Ant Colony Optimization

Christian Blum

Albcom Research Group
Universitat Politècnica De Catalunya
Barcelona, Spain
cblum@lsi.upc.edu

Important: Due to copyright restrictions, this public set of slides lacks many photos and other additional material used in the tutorial presentation

Copyright is held by the author/owner(s).
GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

Tutorial outline (1)

Topics:

- **Swarm intelligence:** Short intro and examples
  - Self-synchronized sleep-wake periods (ants)
  - Clustering and Sorting (ants)
  - Division of Labour / Task allocation (ants + bees)
  - Self-synchronization (fireflies)
  - Flocking (birds + fish)

Tutorial outline (2)

Topics:

- **Ant colony optimization:**
  - How does it work?
  - Application example: Travelling Salesman Problem
  - Closer lock at algorithmic components
- **Ant colony optimization hybrids**
  - Hybridization with problem relaxation, bounding information, etc.
- **Ant colony optimization for continuous search spaces**

Swarm Intelligence

Short introduction and examples

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

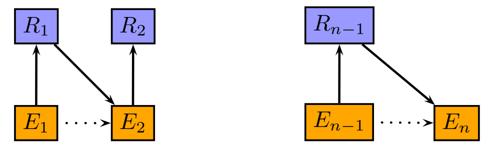
Examples of social insects:

- Ants
- Termites
- Some wasps and bees

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

Examples:

- Self-synchronized sleep-wake periods (ants)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Self-synchronized sleep-wake periods (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 sleep-wake periods (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

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Self-synchronized sleep-wake periods (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

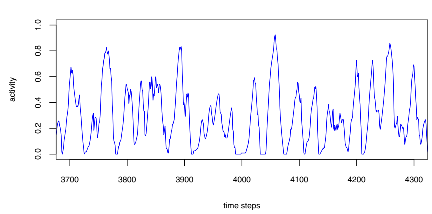
Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Self-synchronized sleep-wake periods (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



Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Self-synchronized sleep-wake periods (5)

Some references:

- 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*, 2011. In press.

Swarm intelligence: examples

Examples:

- Self-synchronized sleep-wake periods (ants)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Cemetery formation (1)

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

- E. D. Lumer and B. Faieta. **Diversity and adaptation in populations of clustering ants.** In Proceedings of the *3rd International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3 (SAB 94)*, pages 501-508. MIT Press (1994)
- D. Merkle, M. Middendorf, A. Scheidler. **Decentralized packet clustering in router-based networks**. *Int. J. Found. Comput. Sci.*, Vol. 16, No. 2, 321-341 (2005)
- J. Handl, J. Knowles and M. Dorigo. **Ant-Based Clustering and Topographic Mapping**. *Artificial Life*, Vol. 12, No. 1, Pages 35-62 (2006)

Swarm intelligence: examples

Examples:

- Self-synchronized sleep-wake periods (ants)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

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:

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

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

Division of Labour / Task Allocation (5)

Note: Response threshold models are used in

- M. Campos, E. Bonabeau, G. Theraulaz, and J.-L. Deneubourg. **Dynamic scheduling and division of labor in social insects**. *Adaptive Behavior*, Vol. 8, No. 3, 83-96 (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*, 7(2):237–254 (2008)
- H. Goldingay and J. van Mourik. **Evolution of Competing Strategies in a Threshold Model for Task Allocation**, In: *Proceedings of SNDP 2010*, Studies in Computational Intelligence Series, Springer Verlag, pages 85–98, 2010.

Swarm intelligence: examples

Examples:

- Self-synchronized sleep-wake periods (ants)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Self-synchronization of fireflies (1)

Used in:

- 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)
- O. Babaoglu, T. Binci, M. Jelasity and A. Montresor. **Firefly-inspired Heartbeat Synchronization in Overlay Networks**, In the Proceedings of the *First International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007)*, pp. 77–86 (2007)
- A. L. Christensen, R. O'Grady and M. Dorigo. **From Fireflies to Fault-Tolerant Swarms of Robots**, *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009

Swarm intelligence: examples

Examples:

- Self-synchronized sleep-wake periods (ants)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Flocking (1)

Definition: The collective motion of a large number of self-propelled 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)

Further references:

- 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)
- L. Spector, J. Klein, C. Perry, and M. Feinstein. **Emergence of Collective Behavior in Evolving Populations of Flying Agents**, Proceedings of the *Genetic and Evolutionary Computation Conference (GECCO)*, LNCS, Springer-Verlag (2003)

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Ant Colony Optimization

A metaheuristics for optimization

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Inspiration of ACO (1)

Communication strategies:

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

© Christian Blum

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

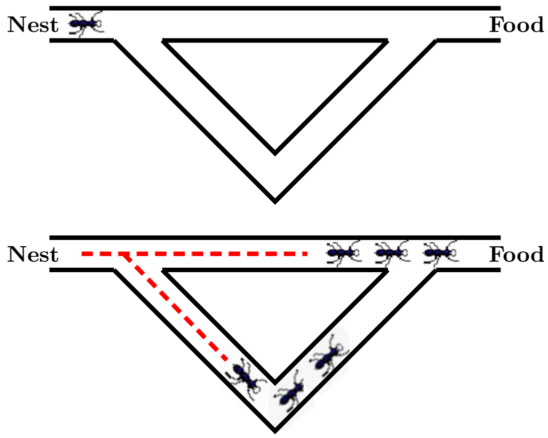
Inspiration of ACO (2)

Communication strategies:

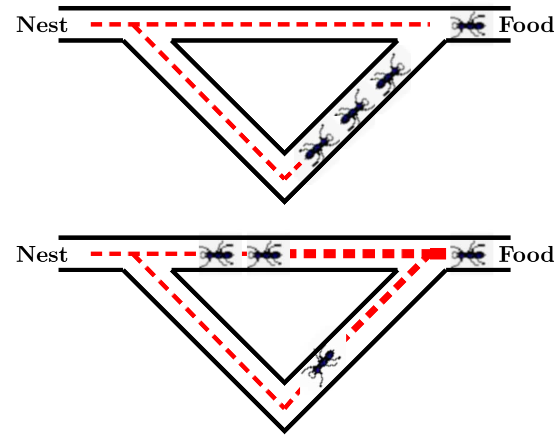
- Direct communication: For example, recruitment
- Indirect communication: via chemical pheromone trails

Basic behaviour:

Inspiration of ACO: double-bridge experiment (1)



Inspiration of ACO: double-bridge experiment (2)

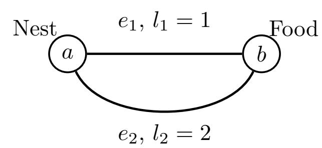


The ant colony optimization metaheuristic

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

Simulation of the foraging behaviour (1)

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:

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

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: $i = 1, 2$

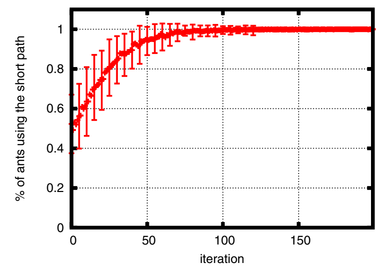
$$\tau_i \leftarrow (1-\rho)\tau_i \ , \ \rho \in (0,1]$$

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

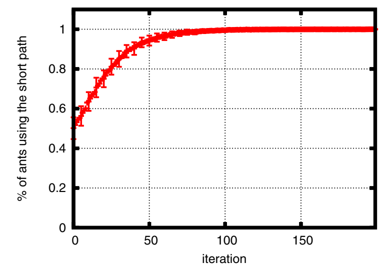
$$\tau_i \leftarrow \tau_i + \frac{1}{l_i}$$

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:

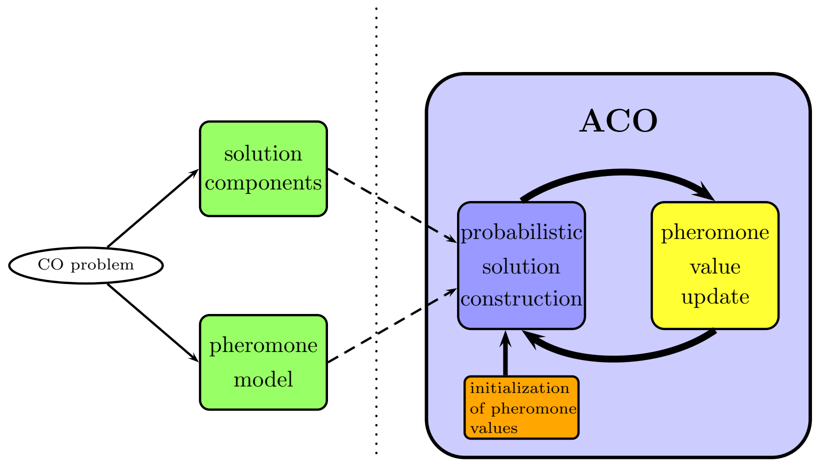
	Real ants	**Simulated ants**
Ants' movement	asynchronous	synchronized
Pheromone laying	while moving	after the trip
Solution evaluation	implicitly	explicit quality measure

Problem: In combinatorial optimization we want to find good solutions

The ant colony optimization metaheuristic

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

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:

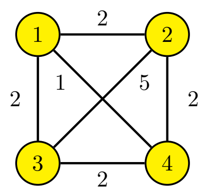
- Simulated Annealing (SA) [Kirkpatrick, 1983]
- Tabu Search (TS) [Glover, 1986]
- Genetic and Evolutionary Computation (EC) [Goldberg, 1989]
- **Ant Colony Optimization (ACO) [Dorigo, 1992]**
- Greedy Randomized Adaptive Search Procedure (GRASP) [Resende, 1995]
- Particle Swarm Optimization (PSO) [Kennedy,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

The ant colony optimization metaheuristic

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

TSP: definition (1)

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

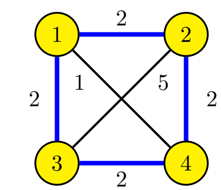


Goal: Find a tour (a Hamiltonian cycle) in G with minimal sum of edge weights.

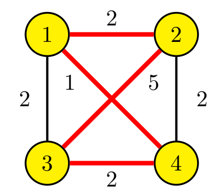
TSP definition (2)

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

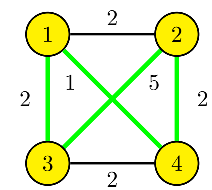
- $\mathcal{S}$ consists of all possible Hamiltonian cycles in G.
- Objective function $f : \mathcal{S} \mapsto \mathbb{R}^+$: $s \in \mathcal{S}$ is defined as the sum of the edge-weights of the edges that are in s.



obj. function value: 8



obj. function value: 10



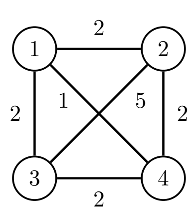
obj. function value: 10

Applying ACO to the TSP

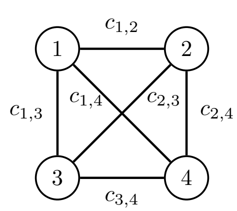
Preliminary step: Definition of the

- solution components
- pheromone model

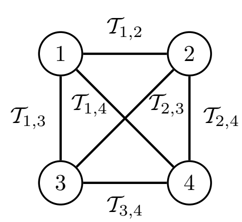
example instance



solution components



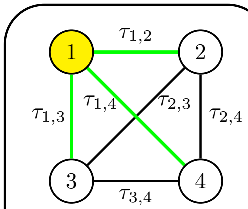
pheromone model



TSP: solution construction

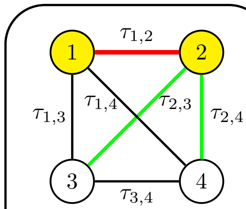
Tour construction:

Step 1



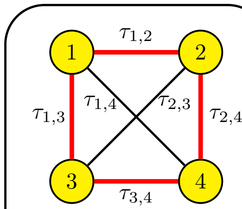
$$\mathbf{p}(c_{i,j}) = \frac{\tau_{i,j}}{\tau_{1,2} + \tau_{1,3} + \tau_{1,4}}$$

Step 2



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

Finished



TSP: pheromone update (1)

$\boxed{\text{Pheromone update:}}$ For example with the Ant System (AS) update rule

Pheromone evaporation

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

Reinforcement

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

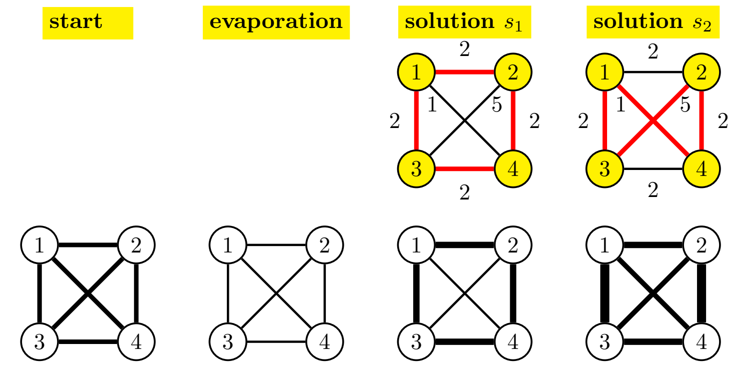
where

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

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

TSP: pheromone update (2)

$\boxed{\text{Pheromone update:}}$ For example with the Ant System (AS) update rule



Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

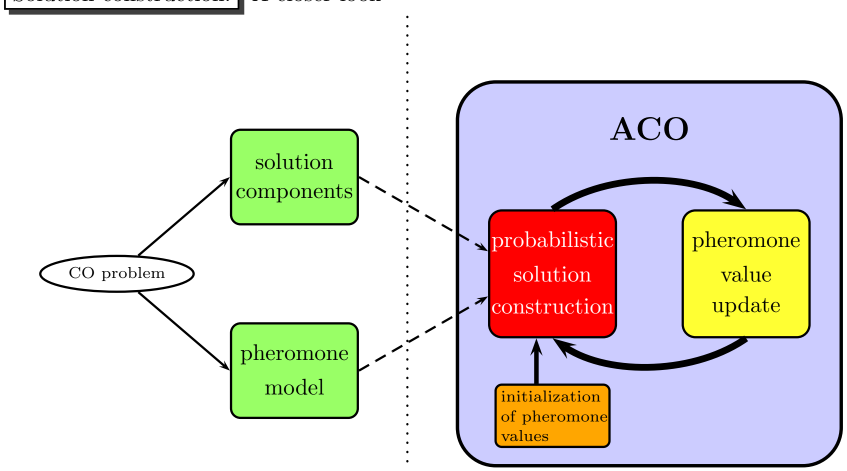
The ant colony optimization metaheuristic

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

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Solution construction (1)

$\boxed{\text{Solution construction:}}$ A closer look



Solution construction (2)

A general constructive heuristic:

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

Problem: How to implement function $\mathsf{ChooseFrom}(N(s^p))$?

Solution construction (3)

Possibilities for implementing $\mathsf{ChooseFrom}(N(s^p))$:

- Greedy algorithms:

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

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 $\mathsf{ChooseFrom}(N(s^p))$:

- Ant colony optimization:

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

where α and β are positive values

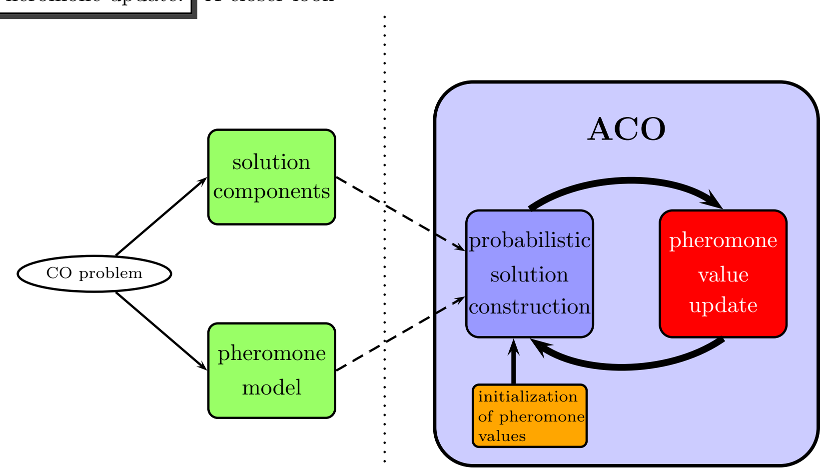
Note: α and β balance between pheromone information and Greedy function

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:

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

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

Successful ACO variants

- Ant Colony System(ACS) [Dorigo, Gambardella, 1997]

  M. Dorigo and L. M. Gambardella. **Ant colony system: a cooperative learning approach to the traveling salesman problem**. *IEEE Trans. Evolutionary Computation*, 1(1), 53–66, 1997

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

  T. Stützle and H. H. Hoos. **MAX-MIN Ant System**. *Future Generation Computer Systems*, 16(8), 889–914, 2000

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

  C. Blum and M. Dorigo. **The hyper-cube framework for ant colony optimization**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(2), 1161–1172, 2004

- Population-based ACO (P-ACO) [Guntsch, Middendorf, 2002]

  M. Guntsch and M. Middendorf. **A population based approach for ACO**. In: *Proceedings of EvoWorkshops 2002*, Springer LNCS, pages 71–80, 2002

Ant Colony Optimization

Hybridization with Other Techniques for Optimization

Ant colony optimization hybrids

Hybridizations of ACO algorithms:

- Example 1: Guiding ACO by problem relaxation
- Example 2: Using large-scale neighborhood search in ACO
- Example 3: Using bounding information in ACO
- Example 4: ACO hybridized with constraint programming

Guiding ACO by problem relaxation (1)

Reference:

- M. Reimann. **Guiding ACO by Problem Relaxation: A Case Study on the Symmetric TSP**, In: *Proceedings of HM 2007*, volume 4771, Springer LNCS, pages 45–56, 2007

Observation:

- On some benchmark instances an optimal minimum-spanning-tree (MST) solution has about 70 – 80% of the edges in common with an optimal TSP solution

Main idea: Use the MST-information to influence the solution construction

Guiding ACO by problem relaxation (2)

Solution construction mode: like nearest-neighbor heuristic

$$\mathbf{p}_{ij} = \frac{\tau_{ij} \cdot \eta_{ij}}{\sum_{k \in \Omega} \tau_{ik} \cdot \eta_{ik}}$$

where i is the current city, and Ω is the set of unvisited cities.

Heuristic information:

Standard	**Hybrid**
$\eta_{ij} = \frac{1}{d_{ij}}$	$\eta_{ij} = \frac{1+\gamma t_{ij}}{d_{ij}}$

where d_{ij} is the distance between i and j, and $t_{ij} = 1$ if edge (i, j) is part of the MST-solution, and $t_{ij} = 0$ otherwise.

Guiding ACO by problem relaxation (3)

Findings:

- Small instances: no significant difference between standard and hybrid
- Large instances:
  1. Hybrid algorithm finds best solutions faster
  2. Hybrid algorithm has a better average and worst case behaviour (statistically significant)

Evaluation:

- Application serves to introduce the idea
- In general: High potential

Guiding ACO by problem relaxation (4)

Further references:

- M. Bavafa, N. Navidi and N. Monsef. **A new approach for profit-based unit commitment using Lagrangian relaxation combined with ant colony search algorithm**, In: *Proceedings of UPEC 2008*, IEEE press, 2008
- C.-H. Chen and C. J. Ting. **Combining Lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem**, *Transportation Research Part E*, 44:1099–1122, 2008
- Z. Ren and Z. Feng. **An ant colony optimization approach to the multi-choice multi-dimensional knapsack problem**, In: *Proceedings of GECCO 2010*, pages 281–288, ACM press, 2010

Ant colony optimization hybrids

Hybridizations of ACO algorithms:

- Example 1: Guiding ACO by problem relaxation
- Example 2: Using large-scale neighborhood search in ACO
- Example 3: Using bounding information in ACO
- Example 4: ACO hybridized with constraint programming

Large-scale neighborhood search (1)

General references:

- R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen. **A survey of very large-scale neighborhood search techniques**, *Discrete Applied Mathematics*, 123(1-3):75–102, 2002
- M. Chiarandini, I. Dumitrescu, and T. Stützle. **Very Large-Scale Neighborhood Search: Overview and Case Studies on Coloring Problems**, In: *Hybrid Metaheuristics–An Emerging Approach to Optimization*, volume 114 of Studies in Computational Intelligence, pages 117–150, Springer Verlag, Berlin, Germany, 2008

Key issues in local search:

- Defining an appropriate neighborhood structure
- Choosing a way of examining the neighborhood of a solution

Large-scale neighborhood search (2)

General tradeoff:

- Small neighborhoods:
  1. **Advantage:** It is fast to find an improving neighbor (if any)
  2. **Disadvantag:** The average quality of the local minima is low
- Large-scale neighborhoods:
  1. **Advantage:** The average quality of the local minima is high
  2. **Disadvantage:** Finding an improving neighbor might itself be NP-hard due to the size of the neighborhood

Ways of examining large neighborhoods:

- Heuristically
- In some cases an efficient exact technique may exist

Using large-scale neighborhood search in ACO (1)

Specific reference:

- C. Blum and M. J. Blesa. **Combining ant colony optimization with dynamic programming for solving the k-cardinality tree problem**, In: *Proceedings of IWANN 2005*, volume 3512 of Springer LNCS, pages 25–33, 2005

Definition: 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 $k < |V|$

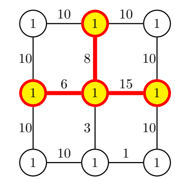
Using large-scale neighborhood search in ACO (2)

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

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

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

Example: A 3-cardinality tree



Using large-scale neighborhood search in ACO (3)

Working of a standard ACO:

- Trees are constructed step-by-step, adding one edge at a time
- To each tree is applied a 1-exchange local search algorithm
- To the iteration-best solution is applied a short run of tabu search

Main idea of the hybrid ACO:

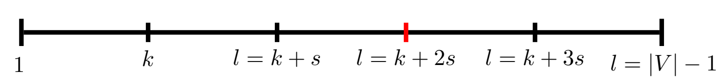
- Instead of k-cardinality trees, construct l-cardinality trees, $k < l \leq |V| - 1$
- To each l-cardinality tree: Apply an efficient dynamic programming algorithm to find the best k-cardinality tree contained in the l-cardinality tree

Using large-scale neighborhood search in ACO (4)

Findings:

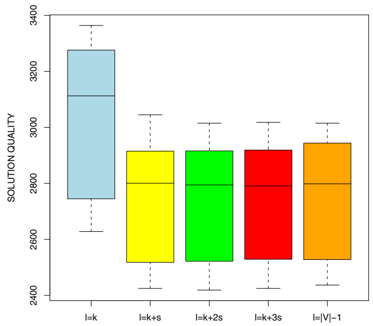
- The hybrid ACO approach outperforms consistently the standard approach
- **For small problems:** the hybrid algorithm is faster
- **For large problems:** the hybrid algorithm is better

Concerning the parameter l:



Using large-scale neighborhood search in ACO (5)

Exemplary results: 20x20 grid graphs, $k = 120$



Using large-scale neighborhood search in ACO (6)

Evaluation:

- Quite specific for KCT: Therefore, rather limited potential
- However: Might be useful for other subset problems
- General idea:
  1. Construct subsets larger than necessary
  2. Find the best subsets contained in the larger subsets

Ant colony optimization hybrids

Hybridizations of ACO algorithms:

- Example 1: Guiding ACO by problem relaxation
- Example 2: Using large-scale neighborhood search in ACO
- Example 3: Using bounding information in ACO
- Example 4: ACO hybridized with constraint programming

Using bounding information in ACO (1)

General idea: Use bounding information during the solution construction for

- ... defining/influencing the heuristic information
- ... excluding partial solutions from further examination

References: **ANTS**

- V. Maniezzo. **Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem**, *INFORMS Journal on Computing*, 11(4):358–369, 1999
- V. Maniezzo and A. Carbonaro. **An ANTS heuristic for the frequency assignment problem**, *Future Generation Computer Systems*, 16:927–935, 2000

Using bounding information in ACO (2)

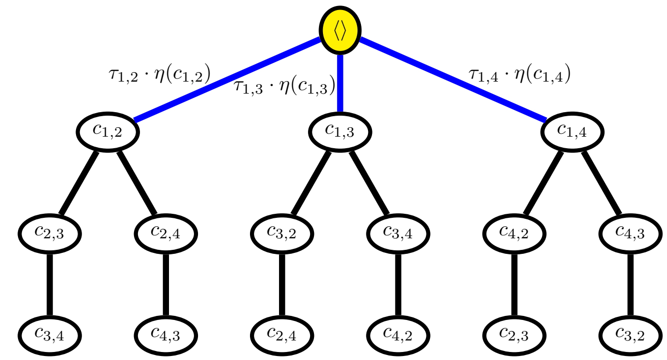
References: **Beam-ACO**

- C. Blum. **Beam-ACO–hybridizing ant colony optimization with beam search: an application to open shop scheduling**, *Computers and Operations Research*, 32:1565–1591, 2005
- J. Caldeira, R. Azevedo, C. A. Silva, and J. M. C. Sousa. **Beam-ACO Distributed Optimization Applied to Supply-Chain Management**, In: *Proceedings of IFSA 2007*, volume 4529 of Springer LNCS, pages 799–809, 2007
- C. Blum. **Beam-ACO for simple assembly line balancing**, *INFORMS Journal on Computing*, (20)4:618–627, 2008.
- M. Modarres and M. Ghandehari. **Generalized cyclic open shop scheduling and a hybrid algorithm**, *Journal of Industrial Systems Enigneering*, 1(4):345–359, 2008.

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Ant colony optimization hybrids: Beam-ACO

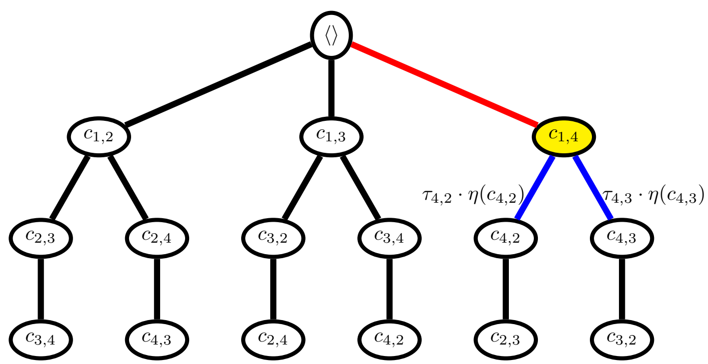
ACO as a tree search algorithm: 1st construction step



Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Ant colony optimization hybrids: Beam-ACO

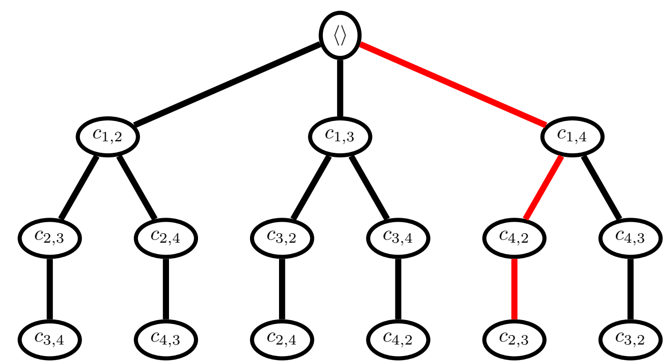
ACO as a tree search algorithm: 2nd construction step



Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

Ant colony optimization hybrids: Beam-ACO

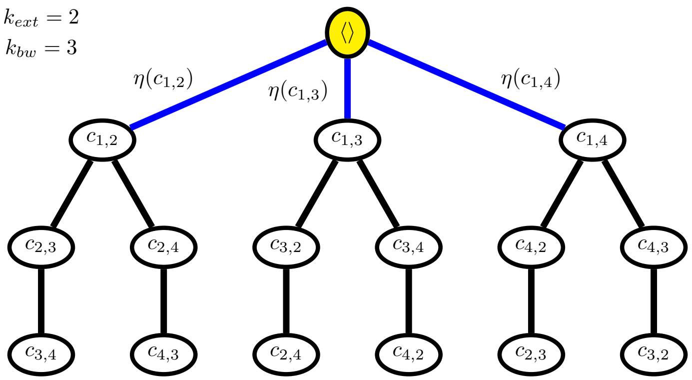
ACO as a tree search algorithm: 3rd construction step



Ant colony optimization hybrids: Beam-ACO

Beam search: 1st construction step

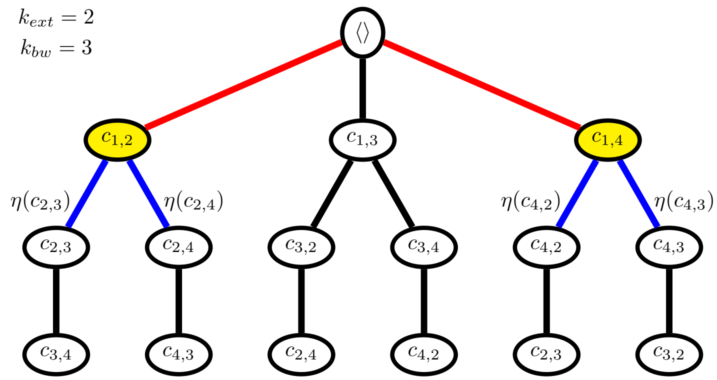
$k_{ext} = 2$
$k_{bw} = 3$



Ant colony optimization hybrids: Beam-ACO

Beam search: 2nd construction step

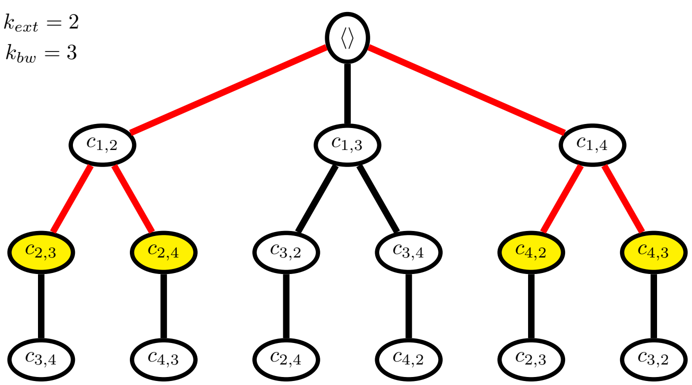
$k_{ext} = 2$
$k_{bw} = 3$



Ant colony optimization hybrids: Beam-ACO

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

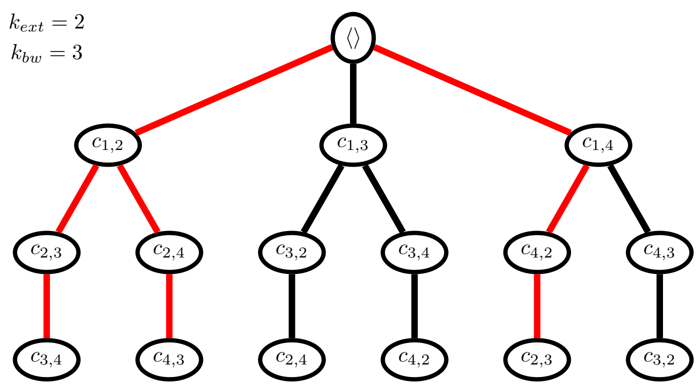
$k_{ext} = 2$
$k_{bw} = 3$



Ant colony optimization hybrids: Beam-ACO

Beam search: 3rd construction step

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



Ant colony optimization hybrids: Beam-ACO

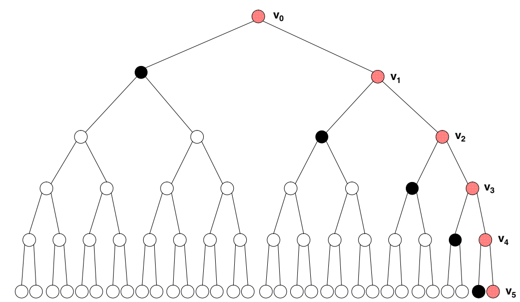
Idea of Beam-ACO: Use probabilistic beam search instead of single solution constructions

Hypothesis

It is most often beneficial to use probabilistic beam search instead of probabilistic single solution construction in construction-based metaheuristics such as **GRASP** or **ant colony optimization (ACO)**

Ant colony optimization hybrids: Beam-ACO

Intuitive example: ideal case



Ant colony optimization hybrids: Beam-ACO

Attention:

- We need black nodes close to the root node of the search tree
- We need a bound that is fast to compute
- We need a bound that does not mislead the algorithm

Evaluation: High potential for ...

- ... problems where constructive algorithms are successful
- ... local search is not especially successful

Ant colony optimization hybrids

Hybridizations of ACO algorithms:

- Example 1: Guiding ACO by problem relaxation
- Example 2: Using large-scale neighborhood search in ACO
- Example 3: Using bounding information in ACO
- Example 4: ACO hybridized with constraint programming

ACO hybridized with constraint programming (1)

References:

- B. Meyer and A. Ernst. **Integrating ACO and Constraint Propagation**, In: *Proceedings of ANTS 2004*, Springer LNCS, pages 166–177, 2004
- D. R. Thiruvady, C. Blum, B. Meyer and A. T. Ernst. **Hybridizing Beam-ACO with Constraint Programming for Single Machine Job Scheduling**, In: *Proceedings of HM 2009*, Springer LNCS, pages 30–44, 2009.
- M. Khichane, P. Albert and C. Solnon **Strong Combination of Ant Colony Optimization with Constraint Programming Optimization**, In: *Proceedings of CPAIOR 2010*, Springer LNCS, 232–245, 2010.

General idea:

- Successively reduce the variable domains by contraint propagation
- Let ACO search the reduced search tree

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

ACO hybridized with constraint programming (2)

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

Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

ACO hybridized with constraint programming (3)

Simple example: **minimize** $f(X, Y, Z) \mapsto \mathbf{R}$

subject to

$$X \in \{1, \ldots, 8\}$$
$$Y, Z \in \{1, \ldots, 10\}$$
$$X \neq 7,\ Z \neq 2$$
$$X - Z = 3Y$$

Constraint propagation:

- Step 1: Use $X \neq 7$ and $Z \neq 2$
  1. $X \in \{1, \ldots, 6, 8\}$
  2. $Y \in \{1, 3, \ldots, 10\}$

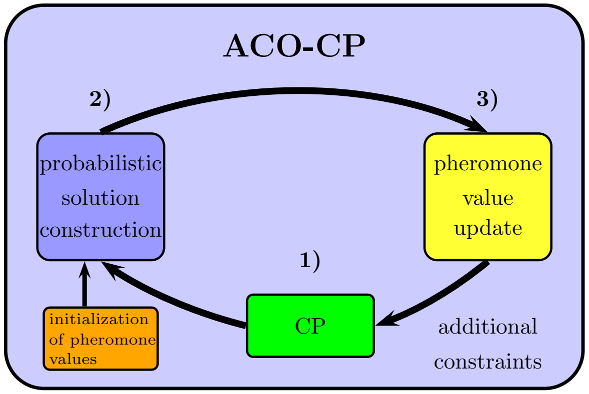
Ant Colony Optimization, July 12, 2011, GECCO 2011, Dublin, Ireland © C. Blum

ACO hybridized with constraint programming (4)

- Step 2: Use $X - Z = 3Y$
  1. Because of the domains of X and Y: $X - Z < 8$
  2. $\Rightarrow 3Y < 8$
  3. $\Rightarrow Y \leq 2$
  4. $\Rightarrow Y \in \{1, 2\}$
- Step 3: Use again $X - Z = 3Y$
  1. Because of the reduced domain of Y: $3Y \geq 3$
  2. $\Rightarrow X - Z \geq 3$
  3. $\Rightarrow X \in \{4, 5, 6, 8\}$ and $Z \in \{1, 3, 4, 5\}$

ACO hybridized with constraint programming (5)

ACO-CP hybrid:



ACO hybridized with constraint programming (6)

Evaluation:

- 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.

Other recent ACO hybrids

Some other papers on hybrids:

- P. Rocca, L. Manica and A. Massa. **Ant colony based hybrid approach for optimal compromise sum-difference patterns synthesis**, *Microwave and Optical Technology Letters*, 52(1):128–132, 2009.
- X. Hu, Q. Ding and Y. Wang. **A Hybrid Ant Colony Optimization and Its Application to Vehicle Routing Problem with Time Windows**, *Life System Modeling and Intelligent Computing*, 97(1):70–76, 2010.
- Y. Mingxin, W. Sun'an, W. Canyang and L. Kunpeng. **Hybrid ant colony and immune network algorithm based on improved APF for optimal motion planning**, *Robotica*, 28(6):833–846, 2010.
- P. S. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni. **Particle swarm and ant colony algorithms hybridized for improved continuous optimization**, *Applied Mathematics and Computation*, 188(1):129–142, 2007

Ant colony optimization for continuous 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

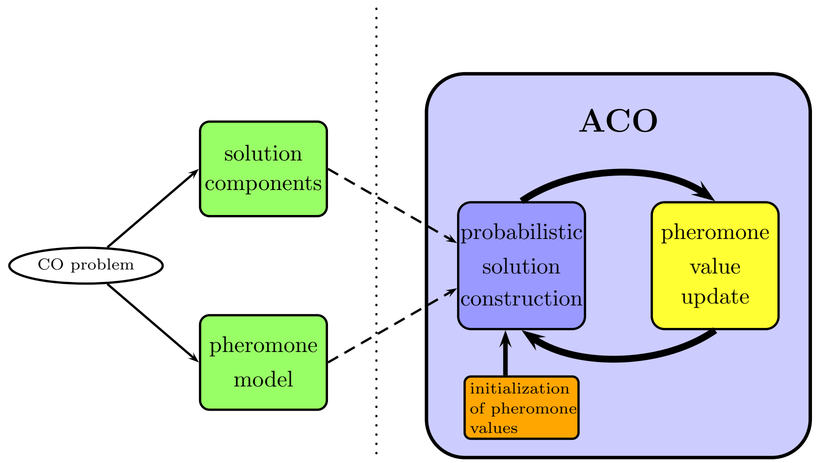
- $\vec{X}^*$ fulfills all constraints
- $f(\vec{X}^*) \leq f(\vec{Y}), \forall\, \vec{Y} \in \mathbb{R}^n$

Ant colony optimization for continuous optimization

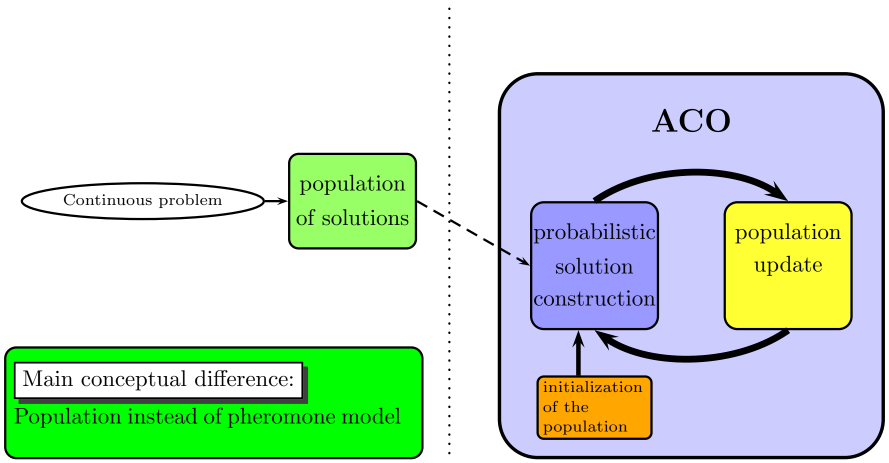
Different approaches:

- K. Socha and M. Dorigo. **Ant colony optimization for continuous domains**, *European Journal of Operational Research*, 185(3):1155–1173, 2008.
- 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.
- P. Korosec, J. Silc and B. Filipic. **The differential ant-stigmergy algorithm**, *Information Sciences*, 2011. In press.
- 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).

Dicrete 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?

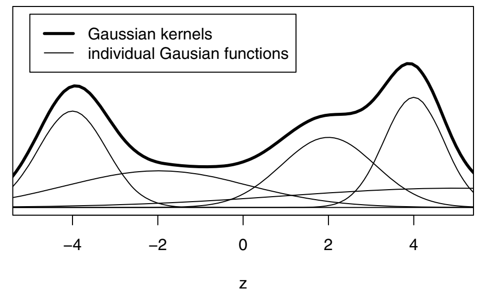
$\rightarrow$ by sampling the following Gaussian kernel probability density function (PDF):

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

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^*}:

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

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

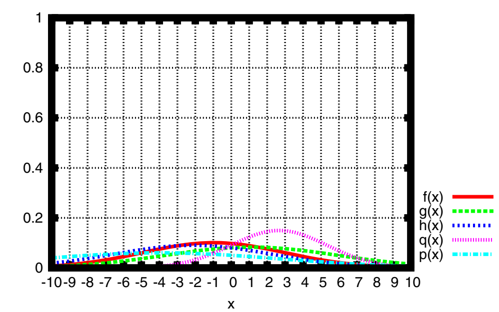
Continuous ACO

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

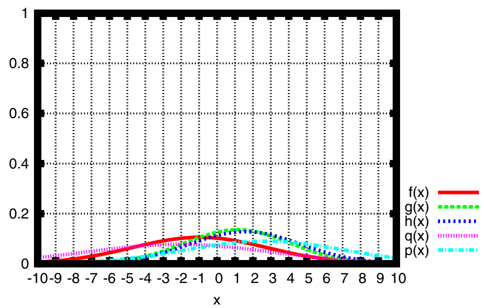
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 1

Continuous ACO

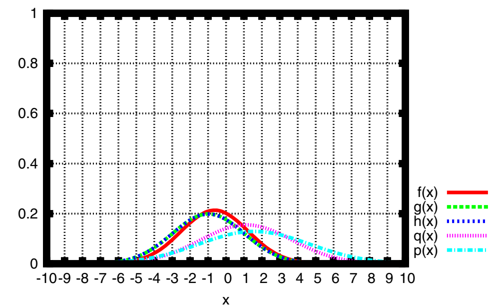
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 2

Continuous ACO

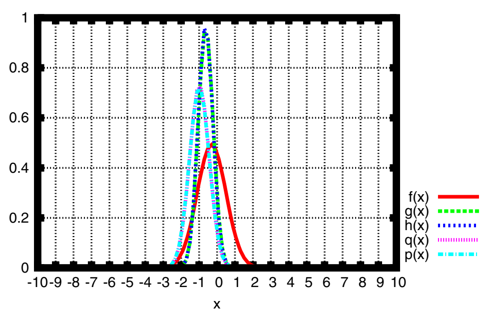
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 3

Continuous ACO

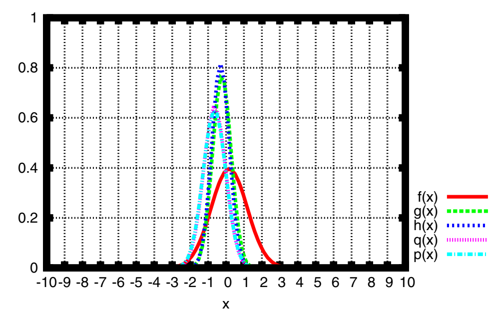
Example: $f(x) = x^2$, population size 5, 3 ants, $rho = 2.0$



Iteration 4

Continuous ACO

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



Iteration 5

Summary and conclusions (1)

Presented topics:

- Origins of ACO: Swarm intelligence
- How to transfer the biological inspiration into an algorithm
- Example application of ACO: TSP
- Hybridizations of ACO algorithms with more classical techniques
- Ant colony optimization 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

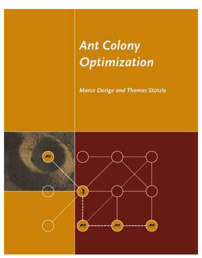
Summary and conclusions (2)

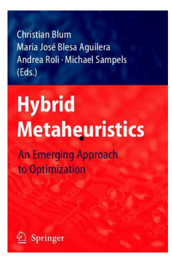
NOT presented topics:

- ACO algorithms for multi-objective optimization
  - See GECCO 2010 tutorial on ACO (by M. López-Ibáñez)
  - M. López-Ibáñez and T. Stützle. **The automatic design of multi-objective ant colony optimization algorithms**, *Technical Report TR/IRIDIA/2011-003*, 2011. Under submission.
- ACO algorithms for dynamic/stochastic problems
  - M. Mavrovouniotis and S. Yang. **Ant colony optimization with immigrants schemes in dynamic environments**, In: *Proceedings of PPSN 2010*, Springer LNCS, 2010
  - L. Bianchi, M. Dorigo, L. M. Gambardella and W. J. Gutjahr. **A survey on metaheuristics for stochastic combinatorial optimization**, *Natural Computing*, 8(2):239–287, 2009

Further Information

Books:





Papers:

- M. Dorigo and T. Stützle. **Ant colony optimization: Overview and Recent Advances**, In: *Handbook of Metaheuristics*, 227–264, Springer Verlag, 2010.
- C. Blum, J. Puchinger, G. Raidl and A. Roli. **Hybrid Metaheuristics in Combinatorial Optimization: A Survey**, *Applied Soft Computing*, 2011. In press.