \leftarrow p_i + \frac{1}{n}(a_i - b_i)$$

# GA as MBS

- Bits in the genome were chosen independently. What about schemata?
- Modeling dependencies between string positions e.g.
  - learning a chain distribution as in ACO starting at the first character of the string and setting the next one by a conditional probability
  - by a matrix of pair-wise joint frequencies
  - by a forest of mutually independent dependency trees
- In order to capture the essential idea of GA (building blocks the probabilistic model must be different from the ACO model (i.e. the pheromone matrix + update))

# ACO Reading suggestions

## General:

- ⑩ M. Dorigo & K. Socha, An Introduction to Ant Colony Optimization: In T. F. Gonzalez, Approximation Algorithms and Metaheuristics, CRC Press, 2007. IridiaTr2006-010r003.pdf
- M. Dorigo, T. Stützle (2004) Ant Colony Optimization, MIT Press.

## Theory:

- M. Dorigo, V. Maniezzo, A. Coloni (1996) Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics B* 26:1, 1-13.
- M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243-278, 2005.

## Applications

- see proceedings of the ANTS conferences or the journal *Swarm Intelligence*