Constraint Programming with Ant Colony Optimization

Madjid Khichane[1,2], Patrick Albert[1], and Christine Solnon[2]

[1] ILOG
[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

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

Pheromone trails

A pheromone trail τ_c is associated with every solution component c
$\rightsquigarrow$ learnt desirability of using c when building a solution

Basic principle of ACO

- initialize pheromone trails to τ_{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 $\mathcal{S}$ = partial solution
  and *cand* = candidate solution components
- Choose $c_j \in cand$ with probability

$$p(c_j) = \frac{[\tau_{\mathcal{S}}(c_j)]^{\alpha}.[\eta_{\mathcal{S}}(c_j)]^{\beta}}{\sum_{c_k \in cand}[\tau_{\mathcal{S}}(s_k)]^{\alpha}.[\eta_{\mathcal{S}}(s_k)]^{\beta}}$$

Basic principle of ACO

- initialize pheromone trails to τ_{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 $\mathcal{S}$ = partial solution
  and *cand* = candidate solution components
- Choose $c_j \in cand$ with probability

$$p(c_j) = \frac{[\tau_{\mathcal{S}}(c_j)]^{\alpha}.[\eta_{\mathcal{S}}(c_j)]^{\beta}}{\sum_{c_k \in cand}[\tau_{\mathcal{S}}(s_k)]^{\alpha}.[\eta_{\mathcal{S}}(s_k)]^{\beta}}$$

$\tau_{\mathcal{S}}(c_j) \rightsquigarrow$ pheromone factor (past experience of the colony)

Basic principle of ACO

- initialize pheromone trails to τ_{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 $\mathcal{S}$ = partial solution
  and *cand* = candidate solution components
- Choose $c_j \in cand$ with probability

$$p(c_j) = \frac{[\tau_{\mathcal{S}}(c_j)]^{\alpha}.[\eta_{\mathcal{S}}(c_j)]^{\beta}}{\sum_{c_k \in cand}[\tau_{\mathcal{S}}(s_k)]^{\alpha}.[\eta_{\mathcal{S}}(s_k)]^{\beta}}$$

$\eta_{\mathcal{S}}(c_j) \rightsquigarrow$ heuristic factor (problem-dependent)

Basic principle of ACO

- initialize pheromone trails to τ_{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 $\mathcal{S}$ = partial solution

  and *cand* = candidate solution components
- Choose $c_j \in cand$ with probability

$$p(c_j) = \frac{[\tau_{\mathcal{S}}(c_j)]^{\alpha} . [\eta_{\mathcal{S}}(c_j)]^{\beta}}{\sum_{c_k \in cand} [\tau_{\mathcal{S}}(s_k)]^{\alpha} . [\eta_{\mathcal{S}}(s_k)]^{\beta}}$$

α, $\beta \rightsquigarrow$ factor weights (parameters)

Basic principle of ACO

- initialize pheromone trails to τ_{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 $(1 - \rho)$
  $\rightsquigarrow$ ρ = evaporation rate ($0 \leq \rho \leq 1$)
- Reward: add pheromone on components of the best solutions
- Bound pheromone trails between τ_{min} and τ_{max}
  $\rightsquigarrow$ 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

Ant-CP procedure

- initialize pheromone trails to τ_{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 τ_{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:

$\tau_{\langle X_i, v_i \rangle}$ = 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 τ_{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

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

  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 τ_{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 $(1 - \rho)$
  $\rightsquigarrow$ ρ = 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

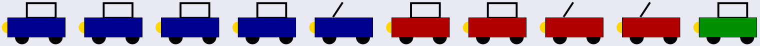
The car sequencing problem

Goal: Sequence cars along an assembly line

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

Example

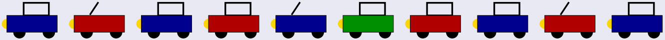
Set of cars to be sequenced:



Sequencing contraints:

 $\leq$ *1/2* ;  $\leq$ *2/5* ;  $\leq$ *1/5* ;  $\leq$ *1/3*

Solution:



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 = class of the ith 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, $\forall$ subseq. s of q_j cars, $\sum_{i \in s} opt_{ij} \leq p_j$

Pheromone structures

Default vs specific pheromone structures

- Default $\rightsquigarrow$ pheromone is laid on variable/value couples
- Specific $\rightsquigarrow$ the user has to define
  - a set of pheromone trails
  - a function τ $\rightsquigarrow$ pheromone factors
  - a function *comp* $\rightsquigarrow$ rewarded components

Comparison of 4 pheromone structures

- Default
- Classes [Gravel et al 04]
  $\rightsquigarrow$ pheromone is laid on couples of consecutive car classes
- Cars [Solnon 00]
  $\rightsquigarrow$ pheromone is laid on couples of consecutive cars
- Empty
  $\rightsquigarrow$ pheromone is not used

Heuristics

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

- $UR(o_i)$ = number of required slots / number of available slots
- $UR(o_i) > 1 \rightsquigarrow$ no solution

Comparison of 2 heuristics

- DSU = sum of utilization rates of required options
  $\rightsquigarrow$ 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 $UR(o_i) = 1$

Table of contents

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)

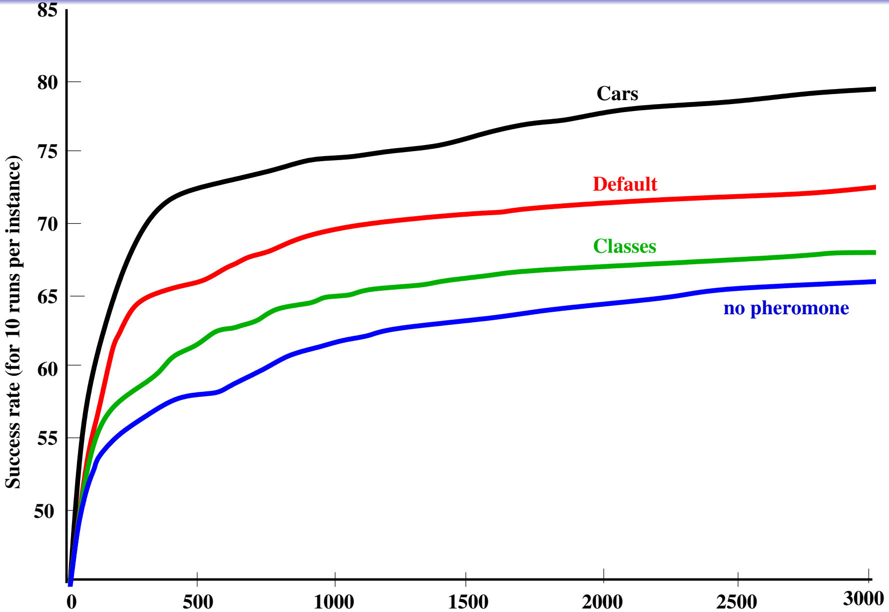
$\rightsquigarrow$ these instances are too easy to evaluate Ant-CP

Instances of [Perron & Shaw 04]

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

$\rightsquigarrow$ nearly half of these instances are difficult

Comparison of the 4 pheromone structures with the DSU heuristic



Comparison of the 2 heuristics

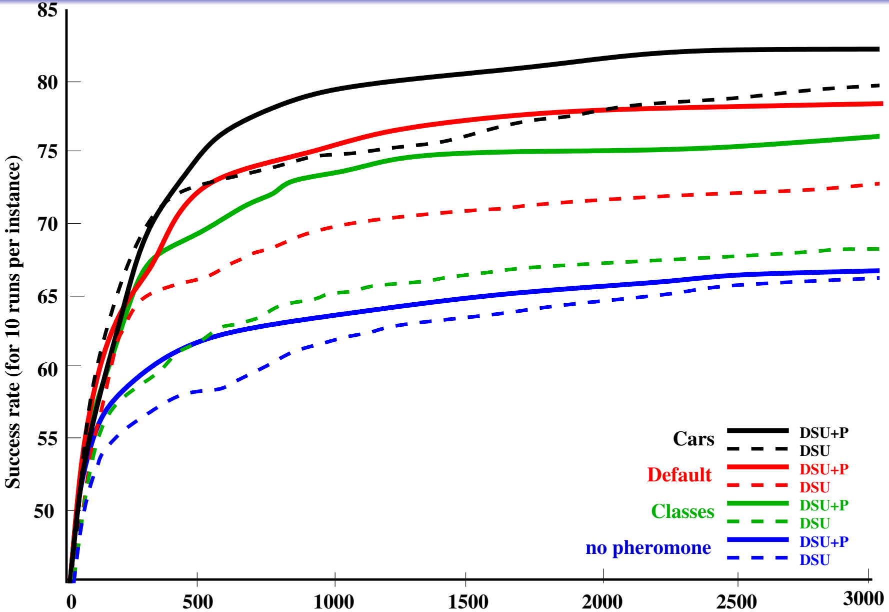


Table of contents

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 $\leadsto$ towards reactive ACO
  - use resampling and similarity ratio to dynamically tune parameters
- Use filtering procedures dedicated to a non backtracking search