Ant Colony Optimization: Introduction and Recent Advances

Christian Blum

Albcom, Lsi,
Universitat Politècnica De Catalunya
Barcelona, Spain

Overview

What is ant colony optimization (ACO)? A technique for optimization whose inspiration is the foraging behaviour of real ant colonies.

Different topics of the tutorial: Part I

- **Swarm intelligence:** Origins and inspiration of ACO
- **The ACO metaheuristic:**
  - How does it work?
  - Application examples:
    - Traveling salesman problem
    - Assembly line balancing

Overview

Different topics of the tutorial: Part II

- **Hybridization** with other optimization techniques
- **Negative search bias:** When ACO algorithms may fail
- Ant colony optimization for **continuous optimization**

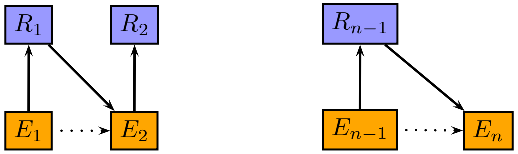
Swarm intelligence

The origins of ant colony optimization

Swarm intelligence

Properties of social societies:

- 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 can be accomplished in cooperation

Swarm intelligence

Natural examples

- Nest building behaviour of wasps
- Agricultural behaviour of leaf-cutter ants
- Nest building behaviour of weaver ants
- Cemetary building behaviour of ants
- Foraging behaviour of ants

Swarm intelligence

Inspiration: Collective behaviour of social insects, flocks of birds, or fish schools

© Soren Breitling

© SuperStock, Inc.

Swarm intelligence

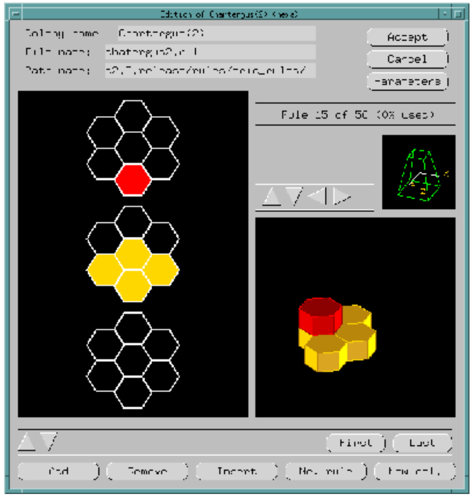
Examples of social insects:

- Ants
- Termites
- Some wasps and bees

Some facts:

- About 2% of all insects are social
- About 50% of all social insects are ants
- The total weight of ants is about the total weight of humans
- Ants colonize the world since 100.000.000 years, humans only since 50.000 years

Swarm intelligence



© Dominique Snyers (`http://www-iasc.enst-bretagne.fr/PROJECTS/SWARM/nest.html`)

Swarm intelligence

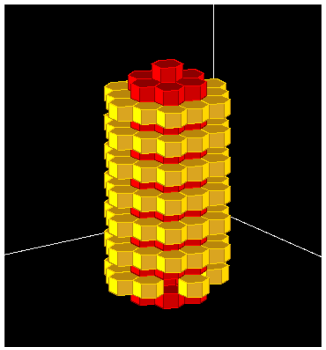
Natural examples

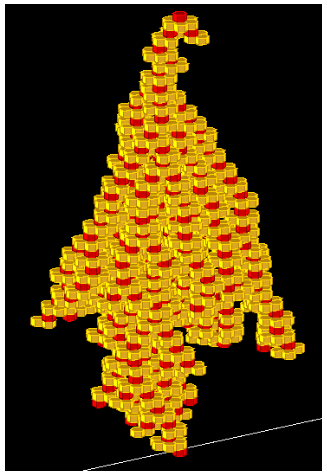
- Nest building behaviour of wasps
- Agricultural behaviour of leaf-cutter ants
- Nest building behaviour of weaver ants
- Cemetary building behaviour of ants
- Foraging behaviour of ants

Swarm intelligence

© Dominique Snyers (`http://www-iasc.enst-bretagne.fr/PROJECTS/SWARM/nest.html`)

Swarm intelligence





© Dominique Snyers (`http://www-iasc.enst-bretagne.fr/PROJECTS/SWARM/nest.html`)

Swarm intelligence

© Alex Wild (`http://www.myrmecos.net`)

Swarm intelligence

Natural examples

- Nest building behaviour of wasps
- Agricultural behaviour of leaf-cutter ants
- Nest building behaviour of weaver ants
- Cemetary building behaviour of ants
- Foraging behaviour of ants

Swarm intelligence

© Alex Wild (`http://www.myrmecos.net`)

Swarm intelligence

© Alex Wild (`http://www.myrmecos.net`)

Swarm intelligence

Swarm intelligence

Natural examples

- Nest building behaviour of wasps
- Agricultural behaviour of leaf-cutter ants
- Nest building behaviour of weaver ants
- Cemetary building behaviour of ants
- Foraging behaviour of ants

Swarm intelligence

Swarm intelligence

Swarm intelligence

Natural examples

- Nest building behaviour of wasps
- Agricultural behaviour of leaf-cutter ants
- Nest building behaviour of weaver ants
- Cemetary building behaviour of ants
- Foraging behaviour of ants

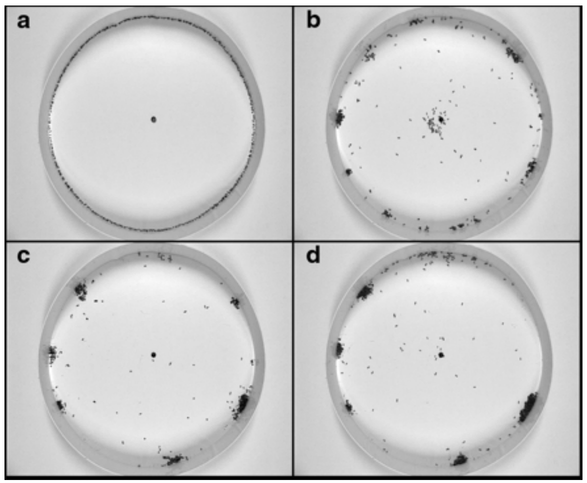
Swarm intelligence

Communication strategies:

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

Basic behaviour:

Swarm intelligence



© by the National Academy of Sciences (PNAS)

Swarm intelligence

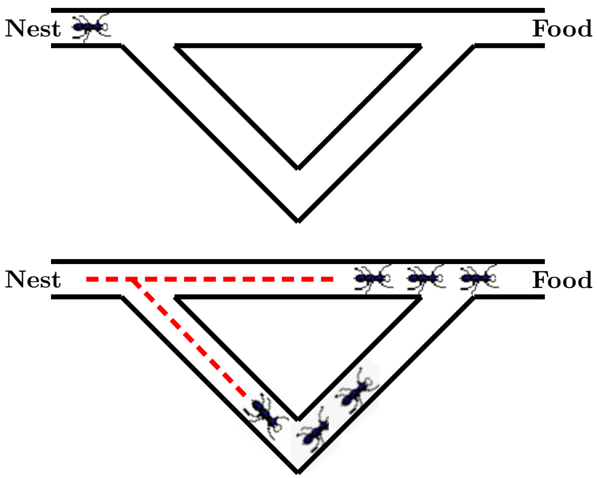
Communication strategies:

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

© Alex Wild (http://www.myrmecos.net)

© Christian Blum

Swarm intelligence



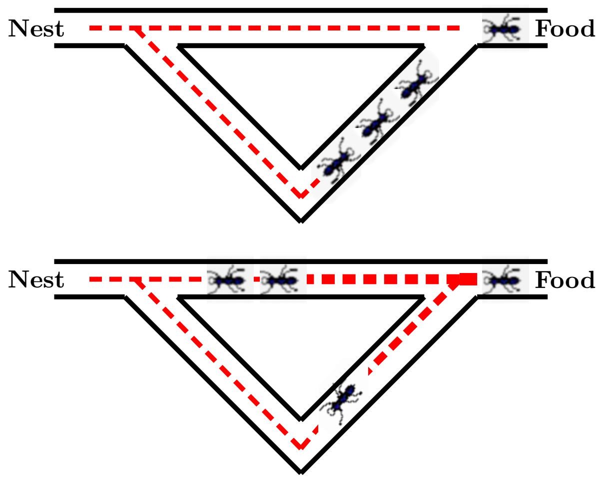
The ant colony optimization metaheuristic

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

Swarm intelligence

© Alex Wild (`http://www.myrmecos.net`)

Swarm intelligence



The ant colony optimization metaheuristic

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:

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

The ant colony optimization metaheuristic

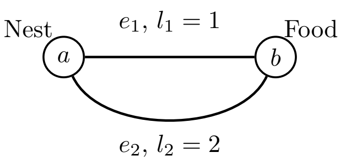
Main differences between model and reality:

	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

The ant colony optimization metaheuristic

Technical simulation:



1. We introduce artificial pheromone parameters:

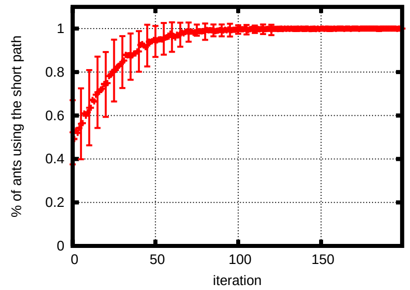
$$\mathcal{T}_1 \text{ for } e_1 \text{ and } \mathcal{T}_2 \text{ for } e_2$$

2. W initialize the phermomone values:

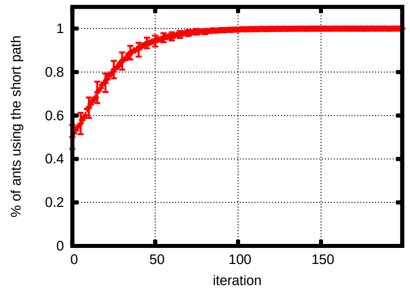
$$\tau_1 = \tau_2 = c > 0$$

The ant colony optimization metaheuristic

Simulation results:



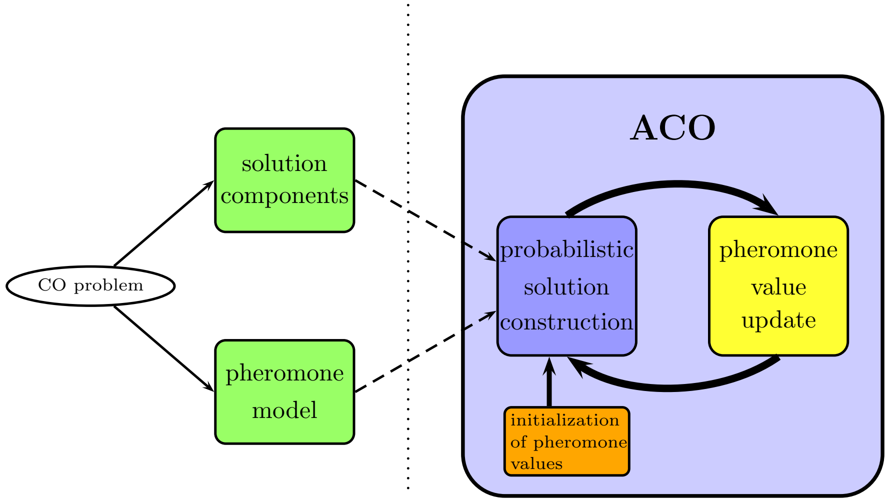
Colony size: 10 ants



Colony size 100 ants

Observation: Optimization capability is due to co-operation

The ant colony optimization metaheuristic



The ant colony optimization metaheuristic

Definition: Metaheuristic

A metaheuristic is a general strategy for tackling CO problems, which is

- not problem-specific
- approximate and usually non-deterministic
- a high-level concept to guide the search process

Goal: Efficiently explore the search space in order to find good solutions in a reasonable amount of computation time.

The ant colony optimization metaheuristic

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

The ant colony optimization metaheuristic

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

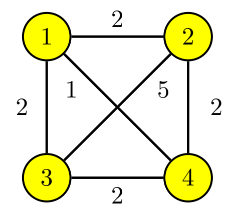
The ant colony optimization metaheuristic

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]
- Guided Local Search (GLS) [Voudouris, 1997]
- Iterated Local Search (ILS) [Stützle, 1999]
- Variable Neighborhood Search (VNS) [Mladenović, 1999]

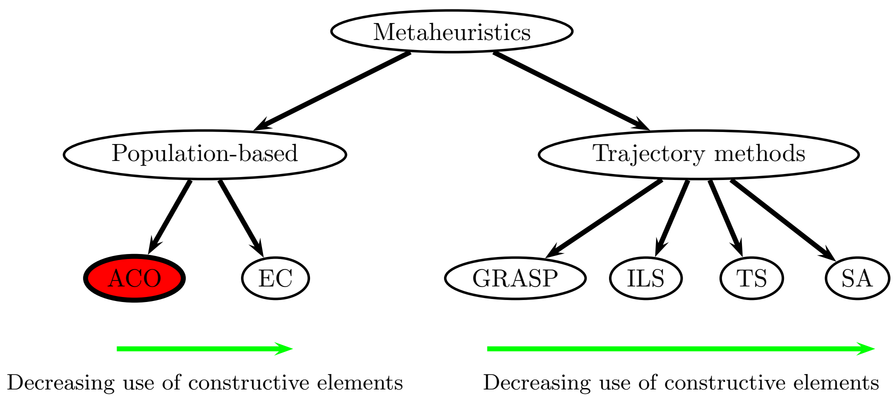
The ant colony optimization metaheuristic

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.

The ant colony optimization metaheuristic



The ant colony optimization metaheuristic

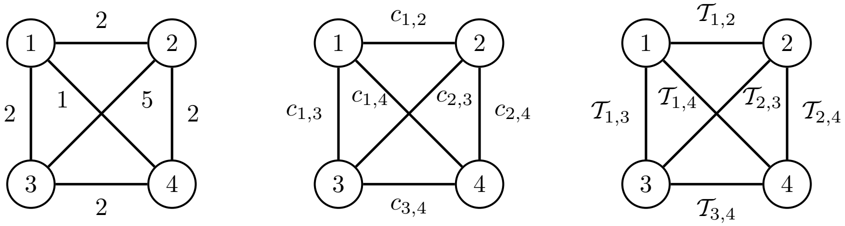
- Simulation of the foraging behaviour
- The ACO metaheuristic
- Example: traveling salesman problem (TSP)
- Example: assembly line balancing
- A closer look at algorithm components

The ant colony optimization metaheuristic

Preliminary step: Definition of the

- solution components
- pheromone model

example instance **solution components** **pheromone model**



The ant colony optimization metaheuristic

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

Pheromone evaporation **Reinforcement**

$$\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j} \qquad \tau_{i,j} \leftarrow \tau_{i,j} + \rho \cdot \sum_{\{s \in S_{iter} \mid c_{i,j} \in s\}} F(s)$$

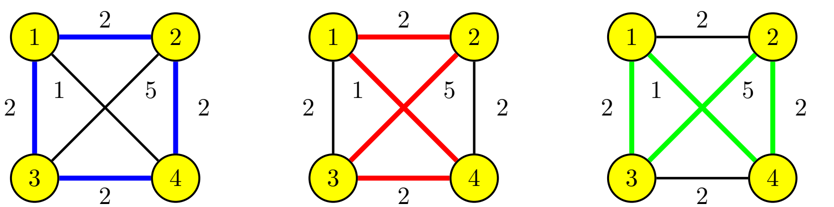
where

- 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 $F(\cdot) = \frac{1}{f(\cdot)}$

The ant colony optimization metaheuristic

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.

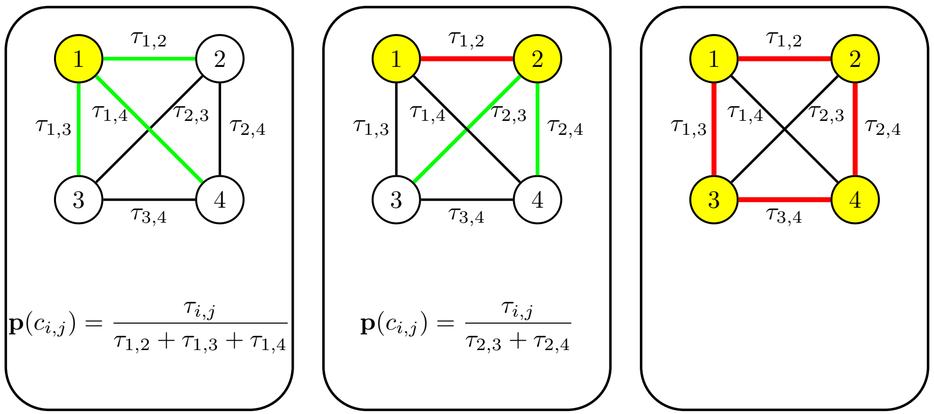


obj. function value: 8 obj. function value: 10 obj. function value: 10

The ant colony optimization metaheuristic

Tour construction:

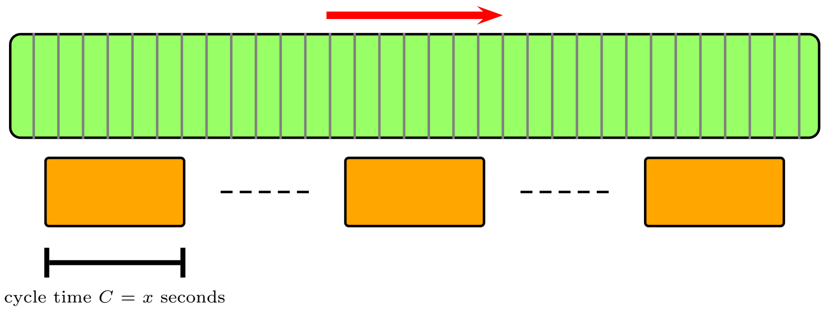
Step 1 **Step 2** **Finished**



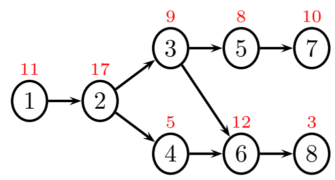
The ant colony optimization metaheuristic

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

The ant colony optimization metaheuristic

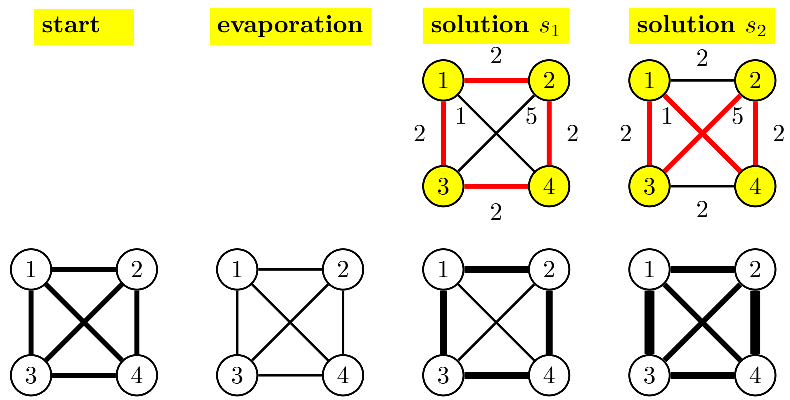


Tasks: Each task i has a time requirement t_i



The ant colony optimization metaheuristic

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



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

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

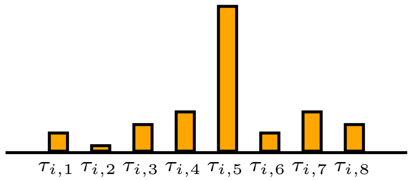
If all tasks assigned: Stop solution construction

The ant colony optimization metaheuristic

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

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

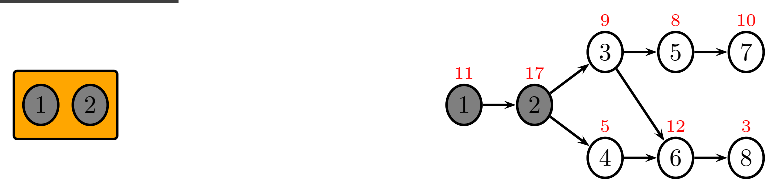
1. Solution components: We consider each possible assignment of
   - a task i
   - to a work station j

   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 $\tau_{i,j}$

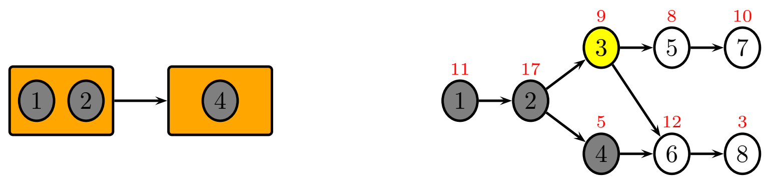
The ant colony optimization metaheuristic

Assumption: Cycle time $C = 30$ seconds

Example situation 1:



Example situation 2:



The ant colony optimization metaheuristic

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

Pheromone evaporation **Reinforcement**

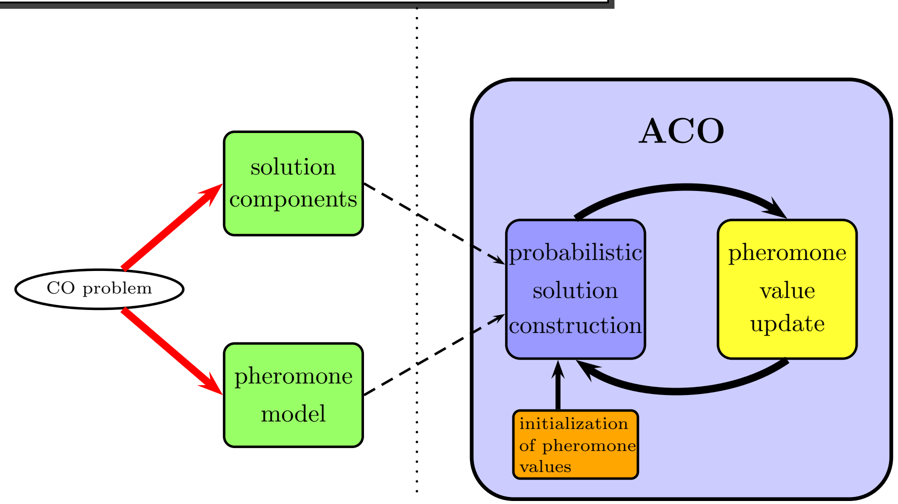
$$\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j} \qquad\qquad \tau_{i,j} \leftarrow \tau_{i,j} + \rho \cdot F(s_{ib}) \quad \forall\, c_{i,j} \in s_{ib}$$

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 $F(\cdot) = \frac{1}{f(\cdot)}$

The ant colony optimization metaheuristic

Definition of solution components and pheromone model

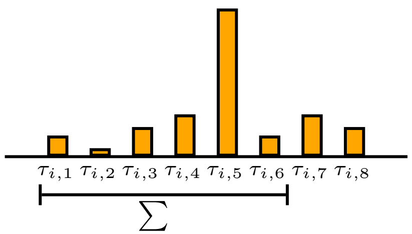


The ant colony optimization metaheuristic

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

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

The ant colony optimization metaheuristic

Definition: Discrete optimization problem

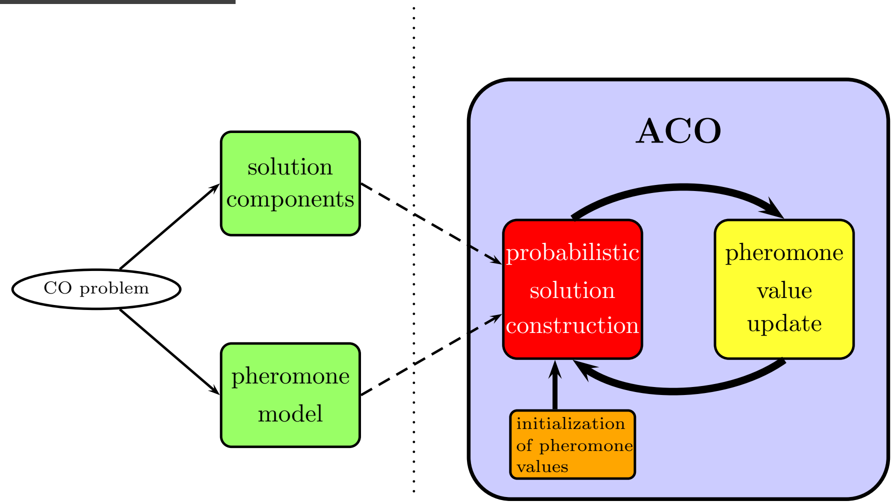
The formulation of a CO problem as a discrete optimization problem $\mathcal{P} = (\mathcal{S}, \Omega, f)$ consists of:

- a search (or solution) space $\mathcal{S}$ defined over
  - a finite set of n discrete decision variables X_i $(i = 1, \ldots, n)$;
  - and a set Ω of *constraints* among the variables;
- an *objective function* $f : \mathcal{S} \to \mathbb{R}^+$ to be minimized.

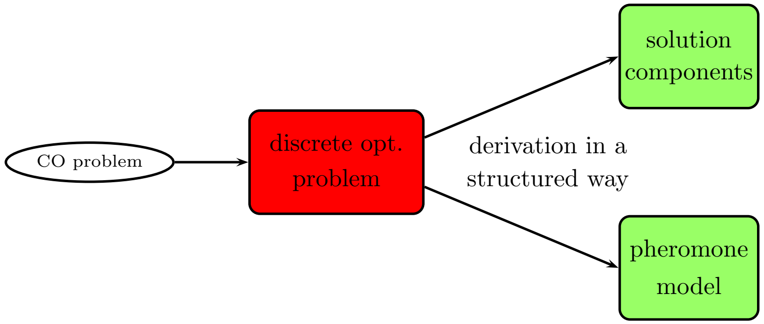
If the set of constraints Ω is empty, $\mathcal{P}$ is an unconstrained problem model, otherwise a constrained problem model.

The ant colony optimization metaheuristic

Solution construction: A closer look



The ant colony optimization metaheuristic



The ant colony optimization metaheuristic

Deriving the generic solution components:

For each combination of X_i and $v_i^j \in D_i = \{v_i^1, \ldots, v_i^{|D_i|}\}$ $\Rightarrow$ generic solution component $c_{i,j} \in C$

Deriving the pheromone model:

For each $c_{i,j} \in C$ $\Rightarrow$ pheromone trail parameter $\mathcal{T}_{i,j} \in \mathcal{T}$

The ant colony optimization metaheuristic

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

- Greedy algorithms:

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

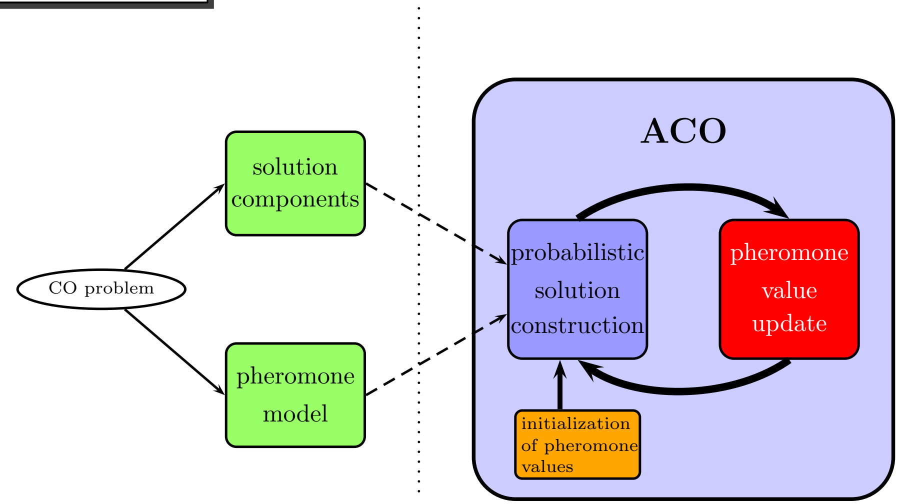
where $\eta : C \mapsto I\!R^+$ is a Greedy function

Examples for Greedy functions:

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

The ant colony optimization metaheuristic

Pheromone update: A closer look



The ant colony optimization metaheuristic

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$(N(s^p))$?

The ant colony optimization metaheuristic

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

- Ant colony optimization:

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

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

The ant colony optimization metaheuristic

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:

- 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

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?

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

Rewriting the HCF update in vector form:

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

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

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

Hybridization with other optimization techniques

The ant colony optimization metaheuristic

Successful ACO variant:

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

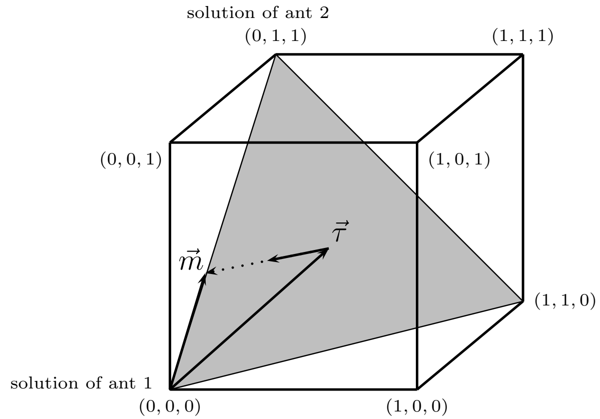
Charactersitic properties:

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

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

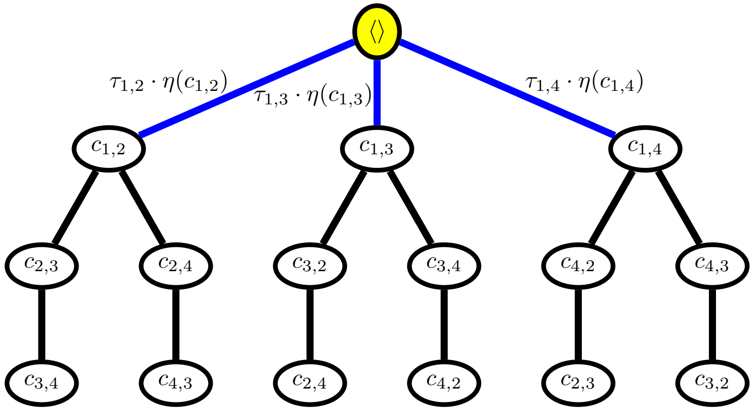
The ant colony optimization metaheuristic

Example: Problem with 3 solutions, 2 ants per iteration



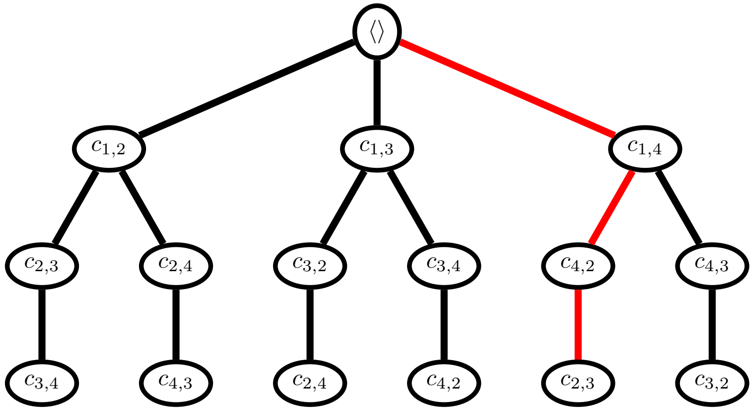
The ant colony optimization metaheuristic

ACO as a tree search algorithm: 1st construction step



The ant colony optimization metaheuristic

ACO as a tree search algorithm: 3rd construction step



The ant colony optimization metaheuristic

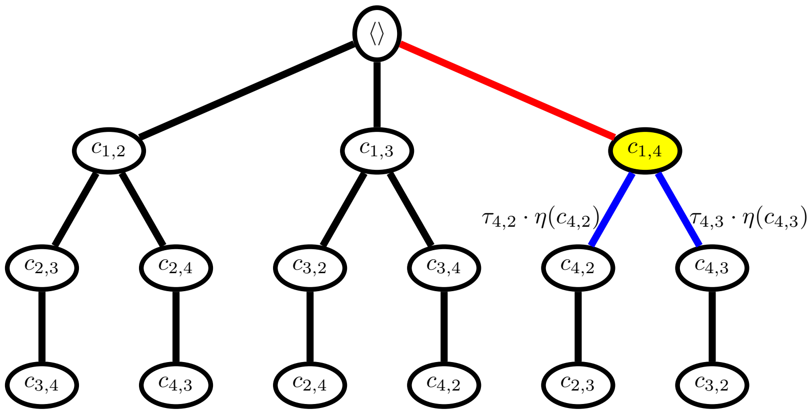
Hybridizations of ACO algorithms:

- Example 1: Hybridization with beam search [Blum, 2004]
- Example 2: Hybridization with constraint programming [Meyer, Ernst, 2004]
- Example 3: ACO and multi-level techniques [Korošec et al., 2004]
- Example 4: Applying ACO to a higher level search space [Blum, Blesa, 2005]

Important concept for 1, 2: ACO can be seen as a tree search method!

The ant colony optimization metaheuristic

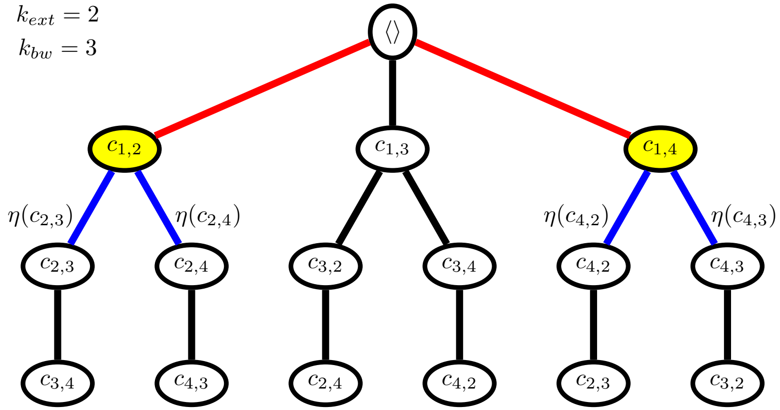
ACO as a tree search algorithm: 2nd construction step



The ant colony optimization metaheuristic

Beam search: 2nd construction step

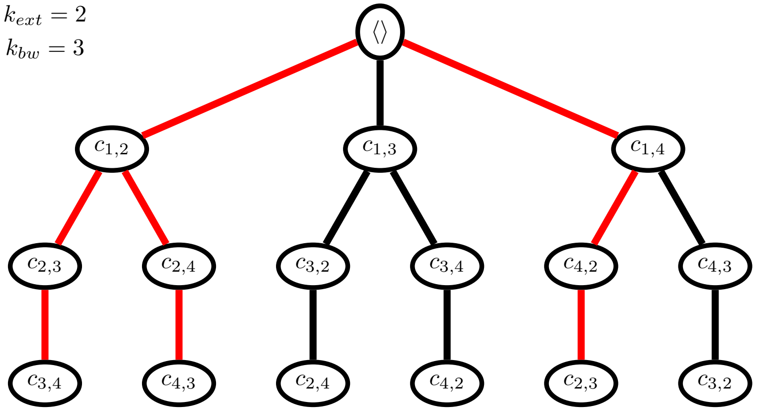
$k_{ext} = 2$
$k_{bw} = 3$



The ant colony optimization metaheuristic

Beam search: 3rd construction step

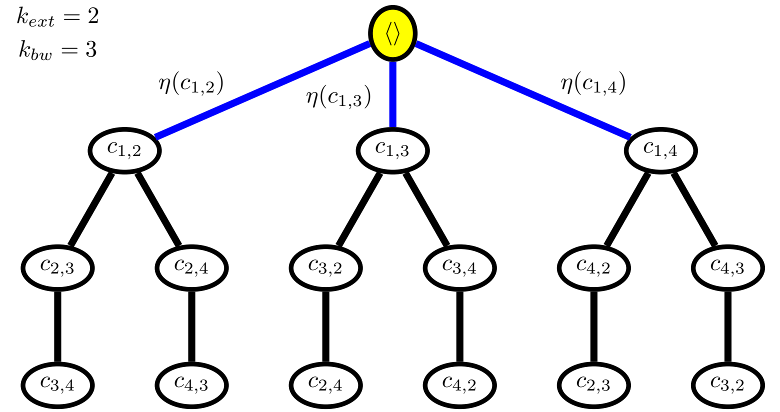
$k_{ext} = 2$
$k_{bw} = 3$



The ant colony optimization metaheuristic

Beam search: 1st construction step

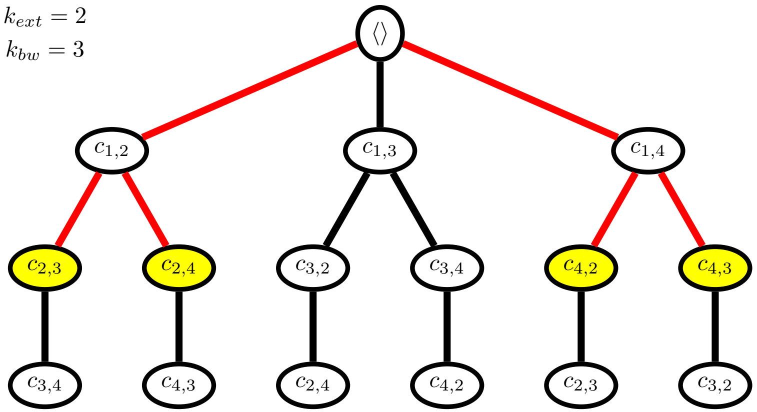
$k_{ext} = 2$
$k_{bw} = 3$



The ant colony optimization metaheuristic

Beam search: after 2nd construction step → use of lower bound

$k_{ext} = 2$
$k_{bw} = 3$



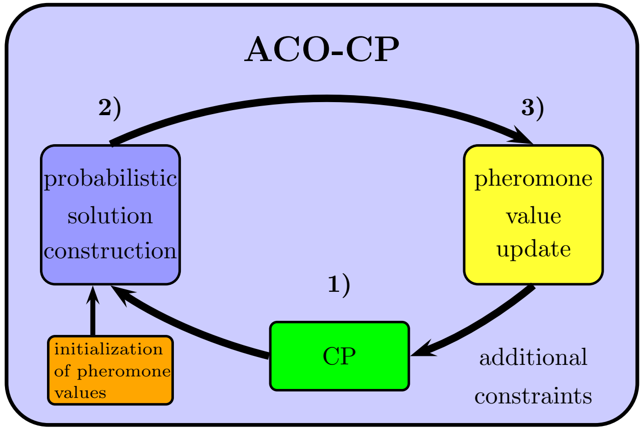
The ant colony optimization metaheuristic

Hybridizations of ACO algorithms:

- Example 1: Hybridization with beam search [Blum, 2004]
- Example 2: Hybridization with constraint programming [Meyer, Ernst, 2004]
- Example 3: ACO and multi-level techniques [Korošec et al., 2004]
- Example 4: Applying ACO to a higher level search space [Blum, Blesa, 2005]

The ant colony optimization metaheuristic

ACO-CP hybrid:



The ant colony optimization metaheuristic

Idea: Beam-ACO, in which each ant performs a probabilistic beam search

Advantages:

- Strong heuristic guidance by a lower bound
- Embedded in the adaptive framework of ACO

Result: Beam-ACO is state-of-the-art for

- open shop scheduling (OSS)
- some assembly line balancing problems

The ant colony optimization metaheuristic

Constraint programming (CP): Study of computational systems based on constraints

How does it work?

- Phase 1:
  - Express CO problem in terms of a discrete problem (variables+domains)
  - Define ("post") constraints among the variables
  - The constraint solver reduces the variable domains
- Phase 2: Labelling
  - Search through the remaining search tree
  - Possibly "post" additional constraints

The ant colony optimization metaheuristic

Hybridizations of ACO algorithms:

- Example 1: Hybridization with beam search [Blum, 2004]
- Example 2: Hybridization with constraint programming [Meyer, Ernst, 2004]
- Example 3: ACO and multi-level techniques [Korošec et al., 2004]
- Example 4: Applying ACO to a higher level search space [Blum, Blesa, 2005]

The ant colony optimization metaheuristic

Application fields of multi-level techniques:

- Originally: graph-based optimization problems
- In general:
  - ⋆ When problem instances can be contracted while maintaining characteristics
  - ⋆ When large-scale problem instances are considered

Multi-level ACO: Very good performance for mesh-partitioning.

The ant colony optimization metaheuristic

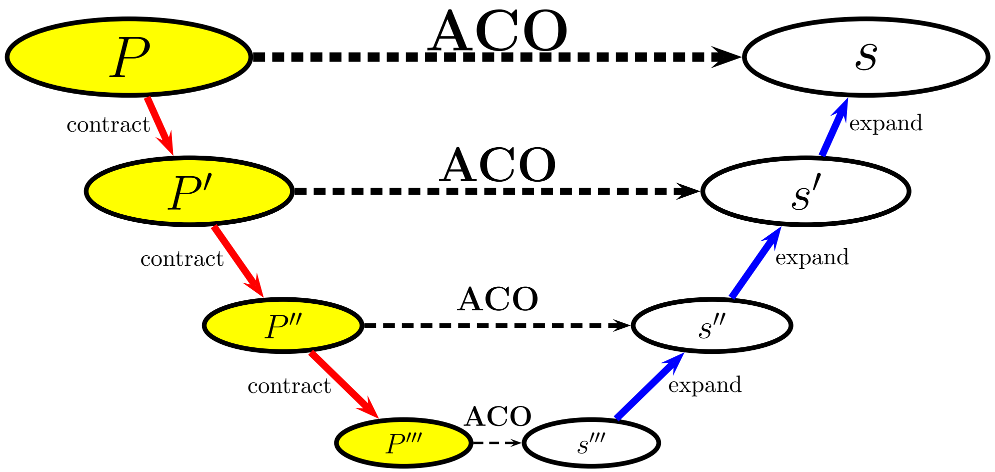
Advantages:

- Advantage of ACO:
  Good in finding high quality solutions for moderately constrained problems.
- Advantage of CP:
  Good in finding feasible solutions for highly constrained problems.

ACO-CP:
Promising for constrained problems with still a high number of feasible solutions.

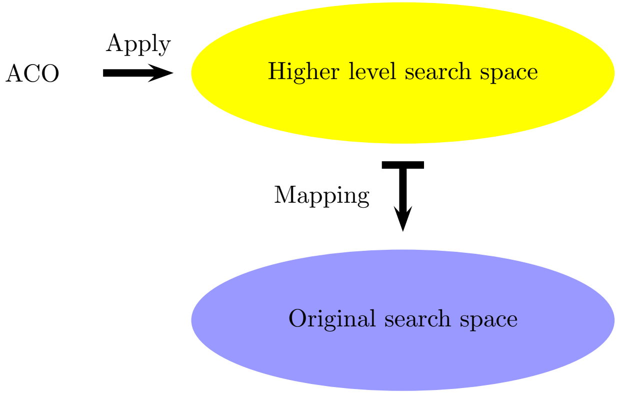
The ant colony optimization metaheuristic

The multi-level framework:



The ant colony optimization metaheuristic

General idea:



The ant colony optimization metaheuristic

Idea:

- Higher level search space: Space of all l-cardinality trees ($l > k$)
- Mapping: Use dynamic programming (Blum; 2004) to find the best k-cardinality tree in an l-cardinality tree

Note: Currently state-of-the-art for the KCT problem

The ant colony optimization metaheuristic

Hybridizations of ACO algorithms:

- Example 1: Hybridization with beam search [Blum, 2004]
- Example 2: Hybridization with constraint programming [Meyer, Ernst, 2004]
- Example 3: ACO and multi-level techniques [Korošec et al., 2004]
- Example 4: Applying ACO to a higher level search space [Blum, Blesa, 2005]

The ant colony optimization metaheuristic

Application example: The k-cardinality tree problem

Given:

- An undirected graph $G = (V, E)$,
- Edge-weights w_e, $\forall\, e \in E$, and node-weights w_v, $\forall\, v \in V$.
- A cardinality $k < |V|$

Let $\mathcal{T}_k$ be the set of all trees in G with exactly k edges.

Goal: Find a k-cardinality tree $T_k \in \mathcal{T}_k$ which minimizes

$$f(T_k) = \left(\sum_{e \in E(T_k)} w_e \right) + \left(\sum_{v \in V(T_k)} w_v \right)$$

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.~~

Negative search bias: When ACO algorithms might fail

Theoretical studies of ant colony optimization

Implicit assumptions in ACO:

Assumption 1:

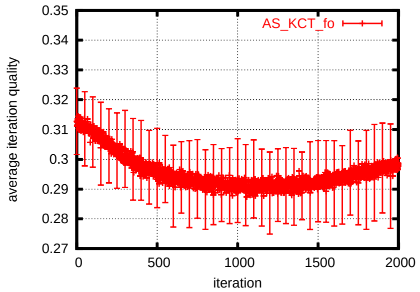
Good solutions are composed of good solution components.
(A solution component is regarded to be good, if the average quality of the solutions that contain it is high.)

Assumption 2:

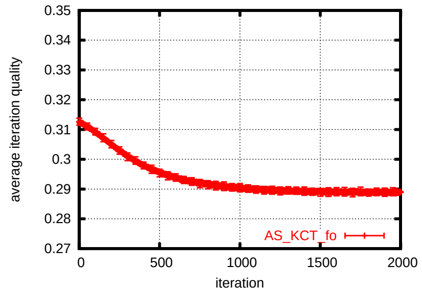
The pheromone update is such that good solution components on average are stronger reinforced than others.

Theoretical studies of ant colony optimization

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



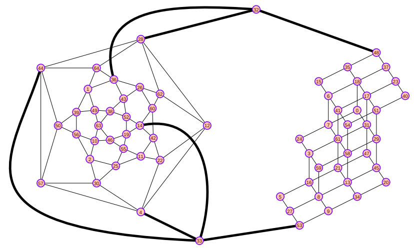
$n_a = 10$



$n_a = 1000$

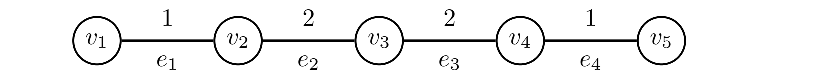
Theoretical studies of ant colony optimization

Instance statistics:

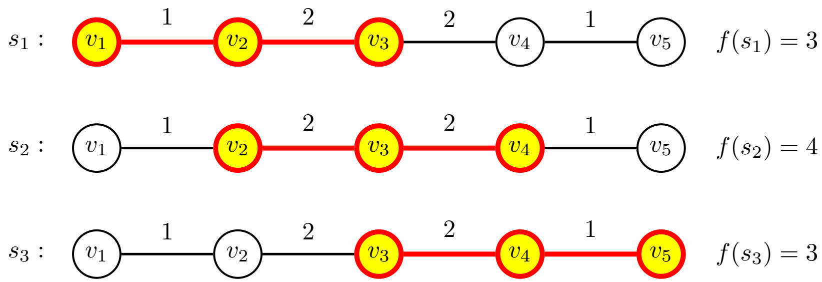


Theoretical studies of ant colony optimization

Example: 2-cardinality tree problem

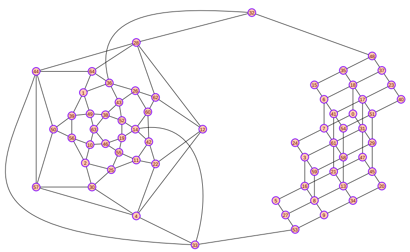


3 different solutions:

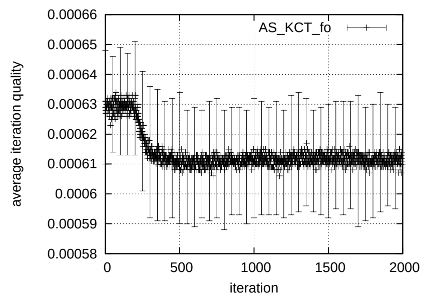


Theoretical studies of ant colony optimization

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



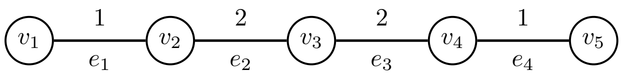
instance gd96c (65 nodes, 125 edges)



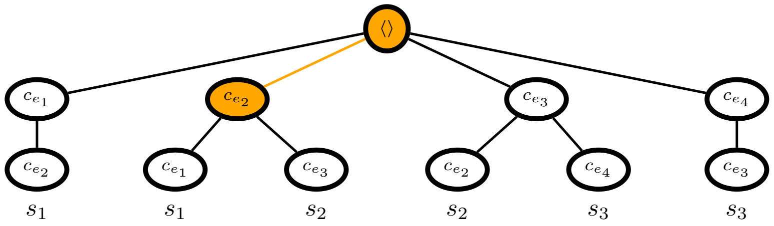
10 ants, $\rho = 0.1$, $k = 30$

Theoretical studies of ant colony optimization

Example: 2-cardinality tree problem



Search tree:



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

Ant colony optimization for continuous optimization

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

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

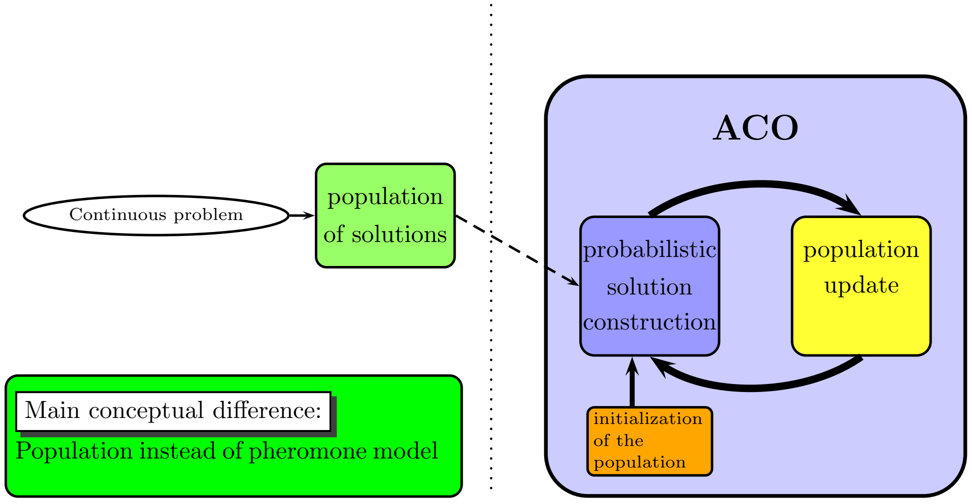
Ant colony optimization for continuous optimization

Different approaches:

- Continuous ACO (CACO) [Bilchev, Parmee, 1995]
- API [Monmarché et al., 2000]
- Contiuous Interacting Ant Colony (CIAC) [Dréo, Siarry, 2002]
- $ACO_{\mathbb{R}}$ [Socha, 2002]

Note: API and $ACO_{\mathbb{R}}$ can be applied to mixed problems

Continuous ant colony optimization



Ant colony optimization for continuous optimization

Continuous optimization

Given:

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

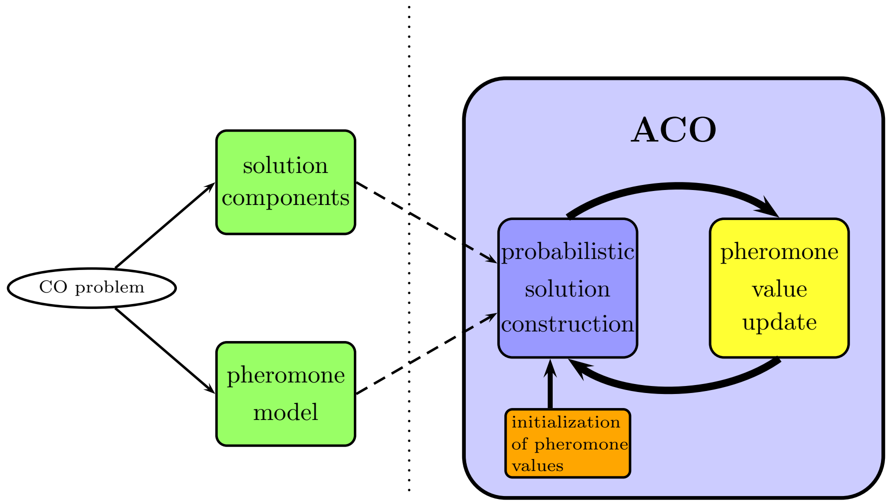
Goal: Find

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

such that

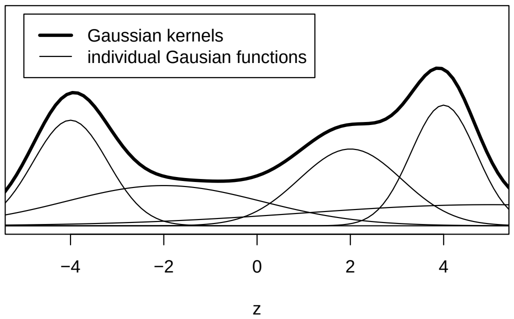
- $\vec{X}^*$ fulfills all constraints
- $f(\vec{X}^*) \leq f(\vec{Y}), \forall\, \vec{Y} \in \mathbb{R}^n$

Dicrete ant colony optimization



Continuous ACO: Probabilistic solution construction

A Gaussian kernel PDF:



Continuous ACO: Probabilistic solution construction

Choice of a Gaussian kernel:

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

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

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

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

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^*}:

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

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

Continuous ACO

Additional feature: Using the correlation between the decision variables

Standard approach: Principal Component Analysis (PCA)

- Advantage: Standard approach, works well for reasonably regular distributions
- Disadvantage: Not so good for more complex functions

Our alternative approach: For each

$$\mathbf{p}(u|j^*) = \frac{d(u, j^*)^4}{\sum_{l=1}^{k} d(l, j^*)^4}$$

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^*}

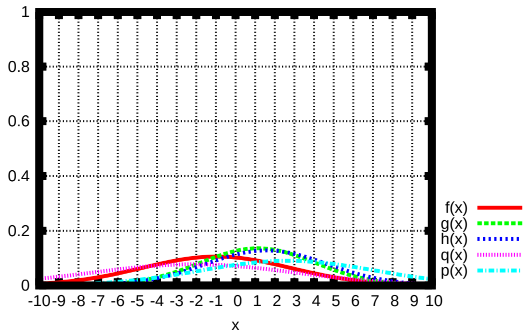
Continuous ACO

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

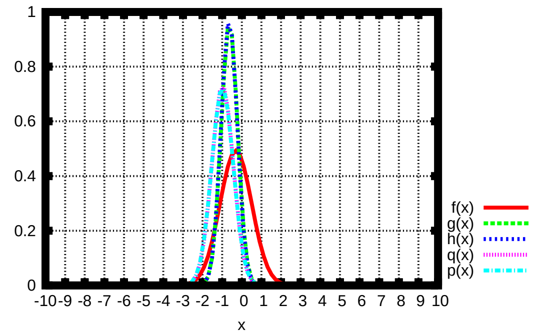
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 2

Continuous ACO

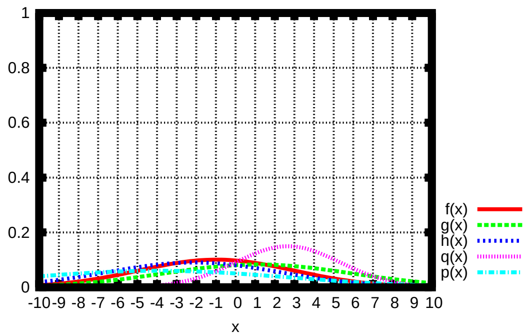
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 4

Continuous ACO

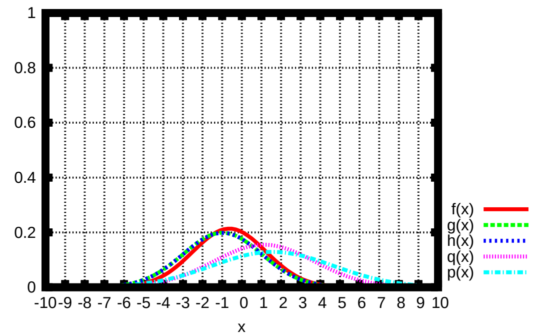
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 1

Continuous ACO

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



Iteration 3

Summary and conclusions

Presented topics:

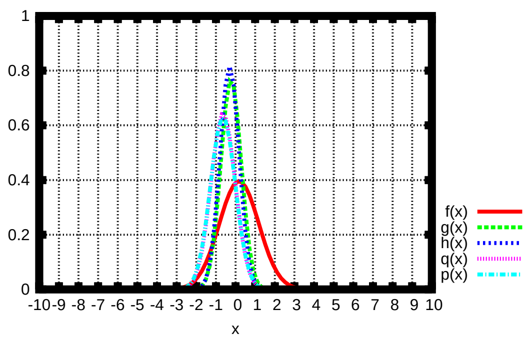
- Origins of ACO: Swarm intelligence
- How to transfer the biological inspiration into an algorithm
- Example applications of ACO: TSP and Assembly line balancing
- Hybridizations of ACO algorithms with more classical techniques
- Negative search bias
- ACO for continuous optimization

Is ACO better than other metaheuristics? **No!** (problem dependant)

Rule of thumb: ACO works well for problems for which well-working constructive heuristics exist

Continuous ACO

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



Iteration 5

Questions?