Using Ant Colony Optimization to guide a CP search

Madjid Khichane\textsuperscript{1}, Patrick Albert\textsuperscript{1}, and Christine Solnon\textsuperscript{2}

\textsuperscript{1} ILOG
\textsuperscript{2} LIRIS, UMR 5205 CNRS / University of Lyon

First Workshop on Autonomous Search
Motivations

**Constraint Programming**

- High level language for modelling problems declaratively
- Propagate & Backtrack search engine
  - may spend unacceptable time to solve some instances
  - add heuristics to guide the search

**Our goal: use ACO to guide a CP search**

- Describe the problem with Ilog solver
- Use Ilog solver to propagate and verify constraints
- Use Ant Colony Optimization to guide the search
Motivations

Constraint Programming
- High level language for modelling problems declaratively
- Propagate & Backtrack search engine
  - may spend unacceptable time to solve some instances
  - add heuristics to guide the search

Our goal: use ACO to guide a CP search
- Describe the problem with Ilog solver
- Use Ilog solver to propagate and verify constraints
- Use Ant Colony Optimization to guide the search
Brief history of ACO

Ant System

[Dorigo 92]: application to the Travelling Salesman Problem

Extensions of Ant System

Ant Colony System [Dorigo & Gambardella 97], $\text{MAX} - \text{MIN}$ 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
The $\text{MAX} - \text{MIN}$ Ant System

- initialize pheromone trails to $\tau_{max}$
- repeat
  1. each ant builds a solution
  2. update pheromone trails
- until optimal solution found or stagnation
The **MAX** − **MIN** Ant System

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

**Greedy randomized construction of a solution**

- Let $S$ = partial solution
  and $cand$ = candidate solution components
- Choose $c_j \in cand$ with probability

$$p(c_j) = \frac{[\tau_S(c_j)]^\alpha \cdot [\eta_S(c_j)]^\beta}{\sum_{c_k \in cand} [\tau_S(s_k)]^\alpha \cdot [\eta_S(s_k)]^\beta}$$
The \( \textbf{MAX} - \textbf{MIN} \) Ant System

- initialize pheromone trails to \( \tau_{\text{max}} \)
- repeat
  1. each ant builds a solution
  2. update pheromone trails
- until optimal solution found or stagnation

Greedy randomized construction of a solution

- Let \( S = \) partial solution
  and \( \text{cand} = \) candidate solution components
- Choose \( c_j \in \text{cand} \) with probability

\[
p(c_j) = \frac{[\tau_S(c_j)]^\alpha \cdot [\eta_S(c_j)]^\beta}{\sum_{c_k \in \text{cand}} [\tau_S(s_k)]^\alpha \cdot [\eta_S(s_k)]^\beta}
\]

\( \tau_S(c_j) \sim \) pheromone factor (past experience of the colony)
The \textbf{MAX - MIN} Ant System

- initialize pheromone trails to $\tau_{\text{max}}$
- repeat
  1. each ant builds a solution
  2. update pheromone trails
- until optimal solution found or stagnation

Greedy randomized construction of a solution

- Let $S = \text{partial solution}$
- and $\text{cand} = \text{candidate solution components}$
- Choose $c_j \in \text{cand}$ with probability

$$p(c_j) = \frac{[\tau_S(c_j)]^\alpha \cdot [\eta_S(c_j)]^\beta}{\sum_{c_k \in \text{cand}} [\tau_S(s_k)]^\alpha \cdot [\eta_S(s_k)]^\beta}$$

$\eta_S(c_j) \sim \text{heuristic factor (problem-dependent)}$
**The MAX – MIN Ant System**

- Initialize pheromone trails to $\tau_{\text{max}}$
- Repeat
  1. Each ant builds a solution
  2. Update pheromone trails
- Until optimal solution found or stagnation

**Greedy randomized construction of a solution**

Let $S =$ partial solution and $\text{cand} =$ candidate solution components.

Choose $c_j \in \text{cand}$ with probability

$$p(c_j) = \frac{[\tau_S(c_j)]^\alpha \cdot [\eta_S(c_j)]^\beta}{\sum_{c_k \in \text{cand}} [\tau_S(s_k)]^\alpha \cdot [\eta_S(s_k)]^\beta}$$

$\alpha, \beta \sim$ factor weights (parameters)
The \textbf{MAX – MIN} Ant System

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

\textbf{Pheromone updating step}

- Evaporation: multiply pheromone trails by $(1 - \rho)$
  \(\sim\ \rho = \text{evaporation rate} \ (0 \leq \rho \leq 1)\)
- Reward: add pheromone on components of the best solutions
- Bound pheromone trails between $\tau_{min}$ and $\tau_{max}$
  \(\sim\ \text{prevent from premature stagnation}\)
# Using ACO to solve CSPs

## Existing work

Build complete assignments / minimize constraint violations

- Repeat
  - Assign a variable to a value chosen w.r.t. ACO
  - Until all variables have been assigned

→ very competitive results... but *ad hoc* algorithms

## New proposition: CP with ants

Build partial consistent assignments / maximize nb of assigned var.

- Repeat
  - Assign a variable to a value chosen w.r.t. ACO
  - Propagate to remove inconsistent values from domains
  - Until propagation detects a failure or all variables assigned

→ straightforward integration within a CP language
Using ACO to solve CSPs

**Existing work**

Build complete assignments / minimize constraint violations

- Repeat
  - Assign a variable to a value chosen w.r.t. ACO
  - Until all variables have been assigned

⇝ very competitive results... but *ad hoc* algorithms

**New proposition: CP with ants**

Build partial consistent assignments / maximize nb of assigned var.

- Repeat
  - Assign a variable to a value chosen w.r.t. ACO
  - Propagate to remove inconsistent values from domains
  - Until propagation detects a failure or all variables assigned

⇝ straightforward integration within a CP language
Table of contents

1. Introduction
2. Overview of Ant-CP
3. Solving the car sequencing problem with Ant-CP
4. Experimental results
5. Intensification/Diversification of Ant-CP search
6. 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:

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

$\sim\sim$ learnt desirability of assigning $X_i$ to $v_i$
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 $X_i$ w.r.t. min domain heuristic
- Choose a value $v_i \in D(X_i)$ with probability

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

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 $(1 - \rho)$
  $\rho = \text{evaporation rate (0 } \leq \rho \leq 1)$
- 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
# Table of contents

1. Introduction
2. Overview of Ant-CP
3. **Solving the car sequencing problem with Ant-CP**
4. Experimental results
5. Intensification/Diversification of Ant-CP search
6. 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

Example

Set of cars to be sequenced:
The car sequencing problem

**Goal:** Sequence cars along an assembly line

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

**Example**

Set of cars to be sequenced:

Sequence constraints:

\[ \leq \frac{1}{2} ; \quad \leq \frac{2}{5} ; \quad \leq \frac{1}{5} ; \quad \leq \frac{1}{3} \]
The car sequencing problem

Goal: Sequence cars along an assembly line

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

Example

Set of cars to be sequenced:

Sequencing constraints:

\[ \leq \frac{1}{2} ; \quad \leq \frac{2}{5} ; \quad \leq \frac{1}{5} ; \quad \leq \frac{1}{3} \]

Solution:
CP model for the car sequencing problem

Variables
- For each position $i$ in the sequence, $car_i$ = class of the $i$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 $c$, $\#\{car_i = c\} = nb$ of cars of class $c$ to be produced
  $\rightsquigarrow$ IloDistribute
- Constraint between $car$ and $opt$ variables:
  $\forall$ car $i$ and $\forall$ option $j$, $opt_{ij} = 1$ iff $car_i$ requires option $j$
  $\rightsquigarrow$ IloBoolAbstraction
- Capacity constraints:
  $\forall$ option $j$ and $\forall$ subsequence $s$ of $q_j$ consecutive positions,
  $\sum_{i \in s} opt_{ij} \leq p_j$
## Existing value ordering heuristic

- Heuristic proposed in [Smith 97] and used in [Régin & Puget 97]
  - choose cars that require options with high utilisation rates
- Greedy randomized algorithm [Gottlieb, Puchta & Solnon 03]
  - Probability defined w.r.t. sum of dynamic utilisation rates
  - Solve all instances of [Lee et al. 95] (available in CSPlib)
    - ...in less than 0.01 second!

## Definition of the problem-dependent heuristic for Ant-CP

- First experiments without any heuristic: \( \eta(X_i, v_i) = 1 \)
  - evaluate the influence of ACO on the search process
- Then, experiments with the heuristic of [Gottlieb, Puchta & Solnon 03]
  \[ \eta(X_i, v_i) = \sum_{o_k \text{ required by } v_i} \text{dynamic utilization rate of } o_k \]
Test suite

Instances available in CSP lib

2 sets of instances

- 4 (hard) instances of [Smith 97, Régin & Puget 97]
  pb4-72, pb6-76, pb16-81, and pb10-93
  Satisfiable instances with 100 cars and 5 options.
  Utilization rates = 90%

- 70 (easy) instances of [Lee et al. 95]
  grouped into 7 sets of 10 instances w.r.t. utilization rates
  Satisfiable instances with 200 cars and 5 options.
  Utilization rates ∈ {60%, 65%, 70%, 75%, 80%, 85%, 90%}
Parameter setting

- Maximum number of cycles = 2500
- Number of ants = 30
- Heuristic factor weight $\beta$
  - When no heuristic is used, $\beta = 0$
  - When the heuristic based on utilisation rates is used, $\beta = 6$
- Pheromone bounds: $\tau_{min} = 0.01$ and $\tau_{max} = 10$
- Pheromone factor weight $\alpha \in \{1, 2, 3\}$
- Pheromone evaporation rate $\rho \in \{1\%, 2\%, 3\%\}$

Pheromone parameters influence the solution process
Influence of evaporation on the solution process (without heuristic)

- no pheromone (alpha=0)
- alpha=2 ; rho=1%
- alpha=2 ; rho=2%
- alpha=2 ; rho=3%

Percentage of assigned variables in the best solution vs. number of cycles.
Influence of $\alpha$ on the solution process (without heuristic)
Experimental results with the heuristic

- Heuristic only (alpha=0, beta=6)
- Heuristic + Pheromone (alpha=1, beta=6)
- Heuristic + Pheromone (alpha=2, beta=6)
Experimental results with the heuristic

<table>
<thead>
<tr>
<th>Instance</th>
<th>Heuristic only ((\alpha = 0, \beta = 6))</th>
<th>Heuristic + Pheromone ((\alpha = 1, \beta = 6))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SR</td>
<td>#Cycles</td>
</tr>
<tr>
<td>26-82</td>
<td>15%</td>
<td>1576</td>
</tr>
<tr>
<td>16-81</td>
<td>45%</td>
<td>1030</td>
</tr>
<tr>
<td>4-72</td>
<td>95%</td>
<td>706</td>
</tr>
<tr>
<td>41-66</td>
<td>100%</td>
<td>11</td>
</tr>
<tr>
<td>60-*</td>
<td>100%</td>
<td>1</td>
</tr>
<tr>
<td>65-*</td>
<td>100%</td>
<td>1</td>
</tr>
<tr>
<td>70-*</td>
<td>100%</td>
<td>2</td>
</tr>
<tr>
<td>75-*</td>
<td>100%</td>
<td>2</td>
</tr>
<tr>
<td>80-*</td>
<td>100%</td>
<td>2</td>
</tr>
<tr>
<td>85-*</td>
<td>100%</td>
<td>3</td>
</tr>
<tr>
<td>90-*</td>
<td>100%</td>
<td>3</td>
</tr>
</tbody>
</table>
Table of contents

1. Introduction
2. Overview of Ant-CP
3. Solving the car sequencing problem with Ant-CP
4. Experimental results
5. Intensification/Diversification of Ant-CP search
6. 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 \([τ_{min}, τ_{max}]\)
  - Initialize pheromone trails to \(τ_{max}\)
- Risk: convergence to optimality may be too long

⇒ Theoretical proof of convergence to optimality
Measuring Intensification/Diversification

**Resampling ratio (RR) \( \sim \) quantifies diversification**

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

- Maximal diversification \( \iff 0 \leq RR \leq 1 \Rightarrow \text{Stagnation} \)

**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!
Influence of $\alpha$ and $\rho$ on the resampling ratio

<table>
<thead>
<tr>
<th>Number of cycles</th>
<th>250</th>
<th>500</th>
<th>1000</th>
<th>1500</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1, \rho = 1%$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>$\alpha = 2, \rho = 1%$</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
<td>0.006</td>
<td>0.009</td>
</tr>
<tr>
<td>$\alpha = 2, \rho = 2%$</td>
<td>0.028</td>
<td>0.453</td>
<td>0.699</td>
<td>0.780</td>
<td>0.821</td>
</tr>
</tbody>
</table>
Influence of $\alpha$ and $\rho$ on the similarity ratio

- no pheromone (alpha=0)
- alpha=1 ; rho=1%
- alpha=2 ; rho=1%
- alpha=2 ; rho=2%
- alpha=2 ; rho=3%

The graph shows the similarity rate over the number of cycles for different values of $\alpha$ and $\rho$. As $\rho$ increases, the similarity ratio also increases, indicating a stronger influence of the pheromone. The highest similarity rates are observed when $\alpha=2$ and $\rho=3%$. The graph is a visual representation of the results obtained from the study.
Using ACO to guide a CP search

Complementarity of CP and ACO

- CP is used for
  - modelling the problem
  - propagating and checking constraints
- ACO is used to guide the search as a generic value ordering heuristic

First results on the car sequencing problem

- Pheromone actually improves the solution process
- Resampling and similarity ratio provide an insight into the search process
Further work

- More experimentations
  - on other instances of the car sequencing problem
  - on other problems
- Adaptive parameter tuning \(\leadsto\) towards reactive ACO
  - use resampling and similarity ratio to dynamically tune parameters
- Learning from failures