Ant Colony Optimization and Constraint Programming

Christine Solnon

LIRIS, UMR 5205 CNRS / University of Lyon

ACP summer school, 2010
Overview of the talk

1. Introduction
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. Ant Colony Optimization
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. Solving CSPs with ACO
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. Constraint Programming with ACO
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. Conclusion
Solving Combinatorial Optimization Problems (COPs)?

**Complete approaches** → Restrain combinatorial explosion

- Branch & Propagate constraints & Bound objective function
- Ordering heuristics → explore “best” branches first

Proof of optimality... but exponential time complexity in the worst case

**Heuristic approaches** → Get around combinatorial explosion

- Leave out exhaustivity → Search guided with heuristics
  - Intensify the search around the most promising areas
  - Diversify the search to discover new areas

- Two kinds of heuristic approaches
  - Perturbative heuristic approaches
    → build new combinations by perturbating existing ones
  - Constructive heuristic approaches
    → build new combinations from scratch

Polynomial time complexity... but no proof of optimality
Constructive heuristic approaches

Basic idea

- While termination conditions not reached
  - Build one or more new combination wrt a stochastic model
  - Optionally: Update the stochastic model

Well known constructive heuristic approaches

- Greedy randomized approaches
  - Static model

- Estimation of Distribution Algorithms (EDAs)
  - Dynamic model based on distribution probabilities

- Ant Colony Optimization (ACO)
  - Dynamic model based on stigmergy
Basic idea of a greedy construction

Greedy construction of a combination

- Start from an empty combination $c$
- $Cand \leftarrow$ combination components that may be added to $c$
- While $c$ is not complete do
  - Choose the best component of $Cand$ and add it to $c$
    - $\leadsto$ Choice of the component wrt a greedy heuristic
  - Update $Cand$

Quality of greedily constructed combinations

- Optimal for some problems
  - $\leadsto$ Simplex/linear problems over $\mathbb{R}$, Dijkstra/shortest path
- Not optimal (but not so bad) for many other problems
Example: The TSP

- Randomly choose a vertex \( v_j \in V \)
- \( Cand \leftarrow V - \{ v_i \} \)
- While \( Cand \neq \emptyset \) do
  - Choose \( v_j \in Cand \)
  - \( \sim \) Which greedy heuristic ???
  - \( Cand \leftarrow Cand - \{ v_j \} \)
Example: The TSP

- Randomly choose a vertex $v_i \in V$
- $Cand \leftarrow V - \{v_i\}$
- While $Cand \neq \emptyset$ do
  - Choose $v_j \in Cand$
    - Choose the vertex which is closest to the last visited one
  - $Cand \leftarrow Cand - \{v_j\}$
Greedy randomized constructions

Motivations
- Greedy constructions are very fast...
- ...But usually rather far from optimality
- ...And deterministic $\Rightarrow$ different runs return the same solution (unless ties are randomly broken)

Idea
- Introduce a slight amount of randomness
- Perform several greedy randomized constructions...
  ...And return the best constructed combinations
How to introduce randomness?

**Random choice of a component in a set of best components**

- Select the $k$ best components wrt to the greedy heuristic ...
or all components that are at $k\%$ from the best one
- Randomly choose a component in this set

$\rightsquigarrow k =$ parameter for tuning randomness

**Random choice of components wrt probabilities**

- Probability of choosing $c_i \in \text{Cand} : p(c_i) = \frac{h(c_i)^\alpha}{\sum_{c_k \in \text{Cand}} h(c_k)^\alpha}$
- where $h(c_i) =$ greedy heuristic associated with $c_i$
- $\alpha =$ parameter for tuning randomness
Overview of the talk

1. **Introduction**
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. **Ant Colony Optimization**
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. **Solving CSPs with ACO**
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. **Constraint Programming with ACO**
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. **Conclusion**
From static to dynamic models

Greedy randomized constructions

The stochastic model is static
⇝ the probability of choosing a component is always the same

- TSP: probability of using \((i, j)\) depends on \(d_{ij}\)

Idea

- Bias probabilities wrt past constructions
  ⇝ Dynamic stochastic model

  - TSP: proba. of using \((i, j)\) depends on \(d_{ij}\) and past history

- Use stigmergy to progressively bias probabilities
  ⇝ Indirect communication means through pheromone trails

  - TSP: pheromone on \((i, j)\) ⇝ past history of \((i, j)\)
Brief history of ACO

**Ant System**
[Dorigo 92]: application to the Travelling Salesman Problem

**Extensions of Ant System**
Ant Colony System [Dorigo & Gambardella 97], $\mathcal{MAX} - \mathcal{MIN}$ Ant System [Stützle & Hoos 00], Hyper-cube Ant System [Blum, Roli & Dorigo 01], ...

**Many applications**
Vehicle routing, Sequential ordering, Quadratic assignment, Graph coloring, Open shop, Maximum clique, ...

**Generalization**
Ant Colony Optimization (ACO) metaheuristic
Generic ACO framework

- initialize pheromone trails
- repeat
  1. each ant builds a combination
  2. update pheromone trails
- until optimal combination found or stagnation
Generic ACO framework

- initialize **pheromone trails**
- **repeat**
  1. each ant builds a combination
  2. update pheromone trails
- **until** optimal combination found or stagnation

**Definition of a pheromone structure**

Set of components on which pheromone trails are laid

\[ \tau(c) = \text{past experience wrt choosing } c \]
Generic ACO framework

- initialize pheromone trails

repeat

1. each ant builds a combination
2. update pheromone trails

until optimal combination found or stagnation

Greedy randomized construction of a combination

Let $C = \text{partial combination}$ and $cand = \text{candidate components}$

Choose $v_j \in cand$ with probability

$$p(v_j) = \frac{[\tau_C(v_j)]^\alpha \cdot [\eta_C(v_j)]^\beta}{\sum_{v_k \in cand} [\tau_C(v_k)]^\alpha \cdot [\eta_C(v_k)]^\beta}$$
Generic ACO framework

- initialize pheromone trails
- repeat
  1. each ant builds a combination
  2. update pheromone trails
- until optimal combination found or stagnation

Greedy randomized construction of a combination

- Let $C =$ partial combination and $cand =$ candidate components
- Choose $v_j \in cand$ with probability

$$
\rho(v_j) = \frac{[\tau_C(v_j)]^\alpha \cdot [\eta_C(v_j)]^\beta}{\sum_{v_k \in cand} [\tau_C(v_k)]^\alpha \cdot [\eta_C(v_k)]^\beta}
$$

$\tau_C(v_j) \sim$ pheromone factor (past experience of the colony)
**Generic ACO framework**

- Initialize pheromone trails
- Repeat
  1. Each ant builds a combination
  2. Update pheromone trails
- Until optimal combination found or stagnation

**Greedy randomized construction of a combination**

- Let $\mathcal{C} =$ partial combination and $\textit{cand} =$ candidate components
- Choose $v_j \in \textit{cand}$ with probability

$$p(v_j) = \frac{[\tau_{\mathcal{C}}(v_j)]^{\alpha} \cdot [\eta_{\mathcal{C}}(v_j)]^{\beta}}{\sum_{v_k \in \textit{cand}} [\tau_{\mathcal{C}}(v_k)]^{\alpha} \cdot [\eta_{\mathcal{C}}(v_k)]^{\beta}}$$

$\eta_{\mathcal{C}}(v_j) \rightsquigarrow$ heuristic factor (problem-dependent)
Generic ACO framework

- initialize pheromone trails
- repeat
  1. each ant builds a combination
  2. update pheromone trails
- until optimal combination found or stagnation

Greedy randomized construction of a combination

- Let $C =$ partial combination and $\text{cand} =$ candidate components
- Choose $v_j \in \text{cand}$ with probability

$$p(v_j) = \frac{[\tau_C(v_j)]^\alpha \cdot [\eta_C(v_j)]^\beta}{\sum_{v_k \in \text{cand}} [\tau_C(v_k)]^\alpha \cdot [\eta_C(v_k)]^\beta}$$

$\alpha, \beta \sim$ factor weights (parameters)
**Generic ACO framework**

- initialize pheromone trails
- **repeat**
  1. each ant builds a combination
  2. *update pheromone trails*
- **until** optimal combination found or stagnation

**Pheromone updating step**

- Evaporation: multiply pheromone trails by \((1 - \rho)\)
  \(\rho = \text{evaporation rate} \ (0 \leq \rho \leq 1)\)
- Reward: add pheromone on the best solution components
**Example 1: Travelling Salesman Problem**

### Pheromone structure

Pheromone is laid on edges: 
\[ \tau(i, j) \sim \text{desirability of visiting } j \text{ just after } i \]

### At each cycle, each ant builds an hamiltonian cycle

- Random choice of the first vertex
- Probability to go to \( j \) for an ant that is on vertex \( i \):
  \[
  p(j) = \frac{[\tau(i, j)]^\alpha \cdot [1/d(i, j)]^\beta}{\sum_{k \in \text{cand}} [\tau(i, k)]^\alpha \cdot [1/d(i, k)]^\beta}
  \]
  where \( \text{cand} = \text{set of non visited vertices} \)

### Pheromone updating step

- Evaporation
- Add pheromone on the edges of the best cycle
  Quantity proportionally inverse to the length of the cycle
Example 1: Travelling Salesman Problem

Pheromone structure

Pheromone is laid on edges:
\[ \tau(i, j) \sim \text{desirability of visiting } j \text{ just after } i \]

At each cycle, each ant builds an hamiltonian cycle

- Random choice of the first vertex
- Probability to go to \( j \) for an ant that is on vertex \( i \):
  \[
  p(j) = \frac{[\tau(i, j)]^\alpha[1/d(i, j)]^\beta}{\sum_{k \in \text{cand}}[\tau(i, k)]^\alpha[1/d(i, k)]^\beta}
  \]
  where \( \text{cand} = \text{set of non visited vertices} \)

Pheromone updating step

- Evaporation
- Add pheromone on the edges of the best cycle
  Quantity proportionally inverse to the length of the cycle
Example 1: Travelling Salesman Problem

**Pheromone structure**

Pheromone is laid on edges:
\[ \tau(i, j) \] desirability of visiting \( j \) just after \( i \)

**At each cycle, each ant builds an hamiltonian cycle**

- Random choice of the first vertex
- Probability to go to \( j \) for an ant that is on vertex \( i \):

\[
p(j) = \frac{[\tau(i, j)]^\alpha [1/d(i, j)]^\beta}{\sum_{k \in cand} [\tau(i, k)]^\alpha [1/d(i, k)]^\beta}
\]

where \( cand \) = set of non visited vertices

**Pheromone updating step**

- Evaporation
- Add pheromone on the edges of the best cycle
  Quantity proportionally inverse to the length of the cycle
Why (when) does it work?

Because (when) there is a correlation between the quality of a cycle and the number of edges it shares with the optimal tour.
Example 2: Maximum Clique Problem

Pheromone structure

Pheromone is laid on vertices
\[ \tau(i) \rightsquigarrow \text{desirability of selecting } i \text{ in a clique} \]

At each cycle, each ant builds a maximal clique

- Random choice of the first vertex
- Probability to select vertex \( i \):
  \[ p(i) = \frac{[\tau(i)]^\alpha}{\sum_{k \in \text{cand}} [\tau(k)]^\alpha} \]
  where \( \text{cand} = \text{set of vertices connected to all selected vertices} \)

Pheromone updating step

- Evaporation
- Add pheromone on the vertices of the best clique
  Quantity proportional to the size of the clique
Example 2: Maximum Clique Problem

**Pheromone structure**

Pheromone is laid on vertices

\[ \tau(i) \leadsto \text{desirability of selecting } i \text{ in a clique} \]

---

**At each cycle, each ant builds a maximal clique**

- Random choice of the first vertex
- Probability to select vertex \( i \):

\[
p(i) = \frac{[\tau(i)]^\alpha}{\sum_{k \in \text{cand}} [\tau(k)]^\alpha}
\]

where \( \text{cand} = \text{set of vertices connected to all selected vertices} \)

---

**Pheromone updating step**

- Evaporation
- Add pheromone on the vertices of the best clique
  Quantity proportional to the size of the clique
Example 2: Maximum Clique Problem

**Pheromone structure**

Pheromone is laid on vertices
\( \tau(i) \rightarrow \text{desirability of selecting } i \text{ in a clique} \)

**At each cycle, each ant builds a maximal clique**

- Random choice of the first vertex
- Probability to select vertex \( i \):
  \[
  p(i) = \frac{[\tau(i)]^\alpha}{\sum_{k \in \text{cand}}[\tau(k)]^\alpha}
  \]
  where \( \text{cand} = \text{set of vertices connected to all selected vertices} \)

**Pheromone updating step**

- Evaporation
- Add pheromone on the vertices of the best clique
  Quantity proportional to the size of the clique
Hybridization ACO / Local Search

Two views of an ACO/LS hybrid approach

- Ants build initial solutions...
  ... that are improved by local search;
- or LS finds locally optimal solutions...
  ... that are used by ACO to build new starting points.

Choice of a local search strategy

- Compromise between CPU time and solution quality
- Usually: simple greedy LS
Overview of the talk

1. **Introduction**
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. **Ant Colony Optimization**
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. **Solving CSPs with ACO**
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. **Constraint Programming with ACO**
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. **Conclusion**
Intensifying/diversifying search with ACO

Intensification

- Goal: Increase search around promising areas
- Means:
  - Add pheromone on components of best solutions
  - Favor the choice of components with high pheromone trails
- Risk: Premature convergence (stagnation)

Diversification

- Goal: Explore new areas
- Means:
  - Probabilistic choice of components
  - Bound pheromone trails within $[\tau_{min}, \tau_{max}]$
  - Initialize pheromone trails to $\tau_{max}$
- Risk: convergence to optimality may be too long

⇒ Theoretical proof of convergence to optimality
Intensifying/diversifying search with ACO

Influence of the parameters

- $\tau_{\text{min}}, \tau_{\text{max}}$: pheromone lower and upper bounds
  - $\Rightarrow$ Intensification increases when $\tau_{\text{max}} - \tau_{\text{min}}$ increases

- $\text{nbAnts}$: number of ants
  - $\Rightarrow$ Diversification increases when $\text{nbAnts}$ increases

- $\alpha$: weight of the pheromone factor
  - $\Rightarrow$ Intensification increases when $\alpha$ increases

- $\rho$: pheromone evaporation rate
  - $\Rightarrow$ Intensification increases when $\rho$ increases
Balancing intensification/diversification

The best parameter setting depends on the instance to solve:

- Intensify if there is a correlation between
  - the quality of a combination, and
  - the number of pheromone components shared with the optimal combination

- Diversify otherwise.
Quality/pheromone correlation for \texttt{gen200\_p0.9\_55}

![Graph showing the correlation between the number of vertices shared with the maximum clique and the size of the clique for \texttt{gen200\_p0.9\_55}.

The graph plots the number of vertices shared with the maximum clique on the y-axis against the size of the clique on the x-axis. The correlation is indicated by a scatter plot with stars marking the data points.

- The x-axis ranges from 20 to 55, representing the size of the clique.
- The y-axis ranges from 0 to 60, representing the number of vertices shared with the maximum clique.

The data points for \texttt{gen200\_p0.9\_55} show a positive correlation, with the number of vertices shared increasing as the size of the clique increases.
Influence of parameters for $\text{gen400}_{p0.965}$

- $\alpha = 0$ (no pheromone)
- $\alpha = 1$ $\rho = 0.005$
- $\alpha = 1$ $\rho = 0.01$
- $\alpha = 2$ $\rho = 0.02$

Number of cycles (logscale)

Size of the best clique (average on 50 runs)
Quality/pheromone correlation for $C_{125}$

![Quality/pheromone correlation for $C_{125}$](image_url)
Influence of parameters for C500

- **alpha = 0 (no pheromone)**
- **alpha = 1 rho = 0.005**
- **alpha = 1 rho = 0.01**
- **alpha = 2 rho = 0.02**

The graph shows the size of the best clique (average on 50 runs) as a function of the number of cycles (logscale). The different lines correspond to varying values of alpha and rho parameters.
Quality/pheromone correlation for brock200_4
Influence of parameters for brock400_4
Measuring Intensification/Diversification

Resampling ratio (RR) $\sim$ quantifies diversification

$$RR = \frac{|\text{computed solutions}| - |\text{different computed solutions}|}{|\text{computed solutions}|}$$

Maximal diversification $\Leftarrow 0 \leq RR \leq 1 \Rightarrow$ Stagnation

Similarity ratio (SR) $\sim$ quantifies intensification

$$SR = \text{average similarity of the set } S \text{ of computed solutions}$$
$$\sim \text{average similarity of pairs of solutions of } S$$
$$\sim \text{similarity of 2 solutions} = \text{percentage of shared components}$$

SR increases when search is intensified

These 2 ratio may be computed (nearly) for free with appropriate data structures!
Measuring Intensification/Diversification

**Resampling ratio (RR)** quantifies diversification

\[ RR = \frac{\#\{\text{computed solutions}\} - \#\{\text{different computed solutions}\}}{\#\{\text{computed solutions}\}} \]

Maximal diversification \( 0 \leq RR \leq 1 \) → Stagnation

**Similarity ratio (SR)** quantifies intensification

\[ SR = \text{average similarity of the set } S \text{ of computed solutions} \]
\[ \sim \text{average similarity of pairs of solutions of } S \]
\[ \sim \text{similarity of 2 solutions} = \text{percentage of shared components} \]

SR increases when search is intensified

These 2 ratio may be computed (nearly) for free with appropriate data structures!
### Measuring Intensification/Diversification

#### Resampling ratio (RR) \(\rightsquigarrow\) quantifies diversification

\[
RR = \frac{\#\{\text{computed solutions}\} - \#\{\text{different computed solutions}\}}{\#\{\text{computed solutions}\}}
\]

- Maximal diversification \(0 \leq RR \leq 1 \Rightarrow\) Stagnation

#### Similarity ratio (SR) \(\rightsquigarrow\) quantifies intensification

- SR = average similarity of the set \(S\) of computed solutions
  \(\rightsquigarrow\) average similarity of pairs of solutions of \(S\)
  \(\rightsquigarrow\) similarity of 2 solutions = percentage of shared components

- SR increases when search is intensified

These 2 ratio may be computed (nearly) for free with appropriate data structures!
Measuring intensification/Diversification: Example

### Resampling ratio

<table>
<thead>
<tr>
<th>Number of cycles:</th>
<th>500</th>
<th>1000</th>
<th>1500</th>
<th>2000</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1$, $\rho = 0.01$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$\alpha = 2$, $\rho = 0.01$</td>
<td>0.00</td>
<td>0.04</td>
<td>0.06</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>$\alpha = 2$, $\rho = 0.02$</td>
<td>0.06</td>
<td>0.10</td>
<td>0.12</td>
<td>0.13</td>
<td>0.13</td>
</tr>
</tbody>
</table>

### Similarity ratio

![Graph showing similarity ratio](image)
Overview of the talk

1. Introduction
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. Ant Colony Optimization
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. Solving CSPs with ACO
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. Constraint Programming with ACO
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. Conclusion
The car sequencing problem

Goal: Sequence cars along an assembly line

- Each car requires a set of options
- Space cars requiring a same option

Variables are associated with positions in the assembly line:

\[
\begin{array}{cccccccccc}
X1 & X2 & X3 & X4 & X5 & X6 & X7 & X8 & X9 & X10 \\
\end{array}
\]

Domain of a variable = set of cars to be sequenced:

Sequencing constraints:

\[
\begin{align*}
\text{\( \leq \frac{1}{2} \); } & \quad \text{\( \leq \frac{2}{5} \); } & \quad \text{\( \leq \frac{1}{5} \); } & \quad \text{\( \leq \frac{1}{3} \)} \\
\end{align*}
\]

Solution:
ACO for the car sequencing problem

- Greedy randomized algorithm
- ACO 1 $\Rightarrow$ pheromone structure to identify good sequences
- ACO 2 $\Rightarrow$ pheromone structure to identify critical cars
- ACO 1+2 $\Rightarrow$ combine the two pheromone structures
Greedy randomized algorithm

- Start from an empty sequence $\pi$
- While not all cars have been sequenced in $\pi$:
  - Let $cand$ be the set of cars not sequenced in $\pi$
  - Narrow $cand$ to the set of cars that
    - Introduce the fewest new constraint violations
    - Require different sets of options
  - Choose $c_i \in cand$ w.r.t. probability $p(c_i, cand, \pi)$
  - Add $c_i$ at the end of $\pi$
Greedy randomized algorithm

- Start from an empty sequence $\pi$
- While not all cars have been sequenced in $\pi$:
  - Let $cand$ be the set of cars not sequenced in $\pi$
  - Narrow $cand$ to the set of cars that
    - Introduce the fewest new constraint violations
    - Require different sets of options
  - Choose $c_i \in cand$ w.r.t. probability $p(c_i, cand, \pi)$
  - Add $c_i$ at the end of $\pi$

$$p(c_i, cand, \pi) = \frac{[\eta(c_i, \pi)]^{\beta}}{\sum_{c_k \in cand} [\eta(c_k, \pi)]^{\beta}}$$

where
- $\eta(c_i, \pi) =$ problem-dependent heuristic function
  - sum of utilisation rates of options required by $c_i$
- $\beta =$ parameter that controls heuristic weight
ACO algorithm for learning good sequences

Pheromone structure

Pheromone is laid on couples of cars:
\[ \tau_1(c_i, c_j) = \text{pheromone on } (c_i, c_j) \]
\[ \rightsquigarrow \text{experience of the colony / sequencing } c_j \text{ just after } c_i \]

At each cycle, each ant builds a sequence

Probability of adding car \( c_j \) at the end of a sequence \( \pi \)

\[
p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} \cdot [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in \text{cand}} [\tau_\pi(c_k)]^{\alpha_1} \cdot [\eta_\pi(c_k)]^{\beta}}
\]
ACO algorithm for learning good sequences

Pheromone structure

Pheromone is laid on couples of cars:
\[ \tau_1(c_i, c_j) = \text{pheromone on } (c_i, c_j) \]
\[ \xrightarrow{\sim} \text{experience of the colony / sequencing } c_j \text{ just after } c_i \]

At each cycle, each ant builds a sequence

Probability of adding car \( c_j \) at the end of a sequence \( \pi \)

\[
p(c_j) = \frac{[\tau_{\pi}(c_j)]^{\alpha_1} \cdot [\eta_{\pi}(c_j)]^{\beta}}{\sum_{c_k \in \text{cand}} [\tau_{\pi}(c_k)]^{\alpha_1} \cdot [\eta_{\pi}(c_k)]^{\beta}}
\]
**ACO algorithm for learning good sequences**

**Pheromone structure**

Pheromone is laid on couples of cars:

$$\tau_1(c_i, c_j) = \text{pheromone on } (c_i, c_j)$$

$$\xRightarrow{\text{experience of the colony / sequencing } c_j \text{ just after } c_i}$$

**At each cycle, each ant builds a sequence**

Probability of adding car $c_j$ at the end of a sequence $\pi$

$$p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} \cdot [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in \text{cand}} [\tau_\pi(c_k)]^{\alpha_1} \cdot [\eta_\pi(c_k)]^{\beta}}$$

$$\tau_\pi(c_j) = \text{pheromone factor}$$

$$\xRightarrow{\text{if the last car of } \pi \text{ is } c_i \text{ then } \tau_\pi(c_j) = \tau_1(c_i, c_j)}$$

$$\xRightarrow{\text{when choosing the first car of a sequence, } \tau_\pi(c_j) = 1}$$
ACO algorithm for learning good sequences

**Pheromone structure**

Pheromone is laid on couples of cars:
\[ \tau_1(c_i, c_j) = \text{pheromone on } (c_i, c_j) \]
\[ \sim \text{ experience of the colony / sequencing } c_j \text{ just after } c_i \]

**At each cycle, each ant builds a sequence**

Probability of adding car \( c_j \) at the end of a sequence \( \pi \)
\[
\rho(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} \cdot [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in \text{cand}} [\tau_\pi(c_k)]^{\alpha_1} \cdot [\eta_\pi(c_k)]^{\beta}}
\]

\( \eta_\pi(c_j) = \text{local heuristic that evaluates the hardness of } c_j \)
\[ \sim \text{ sum of utilization rates of options required by } c_j \]
ACO algorithm for learning good sequences

Pheromone structure

Pheromone is laid on couples of cars:
\[ \tau_1(c_i, c_j) = \text{pheromone on } (c_i, c_j) \]
\[ \rightsquigarrow \text{experience of the colony / sequencing } c_j \text{ just after } c_i \]

At each cycle, each ant builds a sequence

Probability of adding car \( c_j \) at the end of a sequence \( \pi \)

\[ P(c_j) = \frac{[\tau_{\pi}(c_j)]^{\alpha_1} \cdot [\eta_{\pi}(c_j)]^{\beta}}{\sum_{c_k \in \text{cand}} [\tau_{\pi}(c_k)]^{\alpha_1} \cdot [\eta_{\pi}(c_k)]^{\beta}} \]

At the end of each cycle, reward the best sequences

Add pheromone on consecutive cars in the cycle best sequences

Quantity of pheromone added = \( \frac{1}{\text{number of violated constraints}} \)
ACO algorithm for learning for critical cars

**Pheromone structure**

Pheromone trails associated with cars (grouped in car classes w.r.t. required options):

- $\tau_2(\text{cc})$ = quantity of pheromone associated with car class $\text{cc}$
- $\tau_2(\text{cc})$ = experience of the colony / difficulty of sequencing cars of $\text{cc}$

**At each cycle, each ant builds a sequence**

Probability of adding a car of class $\text{cc}_j$ at the end of a sequence

$$p(\text{cc}_j) = \frac{[\tau_2(\text{cc}_j)]^{\alpha_2}}{\sum_{\text{cc}_k \in \text{cand}}[\tau_2(\text{cc}_k)]^{\alpha_2}}$$

**Pheromone updating step**

- While constructing sequences:
  - add pheromone on classes violating constraints
- At the end of every sequence construction:
  - evaporation
ACO algorithm for learning for critical cars

**Pheromone structure**

Pheromone trails associated with cars (grouped in car classes w.r.t. required options):

\[ \tau_2(\text{cc}) = \text{quantity of pheromone associated with car class } \text{cc} \]

\[ \Rightarrow \text{experience of the colony / difficulty of sequencing cars of } \text{cc} \]

**At each cycle, each ant builds a sequence**

Probability of adding a car of class \( \text{cc}_j \) at the end of a sequence

\[
p(\text{cc}_j) = \frac{[\tau_2(\text{cc}_j)]^{\alpha_2}}{\sum_{\text{cc}_k \in \text{cand}} [\tau_2(\text{cc}_k)]^{\alpha_2}}
\]

**Pheromone updating step**

- While constructing sequences:
  - \( \Rightarrow \) add pheromone on classes violating constraints
- At the end of every sequence construction:
  - \( \Rightarrow \) evaporation
# ACO algorithm for learning for critical cars

## Pheromone structure

Pheromone trails associated with cars (grouped in car classes w.r.t. required options):
- $\tau_2(cc) = \text{quantity of pheromone associated with car class } cc$
- $\tau_2(cc) = \text{experience of the colony / difficulty of sequencing cars of } cc$

## At each cycle, each ant builds a sequence

Probability of adding a car of class $cc_j$ at the end of a sequence:

$$ p(cc_j) = \frac{[\tau_2(cc_j)]^{2\alpha}}{\sum_{cc_k \in \text{cand}}[\tau_2(cc_k)]^{2\alpha}} $$

## Pheromone updating step

- While constructing sequences:
  - add pheromone on classes violating constraints
- At the end of every sequence construction:
  - evaporation
Double ACO algorithm

Combine the two pheromone structures

- To learn for sequences: \( \forall \text{ cars } c_i \text{ and } c_j \)
  \[ \tau_1(c_i, c_j) = \text{experience of the colony / sequence } c_j \text{ after } c_i \]
- To learn for critical cars: \( \forall \text{ car class } cc \)
  \[ \tau_2(cc) = \text{experience of the colony / sequence cars of } cc \]

At each cycle, each ant builds a sequence

Probability of adding car \( c_j \) at the end of a sequence

\[ p(c_j) = \frac{[\tau_1(c_i, c_j)]^{\alpha_1} \cdot [\tau_2(\text{classOf}(c_j))]^{\alpha_2}}{\sum_{c_k \in \text{cand}} [\tau_1(c_i, c_k)]^{\alpha_1} \cdot [\tau_2(\text{classOf}(c_k))]^{\alpha_2}} \]

Pheromone updating steps

- While constructing sequences: add trails on \( \tau_2(cc) \)
- At the end of every sequence construction: evaporate \( \tau_2(cc) \)
- At the end of every cycle: evaporate + add trails on \( \tau_1(c_i, c_j) \)
Double ACO algorithm

Combine the two pheromone structures

- To learn for sequences: ∀ cars $c_i$ and $c_j$
  $\tau_1(c_i, c_j) = \text{experience of the colony / sequence } c_j \text{ after } c_i$
- To learn for critical cars: ∀ car class $cc$
  $\tau_2(cc) = \text{experience of the colony / sequence cars of } cc$

At each cycle, each ant builds a sequence

Probability of adding car $c_j$ at the end of a sequence

$$p(c_j) = \frac{[\tau_1(c_i,c_j)]^{\alpha_1} \cdot [\tau_2(\text{classOf}(c_j))]^{\alpha_2}}{\sum_{c_k \in \text{cand}} [\tau_1(c_i,c_k)]^{\alpha_1} \cdot [\tau_2(\text{classOf}(c_k))]^{\alpha_2}}$$

Pheromone updating steps

- $\Rightarrow$ While constructing sequences: add trails on $\tau_2(cc)$
- $\Rightarrow$ At the end of every sequence construction: evaporate $\tau_2(cc)$
- $\Rightarrow$ At the end of every cycle: evaporate + add trails on $\tau_1(c_i, c_j)$
Double ACO algorithm

Combine the two pheromone structures

- To learn for sequences: ∀ cars $c_i$ and $c_j$
  $\tau_1(c_i, c_j) = \text{experience of the colony / sequence } c_j \text{ after } c_i$
- To learn for critical cars: ∀ car class $cc$
  $\tau_2(cc) = \text{experience of the colony / sequence cars of } cc$

At each cycle, each ant builds a sequence

Probability of adding car $c_j$ at the end of a sequence

$$p(c_j) = \frac{[\tau_1(c_i, c_j)]^{\alpha_1} \cdot [\tau_2(\text{classOf}(c_j))]}{\sum_{c_k \in \text{cand}} [\tau_1(c_i, c_k)]^{\alpha_1} \cdot [\tau_2(\text{classOf}(c_k))]}^{\alpha_2}$$

Pheromone updating steps

- $\rightarrow$ While constructing sequences: add trails on $\tau_2(cc)$
- $\rightarrow$ At the end of every sequence construction: evaporate $\tau_2(cc)$
- $\rightarrow$ At the end of every cycle: evaporate + add trails on $\tau_1(c_i, c_j)$
Solving CSPs with ACO

Experimental results

<table>
<thead>
<tr>
<th>Algo</th>
<th>Heur. $\eta$</th>
<th>Pheromone 1</th>
<th>Pheromone 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\beta$</td>
<td>$\alpha_1$</td>
<td>$\mu_1$</td>
</tr>
<tr>
<td>Greedy($\eta$)</td>
<td>6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ACO($\tau_1$, $\eta$)</td>
<td>6</td>
<td>2</td>
<td>1%</td>
</tr>
<tr>
<td>ACO($\tau_2$)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ACO($\tau_1$, $\tau_2$)</td>
<td>-</td>
<td>2</td>
<td>1%</td>
</tr>
</tbody>
</table>

5000 cycles, 30 ants $\Rightarrow$ 150000 sequence constructions
50 runs per instance on a 2GHz Pentium 4

Benchmark

- Instances of [Lee et al 98] trivially solved in less than 0.01s
- Instances of [Perron & Shaw 04]
  - Satisfiable instances
  - 32, 21 and 29 instances with 100, 300 and 500 cars resp.
  - 20 classes and 8 options
Comparison: success / CPU time

![Graph showing comparison between success rate and CPU time for different methods.](image-url)
Comparison: success / CPU time
Comparison: success / CPU time

- Glouton
- ACO 1
- ACO 2
Comparison: success / CPU time
Comparison with Local Search

Considered approaches

- **SN** = Winner of the ROADEF’2005 challenge [Estellon et al. 05]
  - First solution built in a greedy way
  - Neighborhood = swap / mirror / insert / shuffle
  - First non decreasing neighbor

- **ID Walk** [Neveu et al. 04]
  - First solution built in a greedy way
  - Neighborhood = swap
  - at most \( Max \) neighbors are considered at each move
    - First non decreasing neighbor...
    - ... or best neighbor over \( Max \) neighbors
    - \( \Rightarrow \) Adaptive tuning of \( Max \)
Comparison: 32 instances with 100 cars
Comparison: 21 instances with 300 cars
Comparison: 29 instances with 500 cars
Overview of the talk

1. Introduction
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. Ant Colony Optimization
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. Solving CSPs with ACO
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. Constraint Programming with ACO
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. Conclusion
Solving CSPs with ACO in a generic way

- initialize pheromone trails to $\tau_{max}$
- repeat
  - each ant builds a complete assignment
  - update pheromone trails
- until solution found or max cycles reached
Solving CSPs with ACO in a generic way

- initialize **pheromone trails** to $\tau_{max}$
- repeat
  1. each ant builds a complete assignment
  2. update pheromone trails
- until solution found or max cycles reached

**Pheromone structure associated with a CSP** $(X, D, C)$

Pheromone is laid on variable/value couples:
For every variable $X_i$, and for every value $v_i \in D(X_i)$,

$$\tau_{\langle X_i, v_i \rangle} = \text{quantity of pheromone associated with } \langle X_i, v_i \rangle$$

$\leadsto$ learnt desirability of assigning $v_i$ to $X_i$
Solving CSPs with ACO in a generic way

- Initialize pheromone trails to $\tau_{max}$
- Repeat
  1. Each ant builds a complete assignment
  2. Update pheromone trails
- Until solution found or max cycles reached

Construction of a complete assignment $A$

Iteratively assign variables until all variables assigned:

- Choose a non assigned variable $X_i$
- Choose a value $v_i \in D(X_i)$ with probability

$$p(v_i) = \frac{[\tau \langle X_i, v_i \rangle]^{\alpha} \cdot [1/(1+\Delta \text{violations}(A, X_i, v_i))]^{\beta}}{\sum_{v_k \in D(X_i)} [\tau \langle X_i, v_k \rangle]^{\alpha} \cdot [1/(1+\Delta \text{violations}(A, X_i, v_j))]^{\beta}}$$
Solving CSPs with ACO in a generic way

- initialize pheromone trails to $\tau_{max}$
- repeat
  1. each ant builds a complete assignment
  2. update pheromone trails
- until solution found or max cycles reached

Pheromone updating step

- Evaporation: multiply pheromone trails by $(1 - \rho)$
  $\Rightarrow \rho =$ evaporation rate ($0 \leq \rho \leq 1$)
- Reward the best assignment $A$ of the cycle:
  $\forall \langle X_i, v_i \rangle \in A$, increment $\tau_{\langle X_i, v_i \rangle}$ by $1/\#\text{violations}(A)$
Experimental results: considered instances

CSP solver competition in 2006

- 1195 binary instances defined in extension...
- ...selection of satisfiable instances
- ...selection of instances that pass through my parser

→ 230 instances from 6 benchmarks
Experimental results

- 23 solvers of the competition, all based on complete approaches
  - CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO
  - CPU time limit of 1800s on a 2.16GHZ Intel Core Duo
- Tabu (tabu list length=50, restart every 1,000,000 moves)
  - CPU time limit of 1800s on a 2.16GHZ Intel Core Duo

<table>
<thead>
<tr>
<th>Solver</th>
<th>#solved</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>buggy 2.5.s</td>
<td>207</td>
<td>6 best complete solvers of the competition in 2006</td>
</tr>
<tr>
<td>buggy 2.5</td>
<td>207</td>
<td></td>
</tr>
<tr>
<td>Abscon 109 AC</td>
<td>207</td>
<td></td>
</tr>
<tr>
<td>VALCSP 3.1</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>VALCSP 3.0</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>Abscon 109 ESAC</td>
<td>204</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Experimental results

- 23 solvers of the competition, all based on complete approaches
  - CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO
  - CPU time limit of 1800s on a 2.16GHZ Intel Core Duo
- Tabu (tabu list length=50, restart every 1,000,000 moves)
  - CPU time limit of 1800s on a 2.16GHZ Intel Core Duo

<table>
<thead>
<tr>
<th>Solver</th>
<th>#solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>buggy 2.5.s</td>
<td>207</td>
</tr>
<tr>
<td>buggy 2.5</td>
<td>207</td>
</tr>
<tr>
<td>Abscon 109 AC</td>
<td>207</td>
</tr>
<tr>
<td>VALCSP 3.1</td>
<td>206</td>
</tr>
<tr>
<td>VALCSP 3.0</td>
<td>206</td>
</tr>
<tr>
<td>Abscon 109 ESAC</td>
<td>204</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ACO</td>
<td>218</td>
</tr>
</tbody>
</table>

6 best complete solvers of the competition in 2006.
Experimental results

- 23 solvers of the competition, all based on complete approaches
  - CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO
  - CPU time limit of 1800s on a 2.16GHZ Intel Core Duo
- Tabu (tabu list length=50, restart every 1,000,000 moves)
  - CPU time limit of 1800s on a 2.16GHZ Intel Core Duo

<table>
<thead>
<tr>
<th>Solver</th>
<th>#solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>buggy 2.5.s</td>
<td>207</td>
</tr>
<tr>
<td>buggy 2.5</td>
<td>207</td>
</tr>
<tr>
<td>Abscon 109 AC</td>
<td>207</td>
</tr>
<tr>
<td>VALCSP 3.1</td>
<td>206</td>
</tr>
<tr>
<td>VALCSP 3.0</td>
<td>206</td>
</tr>
<tr>
<td>Abscon 109 ESAC</td>
<td>204</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ACO</td>
<td>218</td>
</tr>
<tr>
<td>Tabu</td>
<td>226</td>
</tr>
</tbody>
</table>

6 best complete solvers of the competition in 2006
From CSPs solving to Constraint Programming

Ants can solve CSPs
- State-of-the-art results with a dedicated ACO algorithm for the car sequencing problem
- Competitive results with a generic ACO algorithm for CSPs

ACO-based CP
- Use a CP language to model the problem by means of constraints
- Use ACO to solve the problem
  - Investigation of two directions:
    - Integration of ACO in Ilog solver → CSPs
    - Integration of ACO in Ilog CP Optimizer → COPs
  - joint work with IBM / Ilog
From CSPs solving to Constraint Programming

Ants can solve CSPs

- State-of-the-art results with a dedicated ACO algorithm for the car sequencing problem
- Competitive results with a generic ACO algorithm for CSPs

ACO-based CP

- Use a CP language to model the problem by means of constraints
- Use ACO to solve the problem
  ⟷ Investigation of two directions:
    - Integration of ACO in Ilog solver ⟷ CSPs
    - Integration of ACO in Ilog CP Optimizer ⟷ COPs
  ⟷ joint work with IBM / Ilog
Overview of the talk

1. **Introduction**
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. **Ant Colony Optimization**
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. **Solving CSPs with ACO**
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. **Constraint Programming with ACO**
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. **Conclusion**
Ant-CP procedure

- initialize pheromone trails to \( \tau_{max} \)
- repeat
  1. each ant builds a partial consistent assignment
  2. update pheromone trails
- until solution found or max cycles reached
Ant-CP procedure

- initialize **pheromone trails** to $\tau_{max}$
- repeat
  1. each ant builds a partial consistent assignment
  2. update pheromone trails
- until solution found or max cycles reached

Pheromone structure associated with a CSP $(X, D, C)$

Pheromone is laid on variable/value couples:
For every variable $X_i$, and for every value $v_i \in D(X_i)$,

$$\tau_{\langle X_i,v_i \rangle} = \text{quantity of pheromone associated with } \langle X_i, v_i \rangle$$

$\rightsquigarrow$ learnt desirability of assigning $v_i$ to $X_i$
Ant-CP procedure

- Initialize pheromone trails to $\tau_{max}$
- Repeat
  1. Each ant builds a partial consistent assignment
  2. Update pheromone trails
- Until solution found or max cycles reached

Construction of a partial consistent assignment $A$

Iteratively assign variables until all variables assigned or failure:
- Choose a non-assigned variable $X_i$
- Choose a value $v_i \in D(X_i)$ with probability

$$p(v_i) = \frac{[\tau_{X_i,v_i}]^\alpha \cdot [\eta(X_i,v_i)]^\beta}{\sum_{v_k \in D(X_i)} [\tau_{X_i,v_k}]^\alpha \cdot [\eta(X_i,v_k)]^\beta}$$

where $\eta(X_i,v_i)$ is a problem-dependent heuristic factor
- Propagate to remove inconsistent values from domains
Ant-CP procedure

- initialize pheromone trails to $\tau_{max}$
- repeat
  1. each ant builds a partial consistent assignment
  2. update pheromone trails
- until solution found or max cycles reached

Pheromone updating step

- Evaporation: multiply pheromone trails by $(1 - \rho)$
  $\sim \rho = \text{evaporation rate} \ (0 \leq \rho \leq 1)$
- Reward the best assignment $A$ of the cycle:
  $\forall \langle X_i, v_i \rangle \in A$, increment $\tau_{\langle X_i, v_i \rangle}$ by $1/(1 + |A_{best}| - |A|)$
  where $A_{best}$ is the best assignment found so far
Pheromone structures

Default pheromone structure
pheromone is laid on variable/value couples

Specific pheromone structures
The user has to define
- a set of pheromone trails
- a function $\tau \sim$ pheromone factors
- a function $comp \sim$ rewarded components
Application to the car sequencing problem: CP model

First model of the User’s manual of ILOG Solver

Variables

- For each position $i$ in the sequence, $car_i = \text{class of the } i\text{th car}$
- For each position $i$ in the sequence and each option $j$, $opt_{ij} = 1$ if $car_i$ requires option $j$ and $opt_{ij} = 0$ otherwise

Constraints

- Constraints on the number of cars to be produced:
  $\forall$ car class $c$, $\#\{car_i = c\} = \text{nb of cars of class } c \text{ to be produced}$
  $\leadsto$ IloDistribute

- Constraint between $car$ and $opt$ variables:
  $\forall$ car $i$ and $\forall$ option $j$, $opt_{ij} = 1$ iff $car_i$ requires option $j$
  $\leadsto$ IloBoolAbstraction

- Capacity constraints:
  $\forall$ option $j$, $\forall$ subseq. $s$ of $q_j$ cars, $\sum_{i\in s} opt_{ij} \leq p_j$
**Application to the car sequencing problem: Pheromone structures**

<table>
<thead>
<tr>
<th>Pheromone Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Pheromone is laid on couples (position/car class)</td>
</tr>
<tr>
<td>Classes [Gravel et al 04]</td>
<td>Pheromone is laid on couples of consecutive car classes</td>
</tr>
<tr>
<td>Cars [Solnon 00]</td>
<td>Pheromone is laid on couples of consecutive cars</td>
</tr>
<tr>
<td>Empty</td>
<td>Pheromone is not used</td>
</tr>
</tbody>
</table>
Application to the car sequencing problem: Heuristic factor

Recall: choice of a value for a variable

\[ p(v_i) = \frac{[\tau(X_i, v_i)]^\alpha \cdot [\eta(X_i, v_i)]^\beta}{\sum_{v_k \in D(X_i)} [\tau(X_i, v_k)]^\alpha \cdot [\eta(X_i, v_k)]^\beta} \]

where \( \eta(X_i, v_i) \) is a problem-dependent heuristic factor

**Utilisation rate \( UR(o_i) \) of an option \( o_i \) [Smith 97]**

- \( UR(o_i) = \) nb of required slots / nb of available slots
- \( UR(o_i) > 1 \rightarrow \) no solution

**Comparison of 2 heuristics**

- \( DSU = \) sum of utilization rates of required options
  \( \rightarrow \) favor cars that require options with high utilisation rates
- \( DSU + P = \) sum of utilization rates of required options
  + failure when \( UR(o_i) > 1 \)
  + filter domains when \( UR(o_i) = 1 \)
Comparison of the 4 pheromone structures with the DSU heuristic

- **Cars**
- **Default**
- **Classes**
- **no pheromone**

Success rate (for 10 runs per instance)

<table>
<thead>
<tr>
<th>Number of cycles</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>500</td>
<td>55</td>
</tr>
<tr>
<td>1000</td>
<td>60</td>
</tr>
<tr>
<td>1500</td>
<td>65</td>
</tr>
<tr>
<td>2000</td>
<td>70</td>
</tr>
<tr>
<td>2500</td>
<td>75</td>
</tr>
<tr>
<td>3000</td>
<td>80</td>
</tr>
</tbody>
</table>
Comparison of the 2 heuristics
Overview of the talk

1. **Introduction**
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. **Ant Colony Optimization**
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. **Solving CSPs with ACO**
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. **Constraint Programming with ACO**
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. **Conclusion**
Motivations

Combinatorial Optimization Problems (COPs)

Optimization problems defined by \((X, D, C, F)\)
- \(X = \{X_1, X_2, \ldots, X_n\}\) is a set of variables
- \(D(X_i)\) is the domain of \(X_i\)
- \(C\) is a set of constraints
- \(F : X \rightarrow \mathbb{R}\) is an objective function

Goal = Satisfy all constraints and maximize \(F\)

CP Optimizer (CPO) / IBM Ilog

The user states the problem... and the computer solves it!
- High-level modeling language
- Search \(\sim\) Branch & Propagate & Bound (B&P&B)
  - Branch on choice points (variable/value assignments)
  - Propagate constraints to prune branches
  - Compute bounds on \(F\) to prune branches
Introduction

Ant Colony Optimization

Solving CSPs with ACO

Constraint Programming with ACO

Conclusion

Integration of ACO in CP Optimizer (1/3)

Description of CPO-ACO

- Describe the problem to solve with the CPO modeling language
- Solve the problem in two phases
  - Phase 1: sample the search space of feasible solutions
    - CPO builds feasible solutions
    - ACO learns for good sub-areas
  - Phase 2: search for an optimal feasible solution
    - CPO performs a B&P&B search
    - ACO guides CPO as a value ordering heuristic
Introduction

Ant Colony Optimization

Solving CSPs with ACO

Constraint Programming with ACO

Conclusion

Integration of ACO in CP Optimizer (2/3)

Phase 1 of CPO-ACO

- Pheromone structure = \{\tau(x_i, v_i), x_i \in X, v_i \in D(x_i)\}
- Initialize all pheromone trails to \(\tau_{\text{max}}\)
- At each cycle:
  - Each ant asks CPO to build a feasible solution
    - The objective function is ignored
    - The value ordering heuristic of CPO is replaced by
      \[ p(v_i) = \frac{[\tau(x_i, v_i)]^\alpha \cdot [1/\text{impact}(v_i)]^\beta}{\sum_{v_j \in D(x_i)}[\tau(x_i, v_j)]^\alpha \cdot [1/\text{impact}(v_j)]^\beta} \]
  - Evaporate all pheromone trails
  - Add pheromone on variable/value couples of the best feasible solution of the cycle and the best feasible solution since the beginning of the search
- Until stagnation or time limit reached
Phase 2 of CPO-ACO

- Bound the objective function with the best feasible solution found during phase 1
- Ask CPO to search for the best feasible solution

  - CPO performs a B&P&B search
  - The value ordering heuristic of CPO is replaced by

    \[ \text{choose } v_i \text{ which maximizes } \tau(x_i, v_i)^\alpha \cdot \frac{1}{\text{impact}(v_i)}^{\beta} \]
# Experimental results

## Results for the MKP

<table>
<thead>
<tr>
<th>Name</th>
<th># I</th>
<th># X</th>
<th>avg (sd)</th>
<th>&gt;_{t-test}</th>
<th>avg (sd)</th>
<th>&gt;_{t-test}</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.100-*</td>
<td>20</td>
<td>100</td>
<td>1.20 (0.30)</td>
<td>0%</td>
<td>0.46 (0.23)</td>
<td>100%</td>
</tr>
<tr>
<td>10.100-*</td>
<td>20</td>
<td>100</td>
<td>1.53 (0.31)</td>
<td>0%</td>
<td>0.83 (0.34)</td>
<td>100%</td>
</tr>
<tr>
<td>30.100-*</td>
<td>20</td>
<td>100</td>
<td>1.24 (0.06)</td>
<td>0%</td>
<td>0.86 (0.08)</td>
<td>85%</td>
</tr>
</tbody>
</table>

## Results for the QAP

<table>
<thead>
<tr>
<th>Name</th>
<th># I</th>
<th># X</th>
<th>avg (sd)</th>
<th>&gt;_{t-test}</th>
<th>avg (sd)</th>
<th>&gt;_{t-test}</th>
</tr>
</thead>
<tbody>
<tr>
<td>bur*</td>
<td>7</td>
<td>26</td>
<td>1.17 (0.43)</td>
<td>0%</td>
<td>0.88 (0.43)</td>
<td>57%</td>
</tr>
<tr>
<td>chr*</td>
<td>11</td>
<td>19</td>
<td>12.11 (6.81)</td>
<td>9%</td>
<td>10.99 (6.01)</td>
<td>45%</td>
</tr>
<tr>
<td>had*</td>
<td>5</td>
<td>16</td>
<td>1.07 (0.89)</td>
<td>0%</td>
<td>0.54 (1.14)</td>
<td>60%</td>
</tr>
<tr>
<td>kra*</td>
<td>2</td>
<td>30</td>
<td>17.46 (3.00)</td>
<td>0%</td>
<td>14.99 (2.79)</td>
<td>100%</td>
</tr>
<tr>
<td>lipa*</td>
<td>6</td>
<td>37</td>
<td>22.11 (0.82)</td>
<td>0%</td>
<td>20.87 (0.75)</td>
<td>100%</td>
</tr>
<tr>
<td>nug*</td>
<td>15</td>
<td>20</td>
<td>8.03 (1.59)</td>
<td>0%</td>
<td>5.95 (1.44)</td>
<td>80%</td>
</tr>
<tr>
<td>rou*</td>
<td>3</td>
<td>16</td>
<td>5.33 (1.15)</td>
<td>0%</td>
<td>3.98 (1.00)</td>
<td>67%</td>
</tr>
<tr>
<td>scr*</td>
<td>3</td>
<td>16</td>
<td>4.60 (2.4)</td>
<td>0%</td>
<td>5.12 (2.60)</td>
<td>0%</td>
</tr>
<tr>
<td>tai*</td>
<td>4</td>
<td>16</td>
<td>6.06 (1.35)</td>
<td>25%</td>
<td>4.84 (1.25)</td>
<td>50%</td>
</tr>
</tbody>
</table>
Overview of the talk

1. Introduction
   - How to solve Combinatorial Optimization Problems?
   - Greedy randomized approaches

2. Ant Colony Optimization
   - Basic principles of ACO
   - Intensifying/diversifying search with ACO

3. Solving CSPs with ACO
   - Solving the car sequencing problem with ACO
   - Solving CSPs with ACO in a generic way

4. Constraint Programming with ACO
   - Integration of ACO in Ilog solver for solving CSPs
   - Integration of ACO in CP Optimizer for solving COPs

5. Conclusion


**Integrating ACO in a CP language**

- Integration in Ilog solver to solve CSPs:
  - Use CP to model, propagate and check constraints
  - Incomplete search guided by pheromone

- Integration in CP Optimizer to solve COPs:
  - Use CP to model, propagate and check constraints
  - Phase 1: incomplete search guided by pheromone
  - Phase 2: complete B&P&B search guided by pheromone

**Next step**

- Get rid of parameter tuning \( \rightsquigarrow \) reactive ACO
- Design propagation procedures for an ACO search
- Integrate problem-dependent heuristics
Some references

- C. Solnon: Ant Colony Optimization and Constraint Programming. Wiley 2010
- C. Solnon: Combining two ACO algorithms for solving the car sequencing problem. EJOR 2008
- M. Khichane, P. Albert, C. Solnon: Strong Combination of Ant Colony Optimization with Constraint Programming Optimization. CPAIOR 2010