Ant Colony Optimization

Christian Blum

ALBCOM, LSI,
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

Tutorial outline (2)

Topics:

- **Ant colony optimization:**
  - How does it work?
  - Application examples:
    - Traveling salesman problem
    - Assembly line balancing
  - Closer lock at algorithmic components
- **Ant colony optimization hybrids**

Tutorial outline (1)

Topics:

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

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

Swarm intelligence

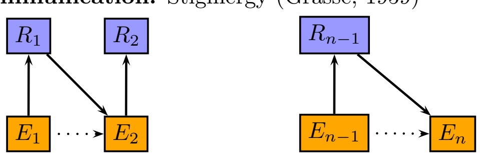
Examples of social insects:

- 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:

- Nest construction (wasps)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Nest construction (1)

Some references:

- R.L. Stewart and R.A. Russell. **A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm**. *Adaptive Behavior*, Vol. 14, No. 1, 21-51 (2006)
- A. Grushin and J.A. Reggia. **Stigmergic self-assembly of prespecified artificial structures in a constrained and continuous environment**. *Journal Integrated Computer-Aided Engineering*, Vol. 13, No. 4, 289-312 (2006)
- E. Bonabeau, S. Guerin, D. Snyers, P. Kuntz and G. Theraulaz. **Three-dimensional architectures grown by simple 'stigmergic' agents**. *Biosystems*, Vol. 56, No. 1, 13-32 (2000)

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:

- Nest construction (wasps)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Swarm intelligence: examples

Examples:

- Nest construction (wasps)
- 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:

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

Division of Labour / Task Allocation (4)

where:

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

Effects:

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

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)
- S. Nouyan, R. Ghizzioli, M. Birattari, and M. Dorigo. **An insect-based algorithm for the dynamic task allocation problem**. *Künstliche Intelligenz*, Vol. 4, 25-31 (2005)
- D. Merkle, M. Middendorf, and A. Scheidler. **Self-organized task allocation for computing systems with reconfigurable components**. In Proceedings of the *20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, page 8 pp., IEEE press (2006)

Self-synchronization of fireflies (1)

Used in:

- G. Werner-Allen, G. Tewari, A. Patel, M. Welsh and R. Nagpal. **Firefly-inspired sensor network synchronicity with realistic radio effects**, Proceedings of the *3rd International Conference on Embedded Networked Sensor Systems*, 142–153 (2005)
- 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)

Swarm intelligence: examples

Examples:

- Nest construction (wasps)
- Cemetery formation (ants)
- Division of Labour / Task allocation (ants + bees)
- Self-synchronization (fireflies)
- Flocking (birds + fish)

Swarm intelligence: examples

Examples:

- Nest construction (wasps)
- 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-propolled entities

Note:

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

Model: Basic rules

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

Ant Colony Optimization

A metaheuristics for optimization

Flocking (2)

Used in:

- 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)

Inspiration of ACO (1)

Communication strategies:

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

Photographer: Christian Blum

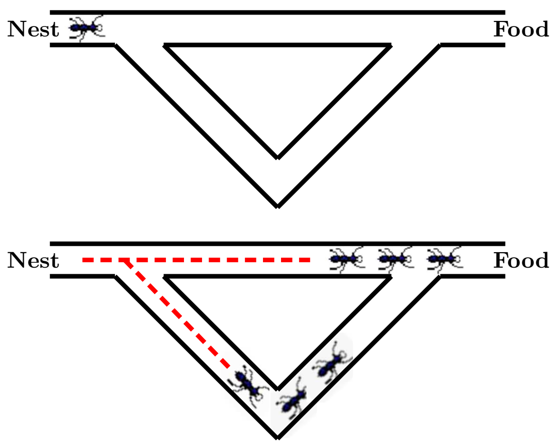
Inspiration of ACO (2)

Communication strategies:

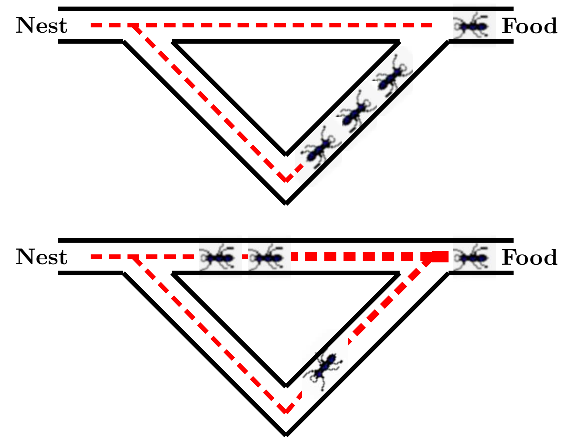
- Direct communication: For example, recruitment
- Indirect communication: via chemical pheromone trails

Basic behaviour:

Inspiration of ACO: double-bridge experiment (1)



Inspiration of ACO: double-bridge experiment (2)

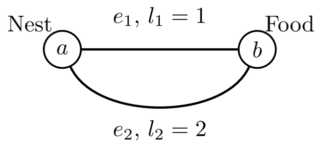


The ant colony optimization metaheuristic

- Simulation of the foraging behaviour
- The ACO metaheuristic
- Example: traveling salesman problem (TSP)
- Example: assembly line balancing
- 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:

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

Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

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: $i = 1, 2$

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

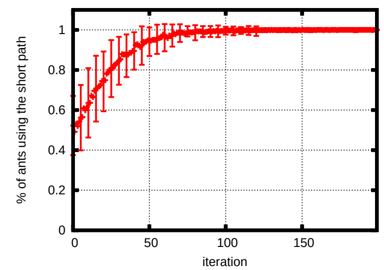
4. Each ant leaves pheromone on its traversed edge e_i:

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

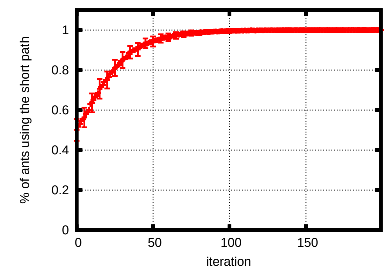
Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

Simulation of the foraging behaviour (3)

Simulation results:



Colony size: 10 ants



Colony size 100 ants

Observation: Optimization capability is due to co-operation

Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

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

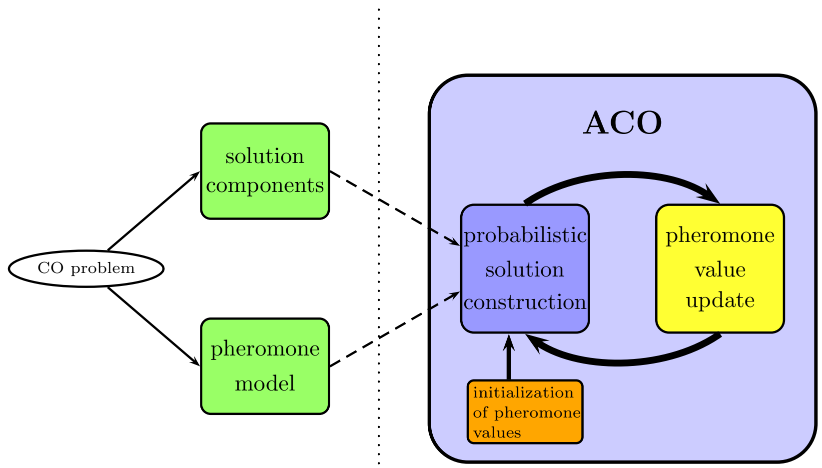
The ant colony optimization metaheuristic

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

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

The ACO framework



Metaheuristics: Timeline of their introduction

Metaheuristics:

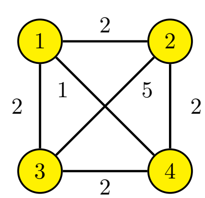
- Simulated Annealing (SA) [Kirkpatrick, 1983]
- Tabu Search (TS) [Glover, 1986]
- Genetic and Evolutionary Computation (EC) [Goldberg, 1989]
- **Ant Colony Optimization (ACO) [Dorigo, 1992]**
- Greedy Randomized Adaptive Search Procedure (GRASP) [Resende, 1995]
- Guided Local Search (GLS) [Voudouris, 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)
- Example: assembly line balancing
- 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.

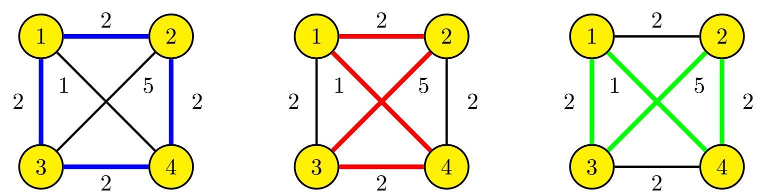


Goal: Find a tour (a Hamiltonian cycle) in G with minimal sum of edge weights.

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



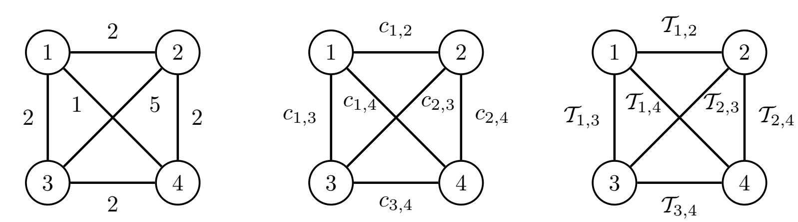
obj. function value: 8 obj. function value: 10 obj. function value: 10

Applying ACO to the TSP

Preliminary step: Definition of the

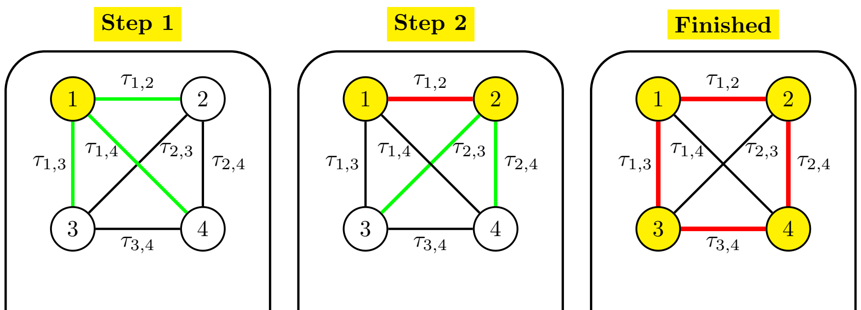
- solution components
- pheromone model

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



TSP: solution construction

Tour construction:



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

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

TSP: pheromone update (1)

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

Pheromone evaporation

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

Reinforcement

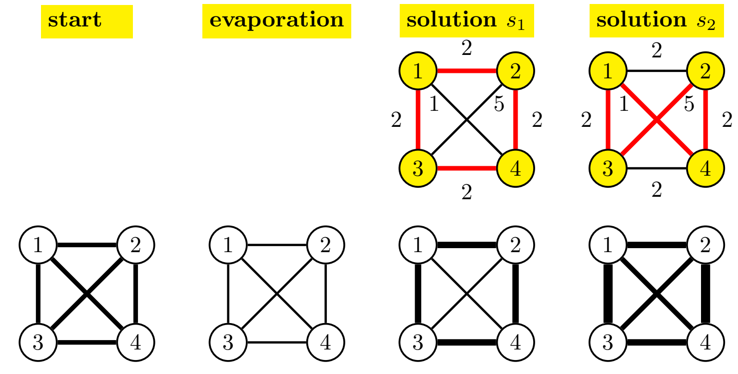
$$\tau_{i,j} \leftarrow \tau_{i,j} + \rho \cdot \sum_{\{s \in S_{iter} | c_{i,j} \in s\}} F(s)$$

where

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

TSP: pheromone update (2)

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



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

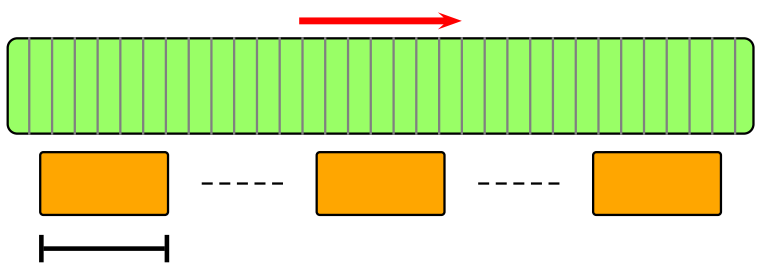
Example: assembly line balancing

Assembly line balancing

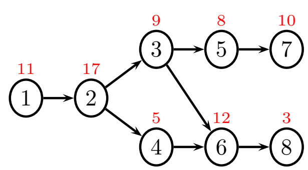
Photographer: J. Bautista

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

SALB: definition (1)



Tasks: Each task i has a time requirement t_i



SALB: definition (2)

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 $\tau_{i,j}$

SALB: solution construction

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^*

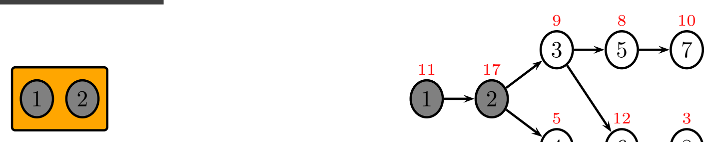
If T is empty: Open a new work station

If all tasks assigned: Stop solution construction

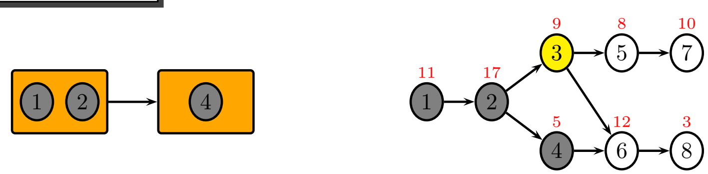
SALB: example of solution construction

Assumption: Cycle time $C = 30$ seconds

Example situation 1:



Example situation 2:



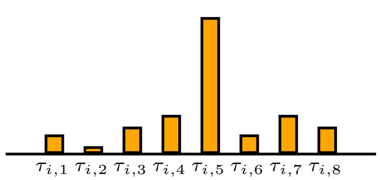
Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

SALB: transition probabilities (1)

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:



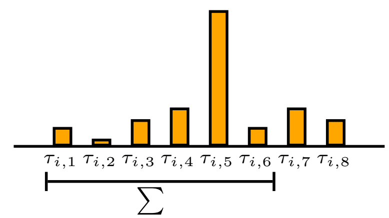
Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

SALB: transition probabilities (2)

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



Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

SALB: pheromone update

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

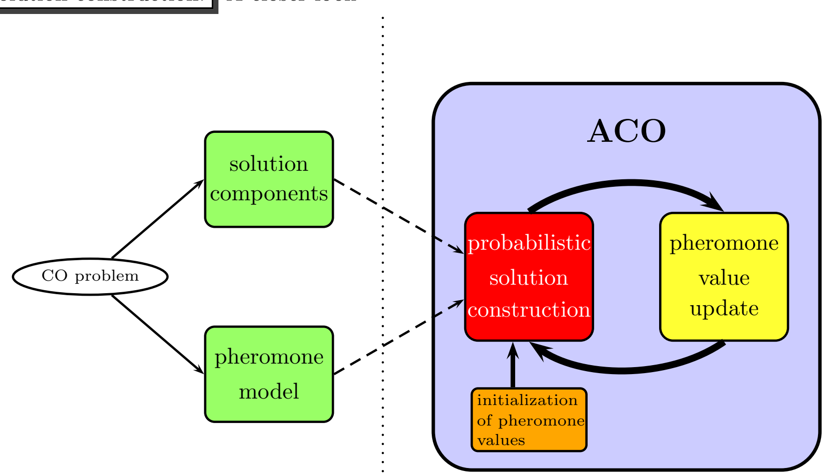
- evaporation rate $\rho \in (0, 1]$
- s_{ib} is the best solution constructed in the current iteration
- quality function $F : S \mapsto \mathbb{R}^+$. We use $F(\cdot) = \frac{1}{f(\cdot)}$

The ant colony optimization metaheuristic

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

Solution construction (1)

Solution construction: A closer look



Solution construction (2)

A general constructive heuristic:

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

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

Solution construction (3)

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

- Greedy algorithms:

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

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

Examples for Greedy functions:

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

Solution construction (4)

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

- Ant colony optimization:

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

  where α and β are positive values

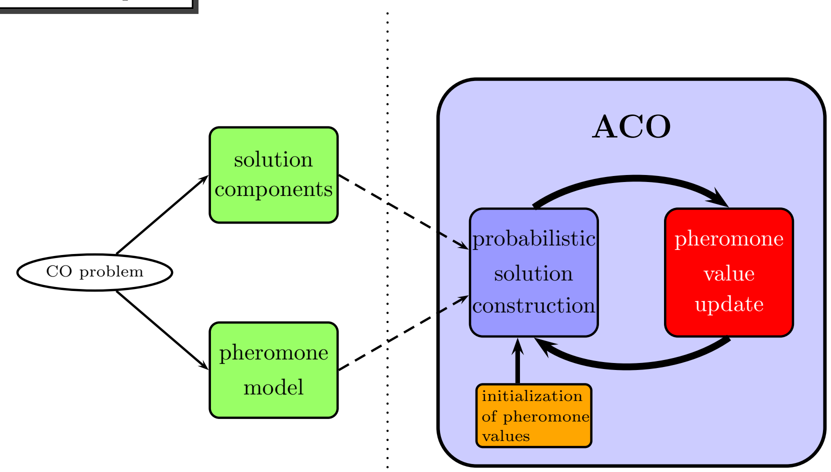
Note: α and β balance between pheromone information and Greedy function

Observations:

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

Pheromone update (1)

Pheromone update: A closer look



Pheromone update (2)

A general update rule:

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

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:	$S_{upd} \leftarrow \mathrm{argmax}\{F(s) \mid s \in S_{iter}\}$ 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($\mathcal{MM}$AS) [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

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: ACO implmented in the multi-level framework
- Example 4: Using bounding information in ACO
- Example 5: ACO hybridized with constraint programming

Ant Colony Optimization

Hybridization with Other Techniques for Optimization

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.

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: ACO implmented in the multi-level framework
- Example 4: Using bounding information in ACO
- Example 5: ACO hybridized with constraint programming

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

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 neigbhorhood

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 $k < |V|$

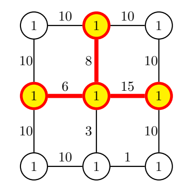
Using large-scale neighborhood search in ACO (2)

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:

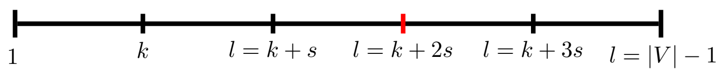
- Instead of k-cardinality trees, construct l-cardinality trees, $k < l \leq |V| - 1$
- To each l-cardinality tree: Apply an efficient dynamic programming algorithm to find the best k-cardinality tree contained in the l-cardinality tree

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

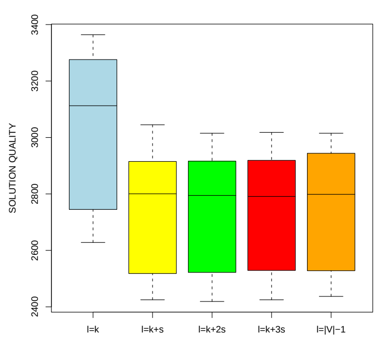
Concerning the parameter l:



Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

Using large-scale neighborhood search in ACO (5)

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



Ant Colony Optimization, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

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, July 8, 2009, GECCO 2009, Montreal, Canada © C. Blum

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: ACO implmented in the multi-level framework
- Example 4: Using bounding information in ACO
- Example 5: ACO hybridized with constraint programming

The multi-level framework (1)

General references:

- C. Walshaw. **Multilevel refinement for combinatorial optimisation**, *Annals of Operations Research*, 131:325–372, 2004
- C. Walshaw. **Multilevel refinement for combinatorial optimisation: boosting metaheuristic performance**, In: *Hybrid Metaheuristics–An Emerging Approach to Optimization*, volume 114 of Studies in Computational Intelligence, pages 261–289, Springer Verlag, Berlin, Germany, 2008

General idea:

- **First:** Iterative coarsening of the original problem instance
- **Then:** Find a solution to the coarsest level
- **Finally:** Iteratively refine this solution at each level

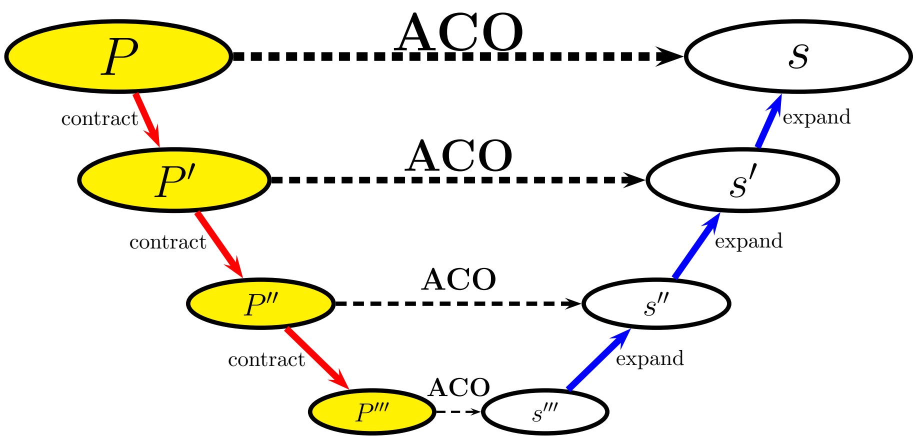
ACO implemented in the multi-level framework (1)

Specific references:

- P. Korosec, J. Silc, and B. Robic. **Solving the mesh-partitioning problem with an ant-colony algorithm**, *Parallel Computing*, 30(5-6):785-801, 2004
- M. Leng and S. Yu. **An Effective Multi-level Algorithm Based on Ant Colony Optimization for Bisecting Graphs**, In: *Proceedings of PAKDD 2007*, volume 4426 of Spriner LNAI, pages 138–149, 2007
- C. Blum, M. Yabar, and M. J. Blesa. **An ant colony optimization algorithm for DNA sequencing by hybridization**, *Computers & Operations Research*, 35:3620–3635, 2008

The multi-level framework (2)

The multi-level framework:



ACO implemented in the multi-level framework (2)

Mesh partitioning: First, transformation into graph k-partitioning

Graph k-partitioning:

- Given: $G = (V, E)$, k
- Solution: k-partition $D = \{D_i\}_1^k$ where $D_1 \cup \ldots \cup D_k = V$ and $D_i \cap D_j = \emptyset \; \forall \, i \neq j$
- Edge-cut: set of edges connecting partitions
- Goal: Find balanced D with minimal edge-cut

ACO implemented in the multi-level framework (3)

Contraction method: At each step

- find a maximal matching
- Collapse the edges involved in the matching

Expansion method: At each step

- expand the nodes containing a collapsed edge

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: ACO implmented in the multi-level framework
- Example 4: Using bounding information in ACO
- Example 5: ACO hybridized with constraint programming

ACO implemented in the multi-level framework (4)

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

Evaluation:

- High potential for problems where contraction makes sense

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

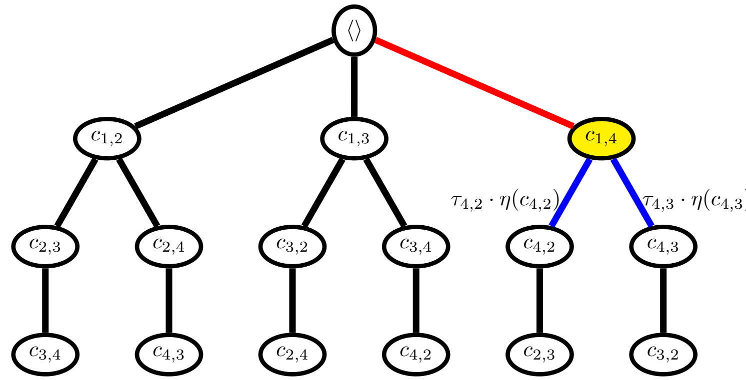
Using bounding information in ACO (2)

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
- J. Caldeira, R. Azevedo, C. A. Silva, and J. M. C. Sousa. **Supply-Chain Management Using ACO and Beam-ACO Algorithms**, In: *Proceedings of FUZZ-IEEE 2007*, pages 1–6, IEEE press, 2007
- C. Blum. **Beam-ACO for simple assembly line balancing**, *INFORMS Journal on Computing*, 2008. In press

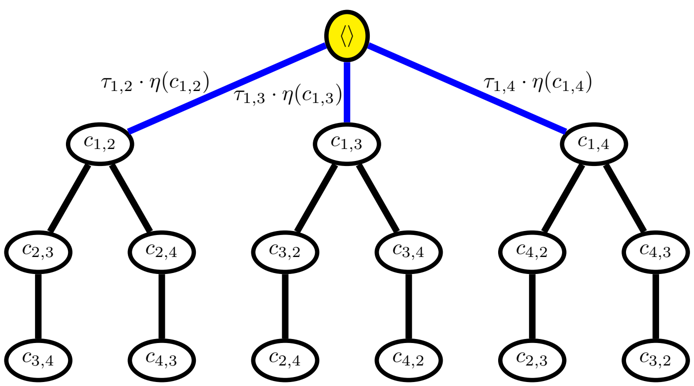
Ant colony optimization hybrids: Beam-ACO

ACO as a tree search algorithm: 2nd construction step



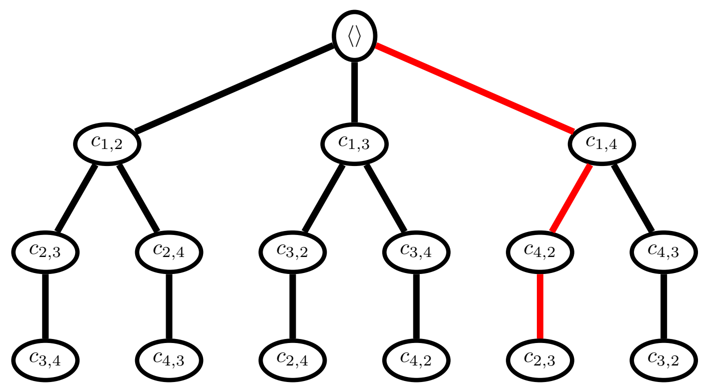
Ant colony optimization hybrids: Beam-ACO

ACO as a tree search algorithm: 1st construction step



Ant colony optimization hybrids: Beam-ACO

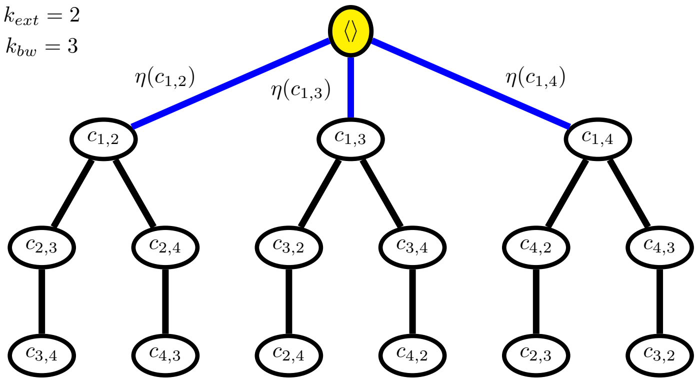
ACO as a tree search algorithm: 3rd construction step



Ant colony optimization hybrids: Beam-ACO

Beam search: 1st construction step

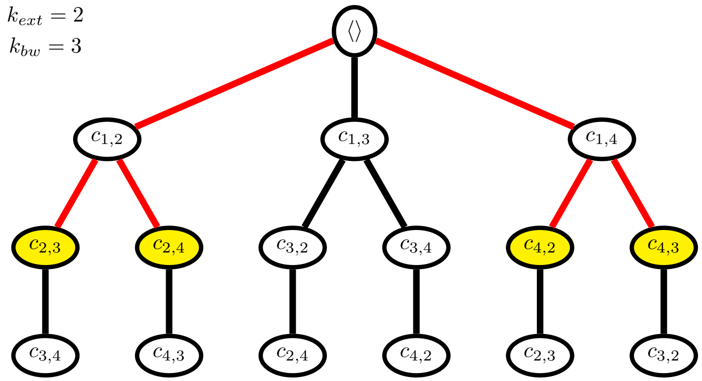
$k_{ext} = 2$
$k_{bw} = 3$



Ant colony optimization hybrids: Beam-ACO

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

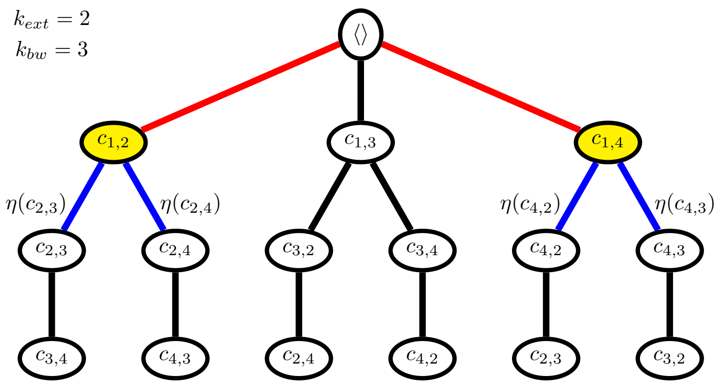
$k_{ext} = 2$
$k_{bw} = 3$



Ant colony optimization hybrids: Beam-ACO

Beam search: 2nd construction step

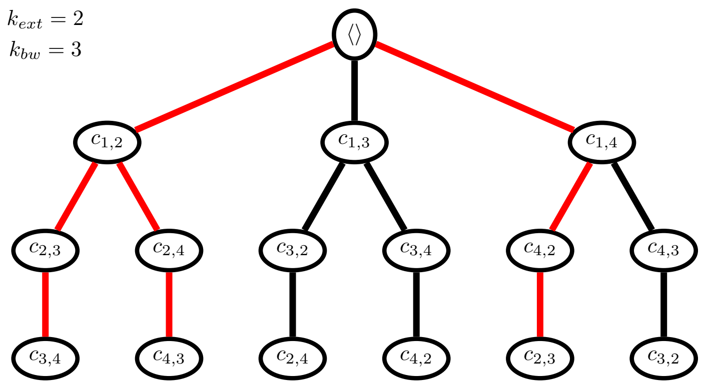
$k_{ext} = 2$
$k_{bw} = 3$



Ant colony optimization hybrids: Beam-ACO

Beam search: 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

Attention:

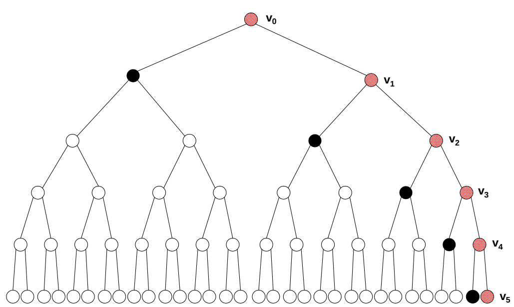
- We need black nodes close to the root node of the search tree
- We need a bound that is fast to compute
- We need a bound that does not mislead the algorithm

Evaluation: High potential for ...

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

Ant colony optimization hybrids: Beam-ACO

Intuitive example: ideal case



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: ACO implmented in the multi-level framework
- Example 4: Using bounding information in ACO
- Example 5: 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*, volume 3172 of Springer LNCS, pages 166–177, 2004
- M. Khichane, P. Albert, and C. Solnon. **CP with ACO**, In: *Proceedings of CPAIOR 2008*, volume 5015 of Springer LNCS, pages 328–332, 2008

General idea:

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

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

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:

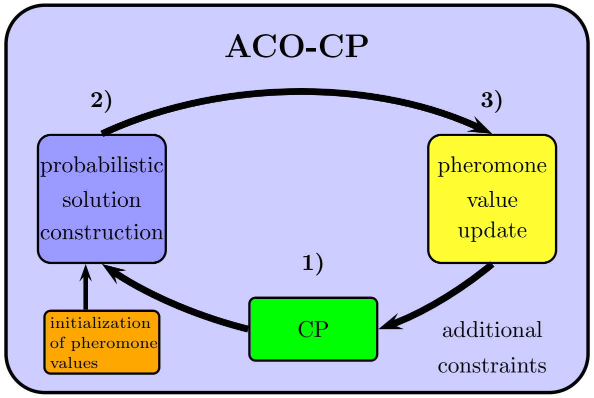
- Step 1: Use $X \neq 7$ and $Z \neq 2$
  1. $X \in \{1, \ldots, 6, 8\}$
  2. $Y \in \{1, 3, \ldots, 10\}$

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:



Other ACO hybrids

Some other papers on hybrids:

- N. Holden and A. A. Freitas. **A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data**, *In: Proceedings of SIS 2005*, pages 100–107, IEEE press, 2005
- D. M. Chitty and M. L. Hernández. **A Hybrid Ant Colony Optimisation Technique for Dynamic Vehicle Routing**, *In: Proccedings of GECCO 2004*, volume 3102 of Springer LNCS, pages 48–59, 2004
- S. Saatchi and C.-C. Hung. **Hybridization of the Ant Colony Optimization with the K-Means Algorithm for Clustering**, *In: Proceedings of SCIA 2005*, volume 3540 of Springer LNCS, pages 511–520, 2005
- 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

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.

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

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

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

Further Information