Constraint Programming with Ant Colony Optimization

Madjid Khichane\textsuperscript{12}, Patrick Albert\textsuperscript{1}, and Christine Solnon\textsuperscript{2}

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

CP-AI-OR’08
Motivations

**Constraint Programming (CP)**
- High level languages for modelling problems declaratively
- Branch & Propagate search engine
  - may spend unacceptable time to solve some instances

**Ant Colony Optimization (ACO)**
- Efficient algorithms for solving specific problems
  - ...but solving a new problem involves a lot of programming

**Our goal: use ACO to guide a CP search**
- Describe the problem with ILOG Solver
- Use ILOG Solver to propagate and check constraints
- Use ACO to guide the search
Basic principle of ACO

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

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

**Pheromone trails**

A pheromone trail $\tau_c$ is associated with every solution component $c$ ⊳ learnt desirability of using $c$ when building a solution
Basic principle of ACO

- 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

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

- 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

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

$\tau_S(c_j)$ \sim pheromone factor (past experience of the colony)
Basic principle of ACO

- 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

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

$\eta_S(c_j) \rightsquigarrow$ heuristic factor (problem-dependent)
Basic principle of ACO

- 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

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

$\alpha, \beta \sim$ factor weights (parameters)
Basic principle of ACO

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

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 $\rightsquigarrow$ minimize constraint violations
- Repeat
  - Assign a variable to a value chosen w.r.t. ACO
  - Until all variables have been assigned

$\rightsquigarrow$ very competitive results... but *ad hoc* algorithms

**New proposition: CP with ants**

Build partial consistent assignments $\rightsquigarrow$ 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

$\rightsquigarrow$ straightforward integration within a CP language
Using ACO to solve CSPs

**Existing work**

Build complete assignments $\rightsquigarrow$ minimize constraint violations
- Repeat
  - Assign a variable to a value chosen w.r.t. ACO
  - Until all variables have been assigned

$\rightsquigarrow$ very competitive results... but *ad hoc* algorithms

**New proposition: CP with ants**

Build partial consistent assignments $\rightsquigarrow$ 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

$\rightsquigarrow$ straightforward integration within a CP language
Table of contents

1 Introduction
2 Description of Ant-CP
3 Application to the Car sequencing
4 Experimental Results
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

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

Pheromone is laid on variable/value couples:

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

$\sim$ 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 $(1 - \rho)$
  $\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 $1/(1 + |A_{best}| - |A|)$
  where $A_{best}$ is the best assignment found so far
Table of contents

1 Introduction
2 Description of Ant-CP
3 Application to the Car sequencing
4 Experimental Results
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

Example

Set of cars to be sequenced:

![Cars to be sequenced](image)

Sequencing constraints:

![Constraints](image)

Solution:

![Solution](image)
# CP model for the car sequencing problem

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, $\sum_{i \in s} opt_{ij} \leq p_j$
# Pheromone structures

## Default vs specific pheromone structures

<table>
<thead>
<tr>
<th>Default</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>pheromone is laid on variable/value couples</td>
<td>the user has to define</td>
</tr>
</tbody>
</table>

- a set of pheromone trails
- a function $\tau \rightsQUAD \text{pheromone factors}$
- a function $\text{comp} \rightsQUAD \text{rewarded components}$

## Comparison of 4 pheromone structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></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>
**Utilisation rate** \( UR(o_i) \) **of an option** \( o_i \) [Smith 97]

- \( UR(o_i) = \frac{\text{number of required slots}}{\text{number of available slots}} \)
- \( UR(o_i) > 1 \Rightarrow \text{no solution} \)

**Comparison of 2 heuristics**

- \( DSU = \text{sum of utilization rates of required options} \)
  - \( \sim \text{favor cars that require options with high utilisation rates} \)
- \( DSU + P = \text{sum of utilization rates of required options} \)
  - \( + \text{failure when } UR(o_i) > 1 \)
  - \( + \text{filter domains when } UR(o_i) = 1 \)
Table of contents

1. Introduction
2. Description of Ant-CP
3. Application to the Car sequencing
4. Experimental Results
5. Conclusion
Test suites

Instances of [Lee et al. 95] available in CSP lib

- 70 satisfiable instances with 200 cars
- All solved by Ant-CP in a few cycles and less than one second (whatever the pheromone strategy and the heuristic factor)

⇝ these instances are too easy to evaluate Ant-CP

Instances of [Perron & Shaw 04]

- 82 instances
- from 100 to 500 cars; 8 options; 20 car classes
- all satisfiable

⇝ nearly half of these instances are difficult
Comparison of the 4 pheromone structures with the DSU heuristic
Comparison of the 2 heuristics

- Success rate (for 10 runs per instance)
- Number of cycles
- DSU
- DSU+P
- Default
- Cars
- no pheromone
- Classes

Graph showing the comparison of the performance of DSU and DSU+P under different conditions (Default, Cars, no pheromone, Classes) over varying numbers of cycles.
Introduction

Description of Ant-CP

Application to the Car sequencing

Experimental Results

Conclusion
Using ACO to guide a CP search

Complementarity of CP and ACO

- Use CP for modelling the problem and for propagating and checking constraints
- Use ACO for guiding the search as a generic value ordering heuristic

First results on the car sequencing problem

- Ant-CP outperforms complete approaches:
  Complete approaches still have difficulties to solve the 70 instances of Lee (see CP’06, CP’07)
  ...whereas these instances are all quickly solved by Ant-CP.
- Ant-CP is an order slower than heuristic approaches dedicated to the car sequencing problem
  ...but the programming effort is also much smaller
Further work

- Validate our approach on other CSPs
- Adaptive parameter tuning \( \rightsquigarrow \) towards reactive ACO
  - use resampling and similarity ratio to dynamically tune parameters
- Use filtering procedures dedicated to a non backtracking search