Ant Colony Optimization for Constraint Satisfaction

Christine Solnon

LIRIS, UMR 5205 CNRS / University of Lyon

Tutorial at CP'2007

Table of contents

A metaheuristic inspired by real ants

Foraging behavior of real ants

Simple and autonomous agents

+ Indirect communication via the environment (stigmergy)

$$\begin{cases} \textit{Ants lay pheromone trails while walking} \\ \textit{Ants randomly choose paths w.r.t. phero. trails} \\ \textit{Pheromone trails evaporate} \end{cases}$$

$\Longrightarrow$ Shortest path between the nest and a food source

$\rightsquigarrow$ **Collective problem solving / swarm intelligence**

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

How to solve a problem with ACO ?

1) Define a construction graph

- Solution components $\rightsquigarrow$ vertices or edges
- Solution $\rightsquigarrow$ best path in the graph

2) Use artificial ants to search for good paths

- Behavior inspired by real ants...

  Ants lay pheromone on vertices and/or edges
  Greedy randomized construction of paths w.r.t. pheromone
  Pheromone evaporates

- ...with extra capabilities

  Pheromone laying is delayed and proportional to solution quality,
  Pheromone is combined with problem-dependent heuristics,
  Hybridation with local search, ...

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System

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

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System

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

Greedy randomized construction of a path

- Let $\mathcal{C}$ = visited vertices and *cand* = candidate vertices
- Choose $v_j \in cand$ with probability

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

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System

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

Greedy randomized construction of a path

- Let $\mathcal{C}$ = visited vertices and *cand* = candidate vertices
- Choose $v_j \in cand$ with probability

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

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

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System

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

Greedy randomized construction of a path

- Let $\mathcal{C}$ = visited vertices and *cand* = candidate vertices
- Choose $v_j \in cand$ with probability

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

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

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System

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

Greedy randomized construction of a path

- Let $\mathcal{C}$ = visited vertices and *cand* = candidate vertices
- Choose $v_j \in cand$ with probability

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

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

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System

- initialize pheromone trails to τ_{max}
- repeat
  1. each ant builds a path
  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 the best path components
- Bound pheromone trails between τ_{min} and τ_{max}
  $\rightsquigarrow$ prevent from premature stagnation

Example: Travelling Salesman Problem

Construction graph

- Complete graph that associates a vertex with each city
- Pheromone is laid on edges:
  $\tau(i,j) \rightsquigarrow$ 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

Example: Travelling Salesman Problem

Construction graph

- Complete graph that associates a vertex with each city
- Pheromone is laid on edges:
  $\tau(i,j) \rightsquigarrow$ 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

Example: Travelling Salesman Problem

Construction graph

- Complete graph that associates a vertex with each city
- Pheromone is laid on edges:
  $\tau(i,j) \rightsquigarrow$ 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

ACO w.r.t. other (meta)heuristic approaches

Instance-based approaches

Construction of new solutions by modifying existing ones

- Genetic algorithms: cross-over + mutation
- Local search: local moves

Model-based approaches

Construction of new solutions using a probabilistic model

- Greedy randomized construction:
  ⇝ static model
- Greedy Randomized Adaptive Search Procedure (GRASP):
  ⇝ adaptive model
- Ant Colony Optimization (ACO):
  ⇝ model biased by previous experience
- Estimation of Distribution Algorithms (EDA):
  ⇝ model built from a population which evolves

ACO w.r.t. other (meta)heuristic approaches

Instance-based approaches

Construction of new solutions by modifying existing ones

- Genetic algorithms: cross-over + mutation
- Local search: local moves

Model-based approaches

Construction of new solutions using a probabilistic model

- Greedy randomized construction:
  $\rightsquigarrow$ static model
- Greedy Randomized Adaptive Search Procedure (GRASP):
  $\rightsquigarrow$ adaptive model
- Ant Colony Optimization (ACO):
  $\rightsquigarrow$ model biased by previous experience
- Estimation of Distribution Algorithms (EDA):
  $\rightsquigarrow$ model built from a population which evolves

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

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 τ_{max}
- Risk: convergence to optimality may be too long

$\Rightarrow$ Theoretical proof of convergence to optimality

Influence of ACO parameters on intensification/diversification

τ_{min}, τ_{max} : pheromone lower and upper bounds

$\rightsquigarrow$ Intensification increases when $\tau_{max} - \tau_{min}$ increases

nbAnts : number of ants

$\rightsquigarrow$ Diversification increases when *nbAnts* increases

α : weight of the pheromone factor

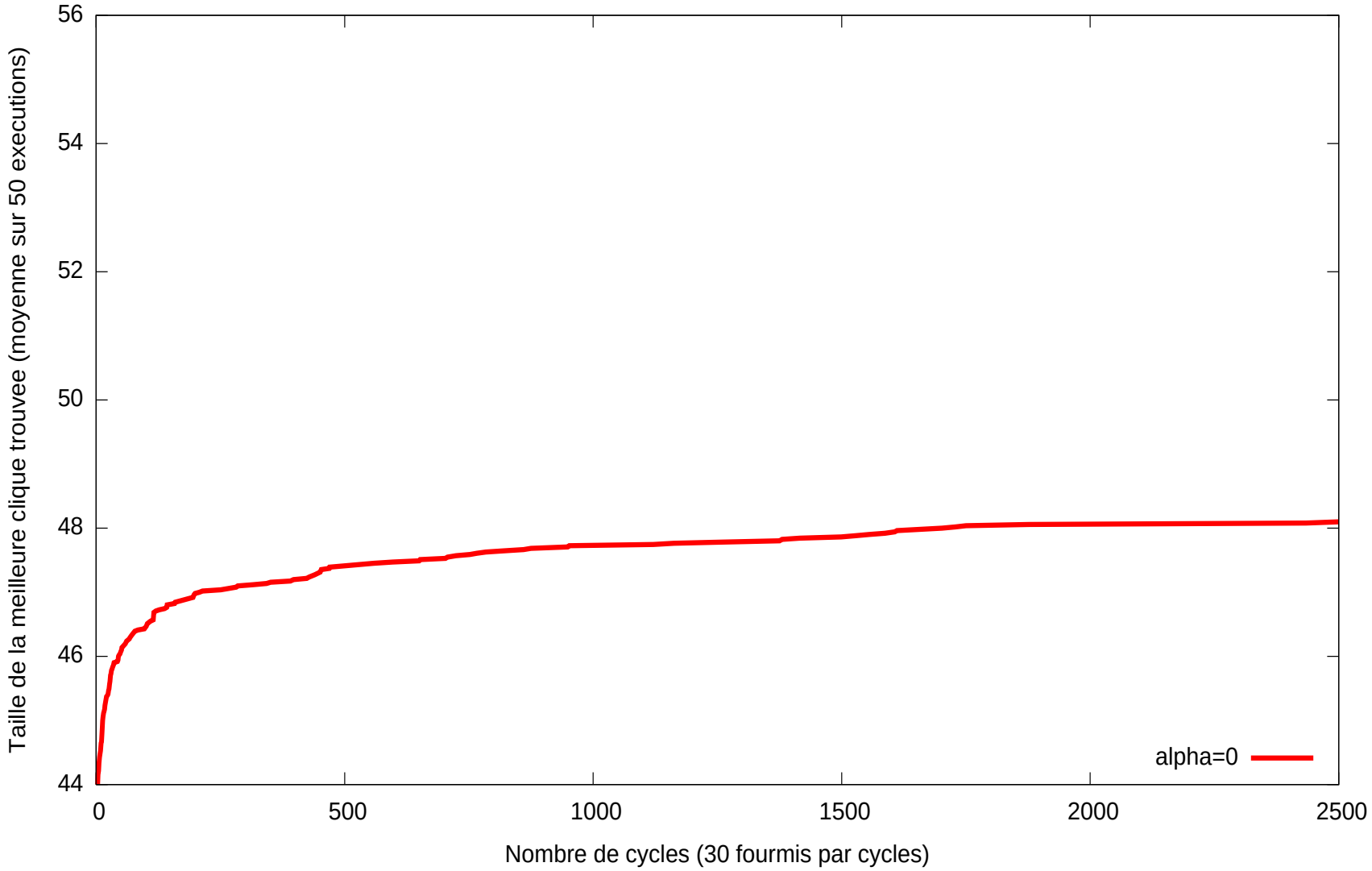
$\rightsquigarrow$ Intensification increases when α increases

ρ : pheromone evaporation rate

$\rightsquigarrow$ Intensification increases when ρ increases

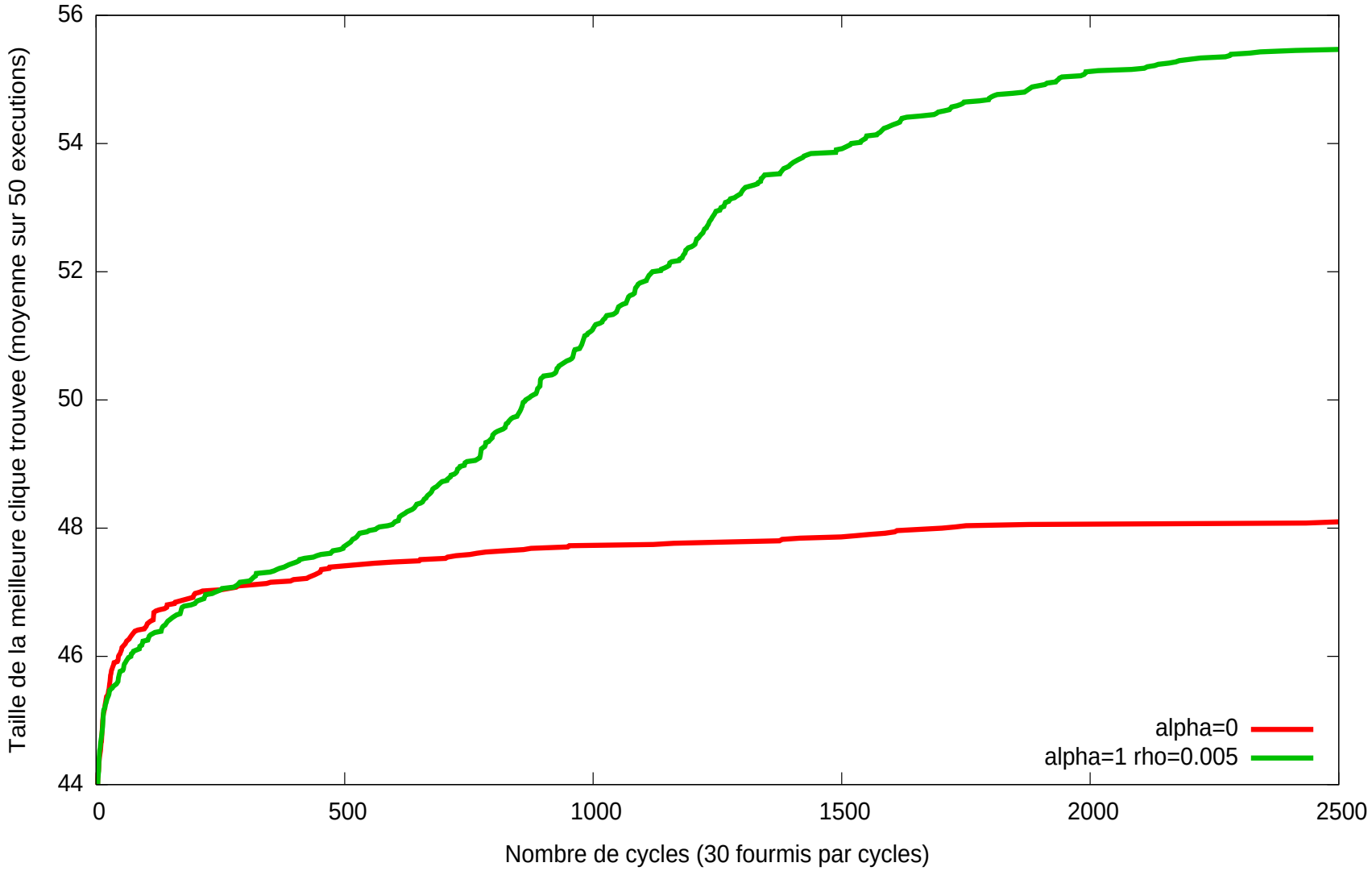
The best parameter setting depends on available time !

Illustration on a maximum clique problem



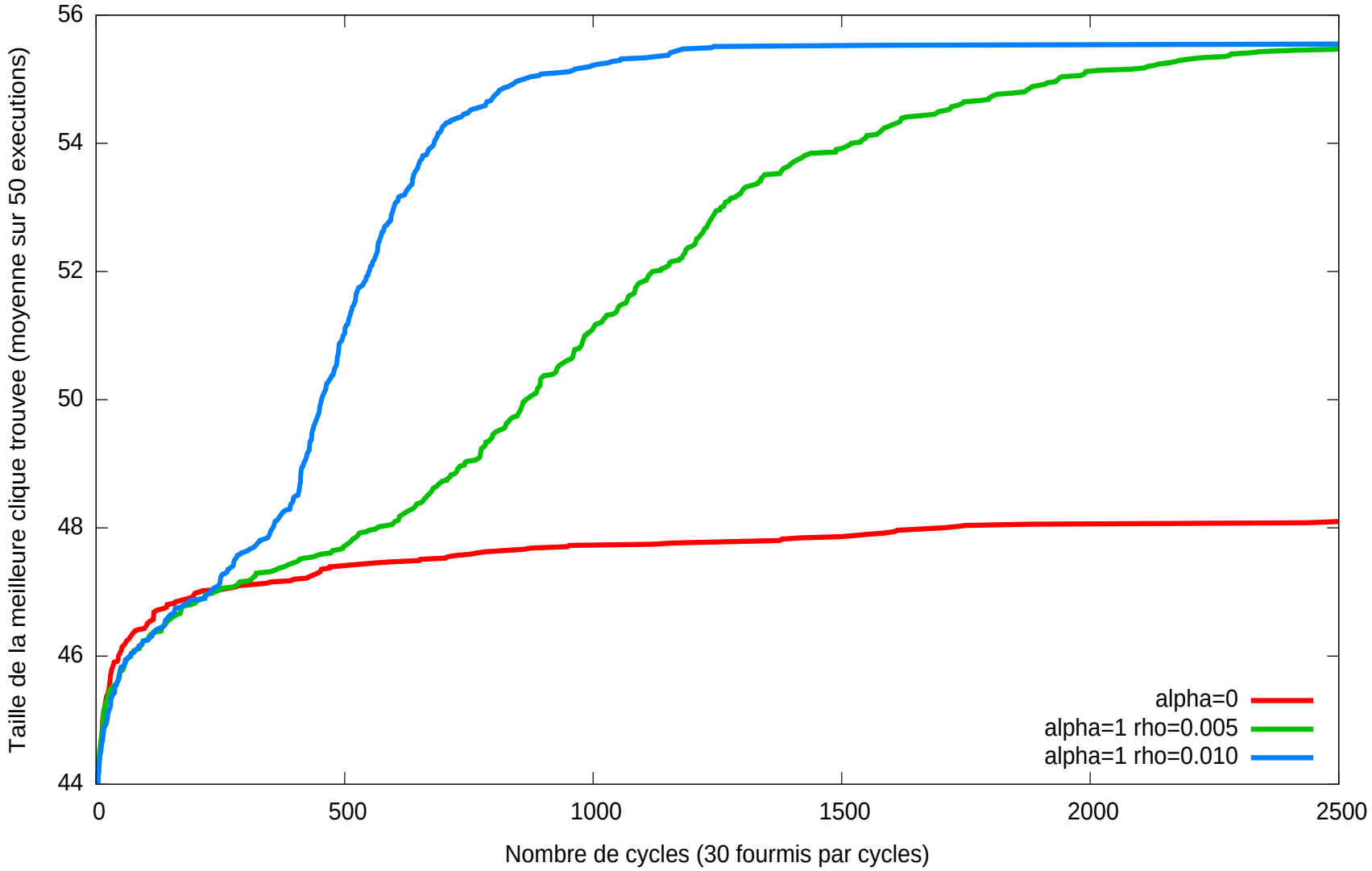
Without pheromone: $\alpha = 0$

Illustration on a maximum clique problem



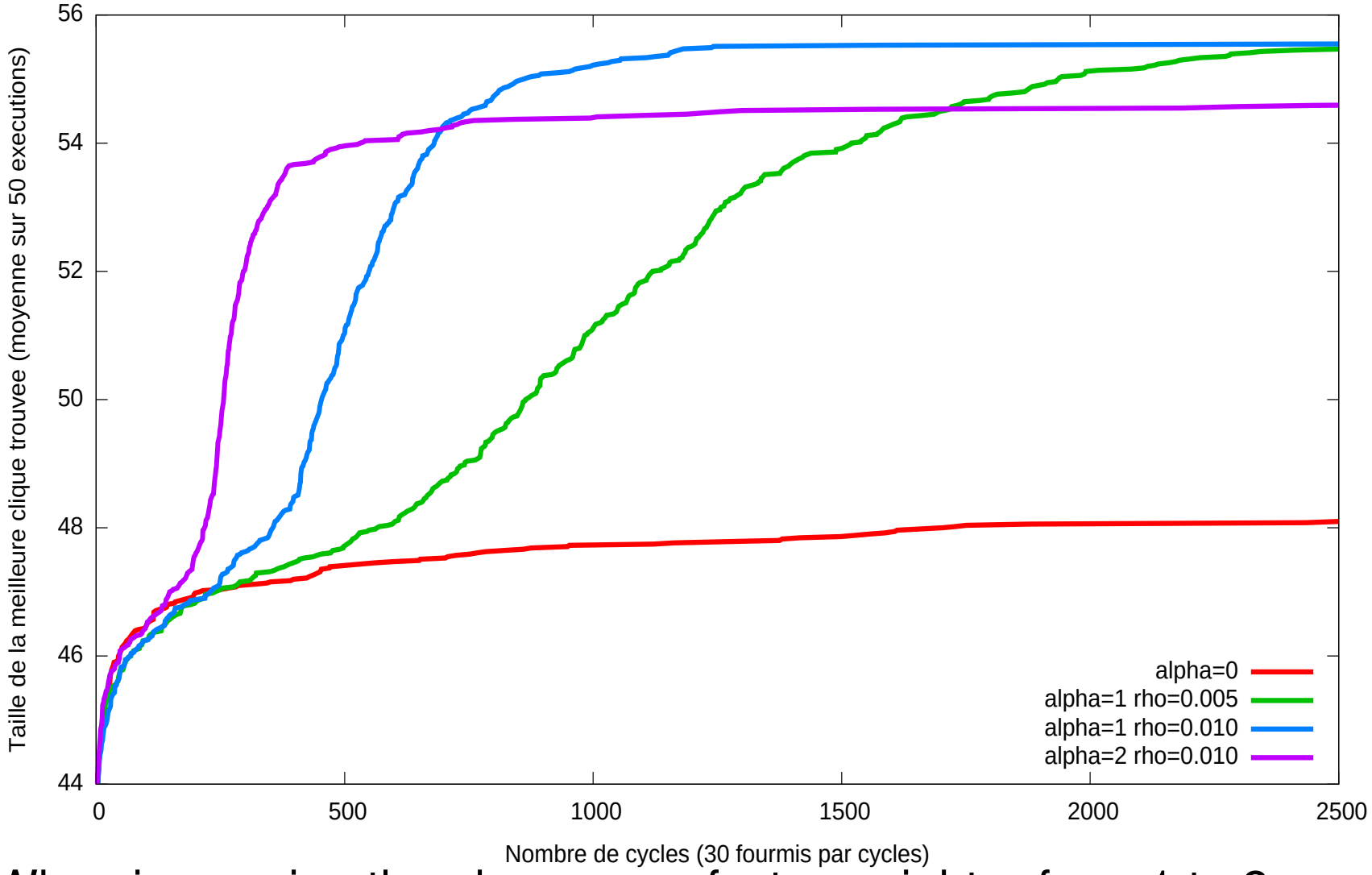
With a small influence of pheromone: $\alpha = 1$, $\rho = 0.5\%$

Illustration on a maximum clique problem



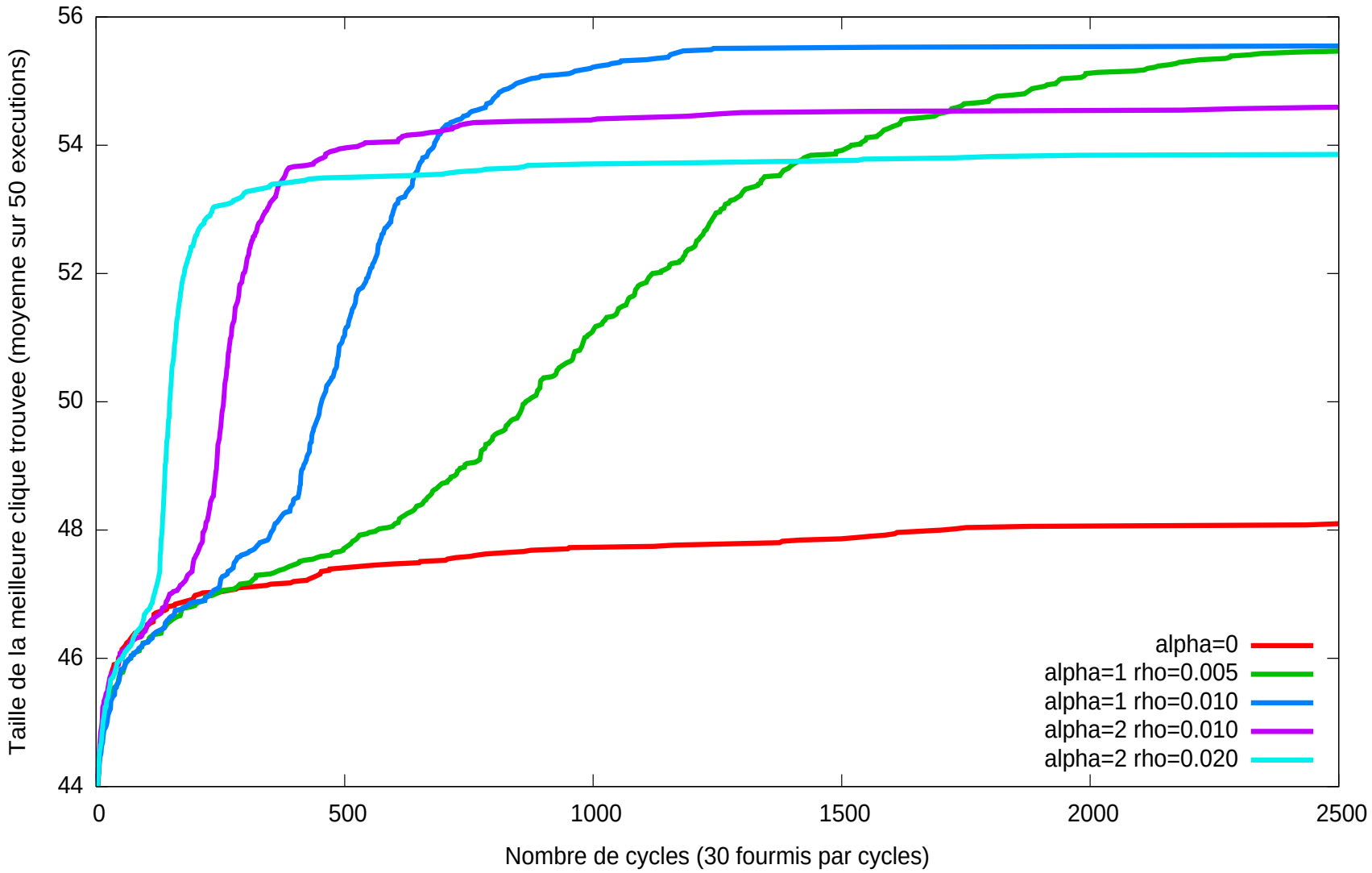
When increasing the evaporation rate ρ from 0.5% to 1%

Illustration on a maximum clique problem



When increasing the pheromone factor weight α from 1 to 2

Illustration on a maximum clique problem



When increasing the evaporation rate ρ from 1% to 2%

Measuring Intensification/Diversification

Resampling ratio (RR) $\rightsquigarrow$ quantifies diversification

- $\text{RR} = \dfrac{\#\{\text{computed solutions}\} - \#\{\text{different computed solutions}\}}{\#\{\text{computed solutions}\}}$
- Maximal diversification $\Leftarrow 0 \leq \text{RR} \leq 1 \Rightarrow$ 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

Resampling ratio (RR) $\rightsquigarrow$ quantifies diversification

- $\text{RR} = \dfrac{\#\{\text{computed solutions}\} - \#\{\text{different computed solutions}\}}{\#\{\text{computed solutions}\}}$
- Maximal diversification $\Leftarrow 0 \leq \text{RR} \leq 1 \Rightarrow$ 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

Resampling ratio (RR) $\rightsquigarrow$ quantifies diversification

- $\text{RR} = \dfrac{\#\{\text{computed solutions}\} - \#\{\text{different computed solutions}\}}{\#\{\text{computed solutions}\}}$
- Maximal diversification $\Leftarrow 0 \leq \text{RR} \leq 1 \Rightarrow$ 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

Number of cycles:	500	1000	1500	2000	2500
$\alpha = 1, \rho = 0.01$	0.00	0.00	0.00	0.00	0.00
$\alpha = 2, \rho = 0.01$	0.00	0.04	0.06	0.07	0.07
$\alpha = 2, \rho = 0.02$	0.06	0.10	0.12	0.13	0.13

Similarity ratio

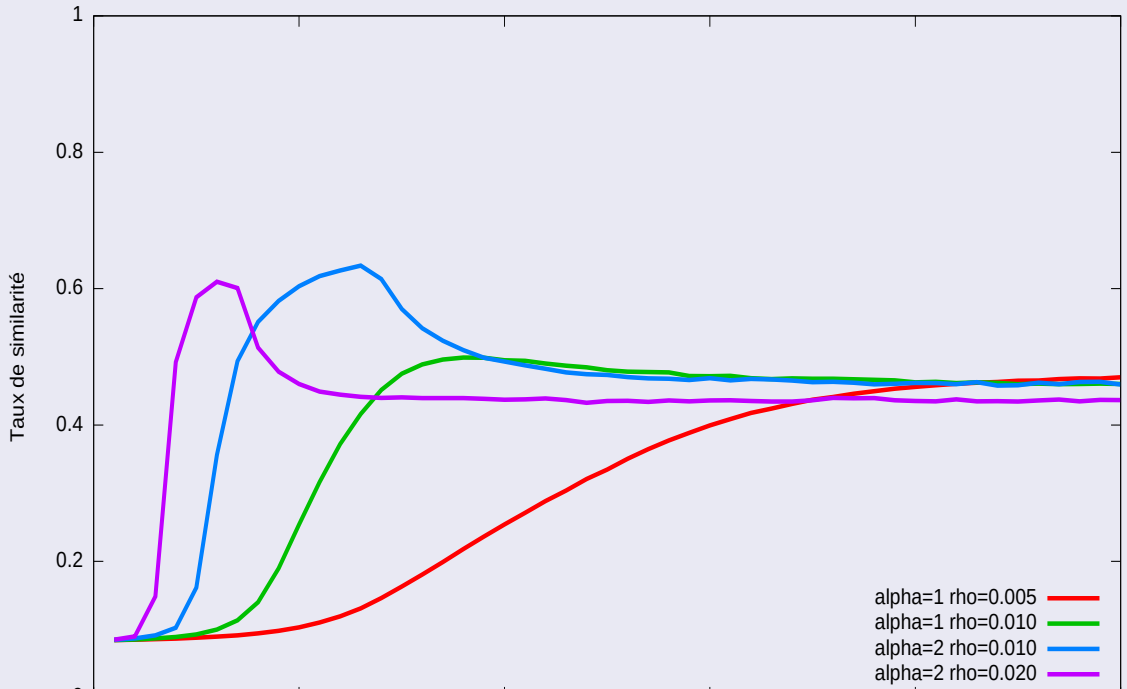


Table of contents

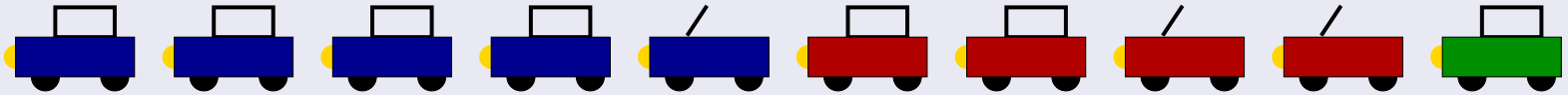
The car sequencing problem

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:



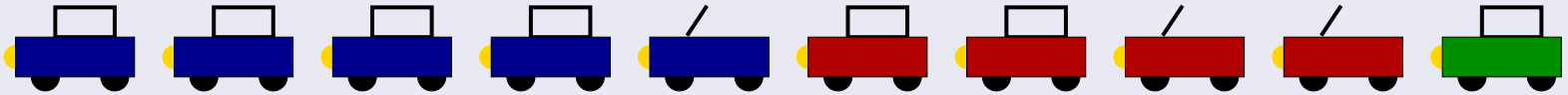
The car sequencing problem

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:

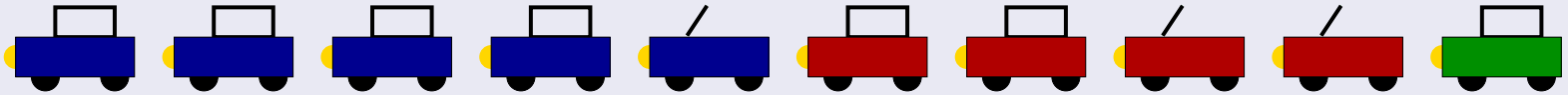
The car sequencing problem

Goal: Sequence cars along an assembly line

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

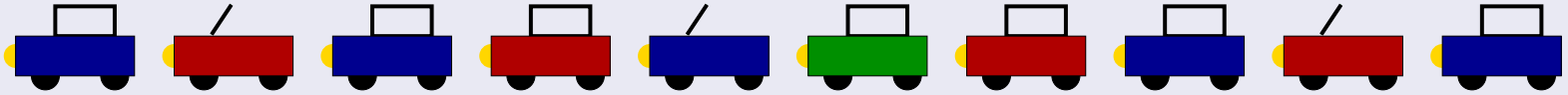
Example

Set of cars to be sequenced:



Sequencing contraints:

Solution:



ACO for the car sequencing problem

- Greedy randomized algorithm
  $\rightsquigarrow$ problem-dependent heuristic for identifying critical cars
- ACO 1
  $\rightsquigarrow$ pheromone structure for identifying good car sequences
- ACO 2
  $\rightsquigarrow$ pheromone structure for identifying critical cars
- ACO 1+2
  $\rightsquigarrow$ combining the two pheromone structures

Greedy randomized algorithm

- Start from an empty sequence π
- While not all cars have been sequenced in π:
  - Let *cand* be the set of cars not sequenced in π
  - narrow *cand* to 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 π

Greedy randomized algorithm

- Start from an empty sequence π
- While not all cars have been sequenced in π:
  - Let *cand* be the set of cars not sequenced in π
  - narrow *cand* to 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 π

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

where

- $\eta(c_i, \pi)$ = problem-dependent heuristic function
  $\rightsquigarrow$ sum of utilisation rates of options required by c_i
- β = parameter that controls heuristic weight

ACO algorithm for learning good sequences

Construction graph

- Complete directed graph s.t. vertices = cars
  $\rightsquigarrow$ Hamiltonian path = sequence of cars
- Pheromone is laid on edges: $\tau_1(c_i, c_j)$ = pheromone on (c_i, c_j)
  $\rightsquigarrow$ experience of the colony / sequencing c_j just after c_i

At each cycle, each ant builds a sequence

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

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

ACO algorithm for learning good sequences

Construction graph

- Complete directed graph s.t. vertices = cars
  $\rightsquigarrow$ Hamiltonian path = sequence of cars
- Pheromone is laid on edges: $\tau_1(c_i, c_j)$ = pheromone on (c_i, c_j)
  $\rightsquigarrow$ experience of the colony / sequencing c_j just after c_i

At each cycle, each ant builds a sequence

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

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

ACO algorithm for learning good sequences

Construction graph

- Complete directed graph s.t. vertices = cars
  $\rightsquigarrow$ Hamiltonian path = sequence of cars
- Pheromone is laid on edges: $\tau_1(c_i, c_j)$ = pheromone on (c_i, c_j)
  $\rightsquigarrow$ experience of the colony / sequencing c_j just after c_i

At each cycle, each ant builds a sequence

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

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

$\tau_\pi(c_j)$ = pheromone factor
$\rightsquigarrow$ if the last car of π is c_i then $\tau_\pi(c_j) = \tau_1(c_i, c_j)$
$\rightsquigarrow$ when choosing the first car of a sequence, $\tau_\pi(c_j) = 1$

ACO algorithm for learning good sequences

Construction graph

- Complete directed graph s.t. vertices = cars
  $\rightsquigarrow$ Hamiltonian path = sequence of cars
- Pheromone is laid on edges: $\tau_1(c_i, c_j)$ = pheromone on (c_i, c_j)
  $\rightsquigarrow$ experience of the colony / sequencing c_j just after c_i

At each cycle, each ant builds a sequence

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

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

$\eta_\pi(c_j)$ = local heuristic that evaluates the hardness of c_j
$\rightsquigarrow$ sum of utilization rates of options required by c_j

ACO algorithm for learning good sequences

Construction graph

- Complete directed graph s.t. vertices = cars
  $\rightsquigarrow$ Hamiltonian path = sequence of cars
- Pheromone is laid on edges: $\tau_1(c_i, c_j)$ = pheromone on (c_i, c_j)
  $\rightsquigarrow$ experience of the colony / sequencing c_j just after c_i

At each cycle, each ant builds a sequence

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

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

At the end of each cycle, reward the best sequences

Add pheromone on consecutive cars in the cycle best sequences
Quantity of pheromone added $= 1/$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) :
$\rightsquigarrow$ $\tau_2(cc)$ = quantity of pheromone associated with car class cc
$\rightsquigarrow$ 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

$$p(cc_j) = \frac{[\tau_2(cc_j)]^{\alpha_2}}{\sum_{cc_k \in cand} [\tau(cc_k)]^{\alpha_2}}$$

Pheromone updating step

- While constructing sequences:
  $\rightsquigarrow$ add pheromone on classes violating constraints
- At the end of every sequence construction:
  $\rightsquigarrow$ evaporation

ACO algorithm for learning for critical cars

Pheromone structure

Pheromone trails associated with cars
(grouped in car classes w.r.t. required options) :
$\rightsquigarrow$ $\tau_2(cc)$ = quantity of pheromone associated with car class cc
$\rightsquigarrow$ 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

$$p(cc_j) = \frac{[\tau_2(cc_j)]^{\alpha_2}}{\sum_{cc_k \in cand}[\tau(cc_k)]^{\alpha_2}}$$

Pheromone updating step

- While constructing sequences:
  $\rightsquigarrow$ add pheromone on classes violating constraints
- At the end of every sequence construction:
  $\rightsquigarrow$ evaporation

ACO algorithm for learning for critical cars

Pheromone structure

Pheromone trails associated with cars
(grouped in car classes w.r.t. required options) :
$\rightsquigarrow$ $\tau_2(cc)$ = quantity of pheromone associated with car class cc
$\rightsquigarrow$ 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

$$p(cc_j) = \frac{[\tau_2(cc_j)]^{\alpha_2}}{\sum_{cc_k \in cand}[\tau(cc_k)]^{\alpha_2}}$$

Pheromone updating step

- While constructing sequences:
  $\rightsquigarrow$ add pheromone on classes violating constraints
- At the end of every sequence construction:
  $\rightsquigarrow$ evaporation

Double ACO algorithm

Combine the two pheromone structures

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

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

Pheromone updating steps

$\rightsquigarrow$ While constructing sequences: add trails on $\tau_2(cc)$
$\rightsquigarrow$ At the end of every sequence construction: evaporate $\tau_2(cc)$
$\rightsquigarrow$ 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: $\forall$ cars c_i and c_j
  $\tau_1(c_i, c_j)$ = experience of the colony / sequence c_j after c_i
- To learn for critical cars: $\forall$ car class cc
  $\tau_2(cc)$ = 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

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

Pheromone updating steps

$\rightsquigarrow$ While constructing sequences: add trails on $\tau_2(cc)$
$\rightsquigarrow$ At the end of every sequence construction: evaporate $\tau_2(cc)$
$\rightsquigarrow$ 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: $\forall$ cars c_i and c_j
  $\tau_1(c_i, c_j)$ = experience of the colony / sequence c_j after c_i
- To learn for critical cars: $\forall$ car class cc
  $\tau_2(cc)$ = 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

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

Pheromone updating steps

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

Experimental results

Exerimental settings

Algo	Heur. η β	Pheromone 1 α_1	ρ_1	τ_{min_1}	τ_{max_1}	Pheromone 2 α_2	ρ_2	τ_{min_2}
Greedy(η)	6	-	-	-	-	-	-	-
ACO(τ_1, η)	6	2	1%	0.01	4	-	-	-
ACO(τ_2)	-	-	-	-	-	6	3%	1
ACO(τ_1, τ_2)	-	2	1%	0.01	4	6	3%	1

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

Benchmark

- Instances of [Lee et all 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

Experimental results

Exerimental settings

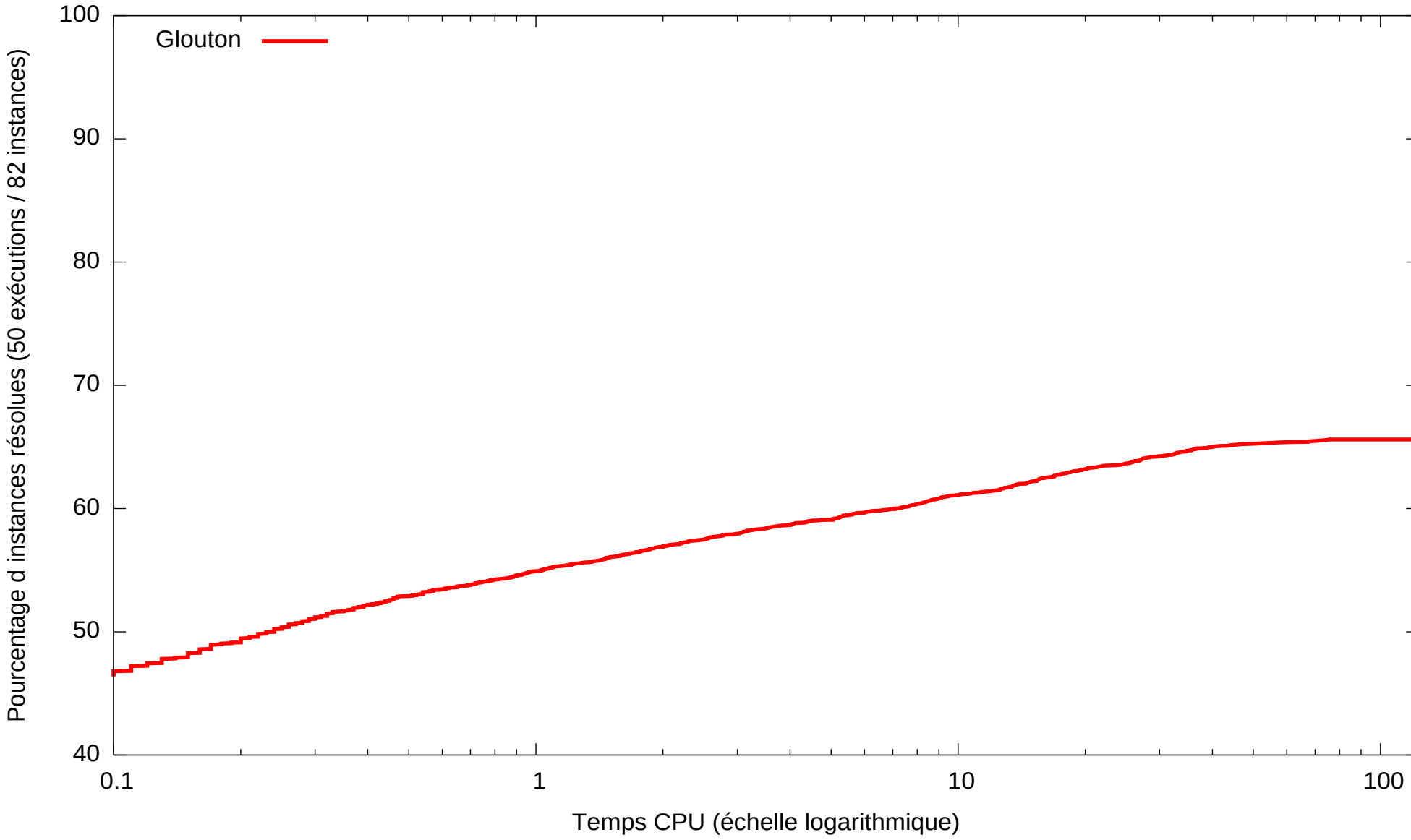
Algo	Heur. η β	Pheromone 1 α_1	ρ_1	τ_{min_1}	τ_{max_1}	Pheromone 2 α_2	ρ_2	τ_{min_2}
Greedy(η)	6	-	-	-	-	-	-	-
ACO(τ_1, η)	6	2	1%	0.01	4	-	-	-
ACO(τ_2)	-	-	-	-	-	6	3%	1
ACO(τ_1, τ_2)	-	2	1%	0.01	4	6	3%	1

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

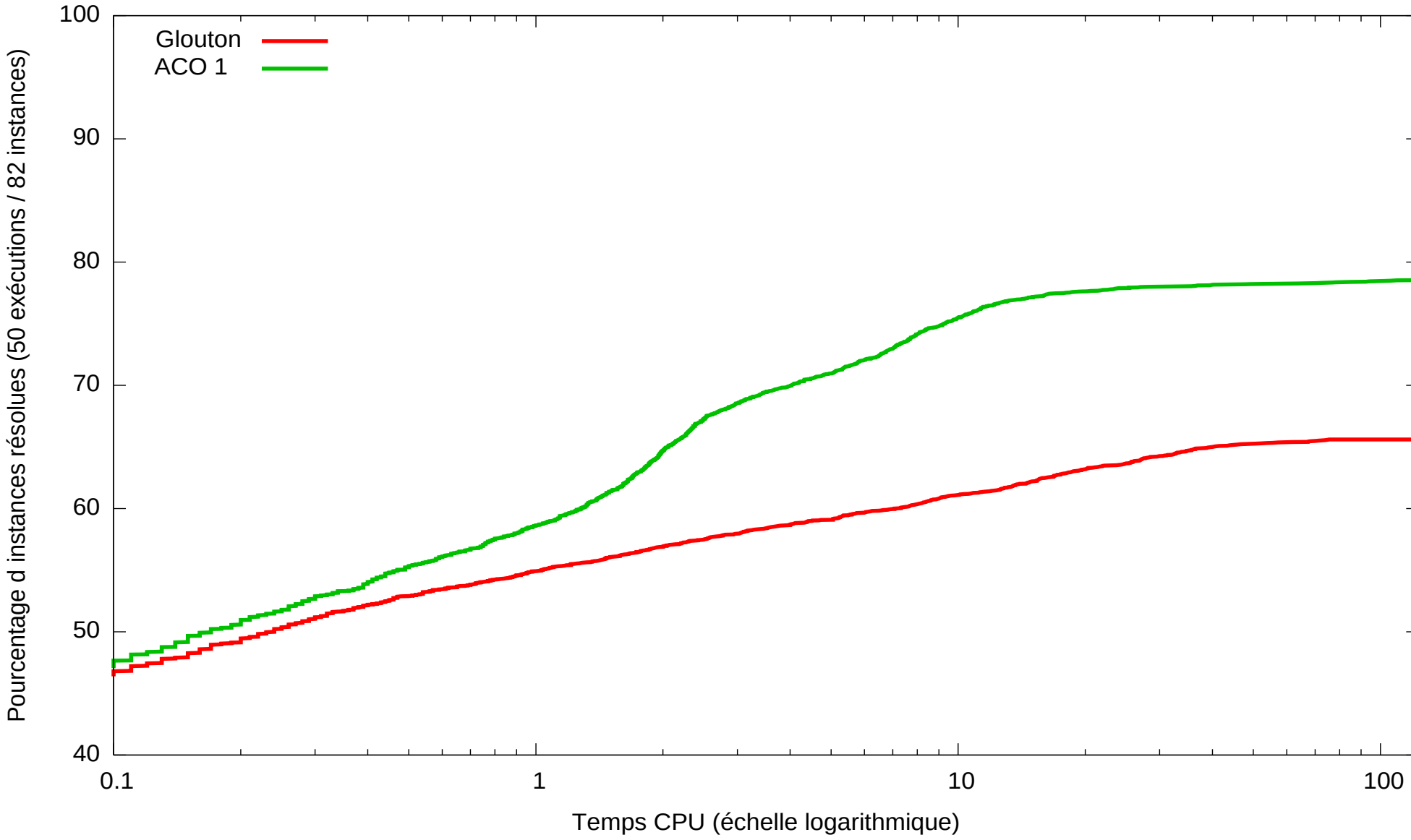
Benchmark

- Instances of [Lee et all 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

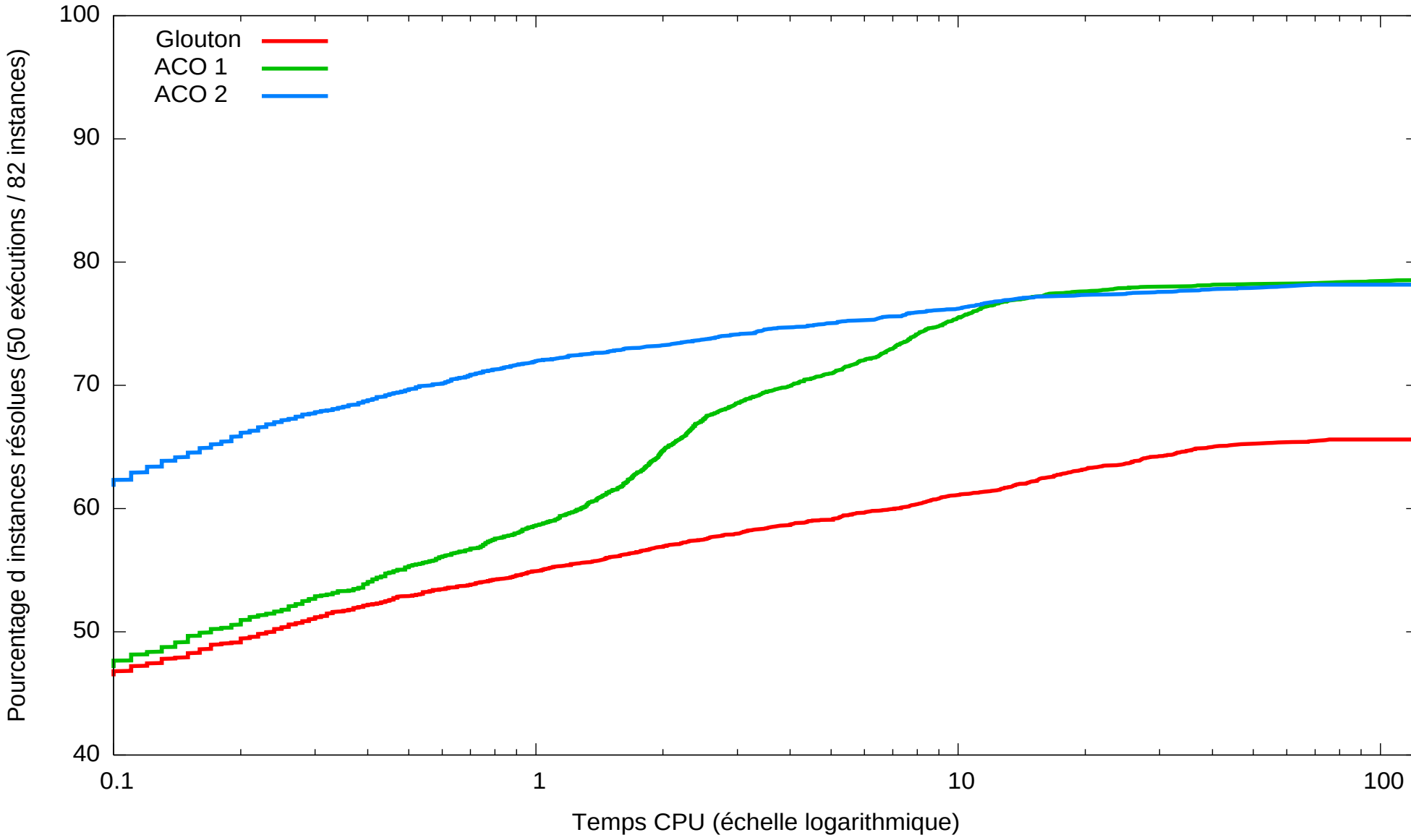
Comparison: success / CPU time



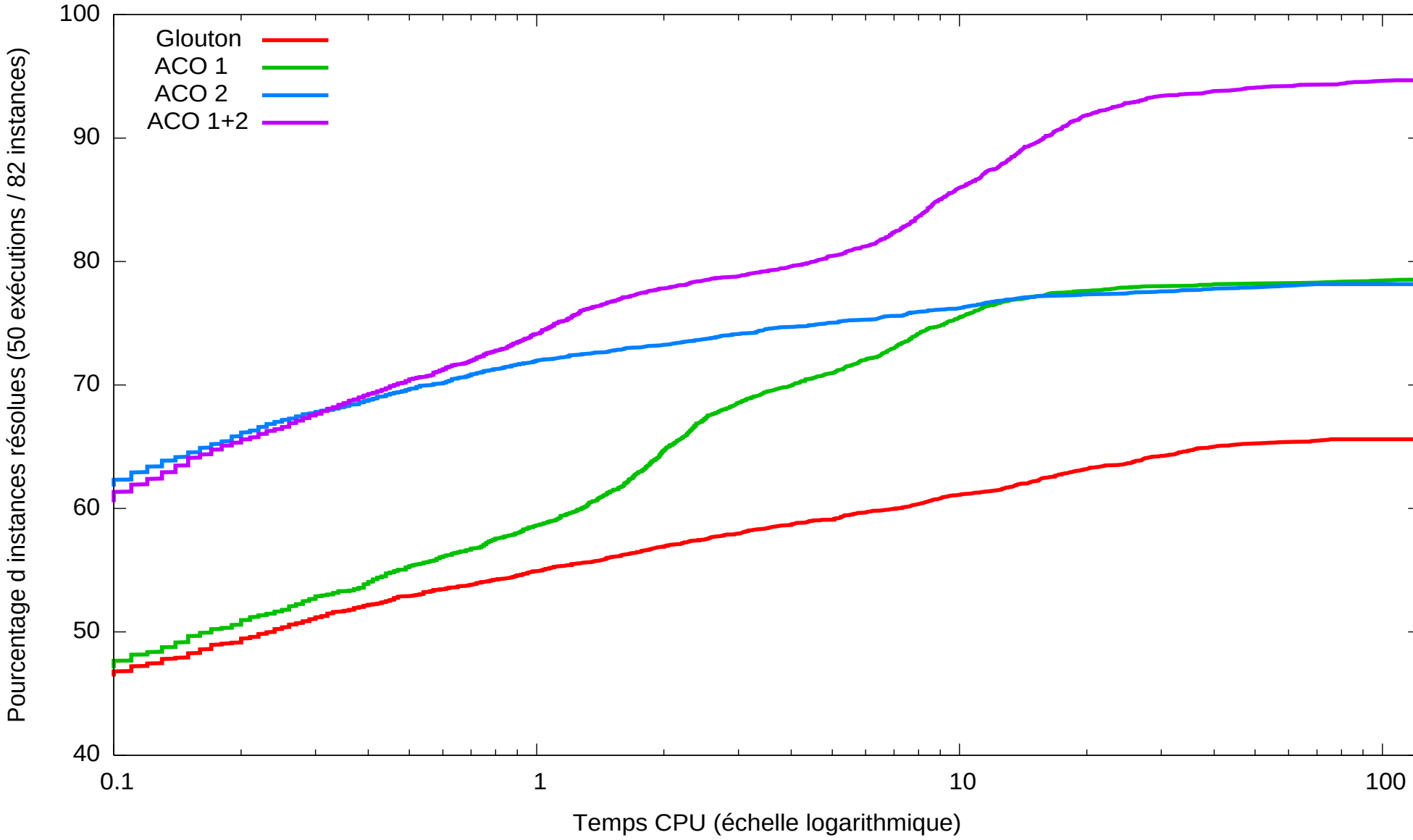
Comparison: success / CPU time



Comparison: success / CPU time



Comparison: success / CPU time

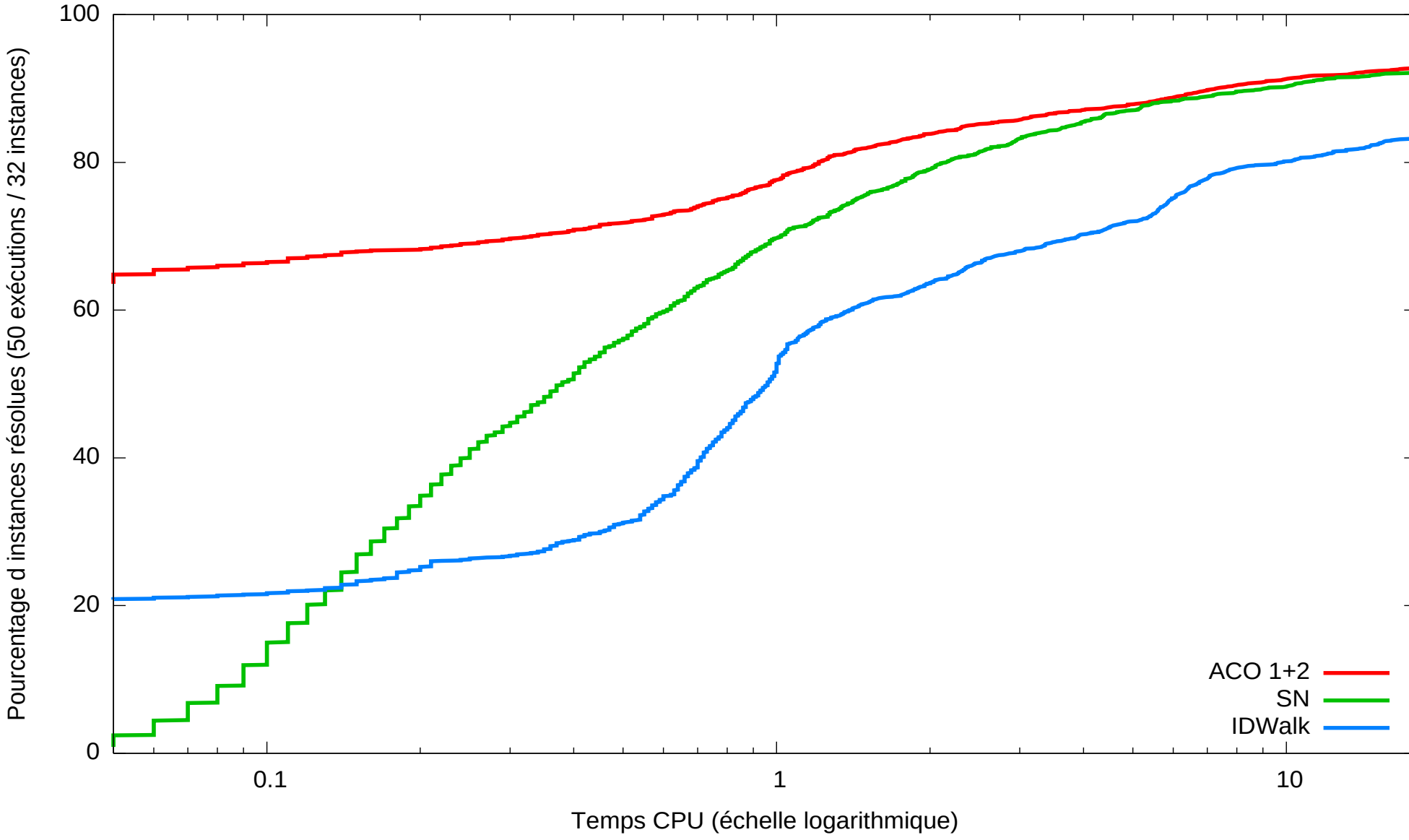


Comparison with Local Search

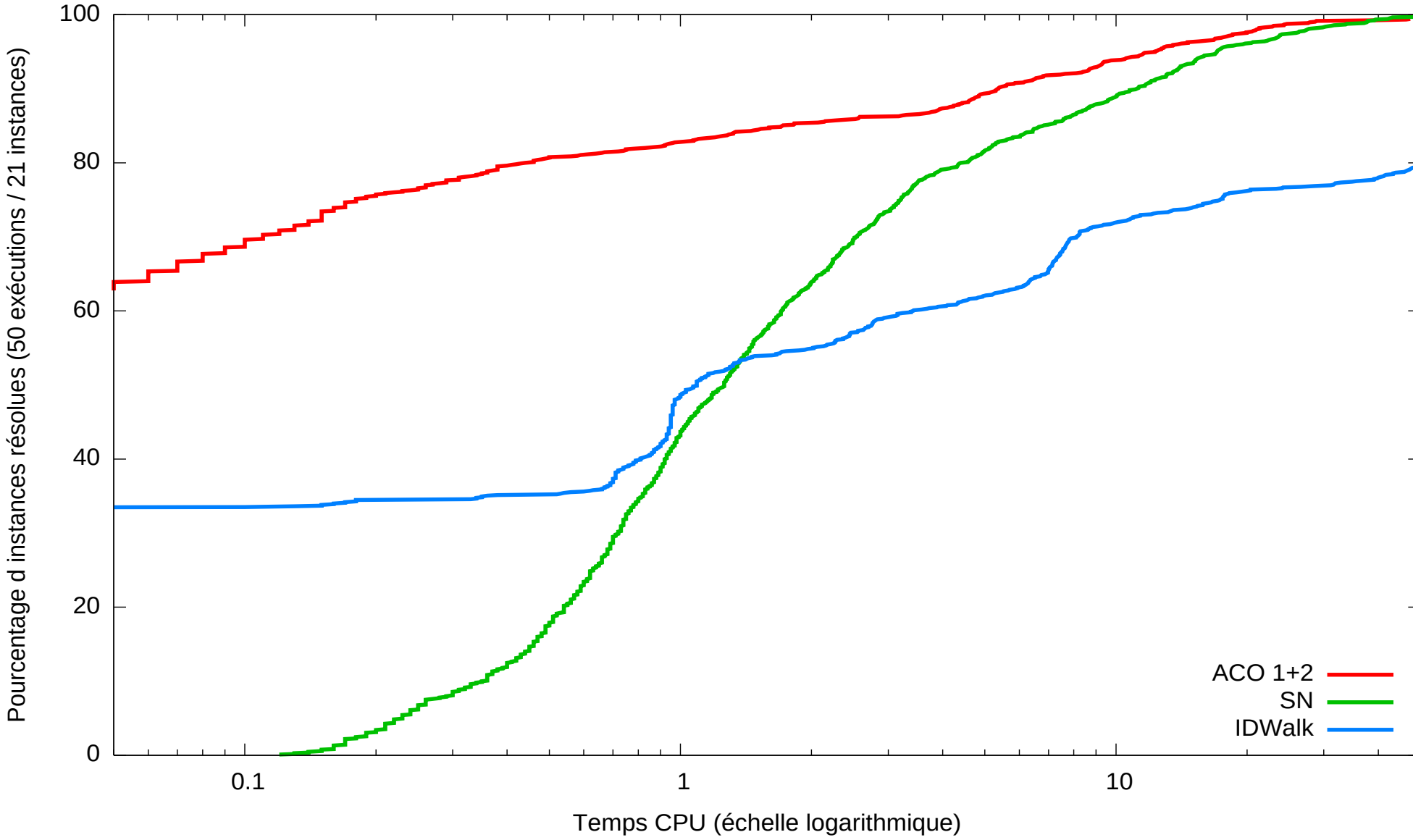
Considered approaches

- 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
    $\rightsquigarrow$ First non decreasing neighbor...
    ... or best neighbor over *Max* neighbors
    $\rightsquigarrow$ Adaptive tuning of *Max*

Comparison: 32 instances with 100 cars



Comparison: 21 instances with 300 cars



Comparison: 29 instances with 500 cars

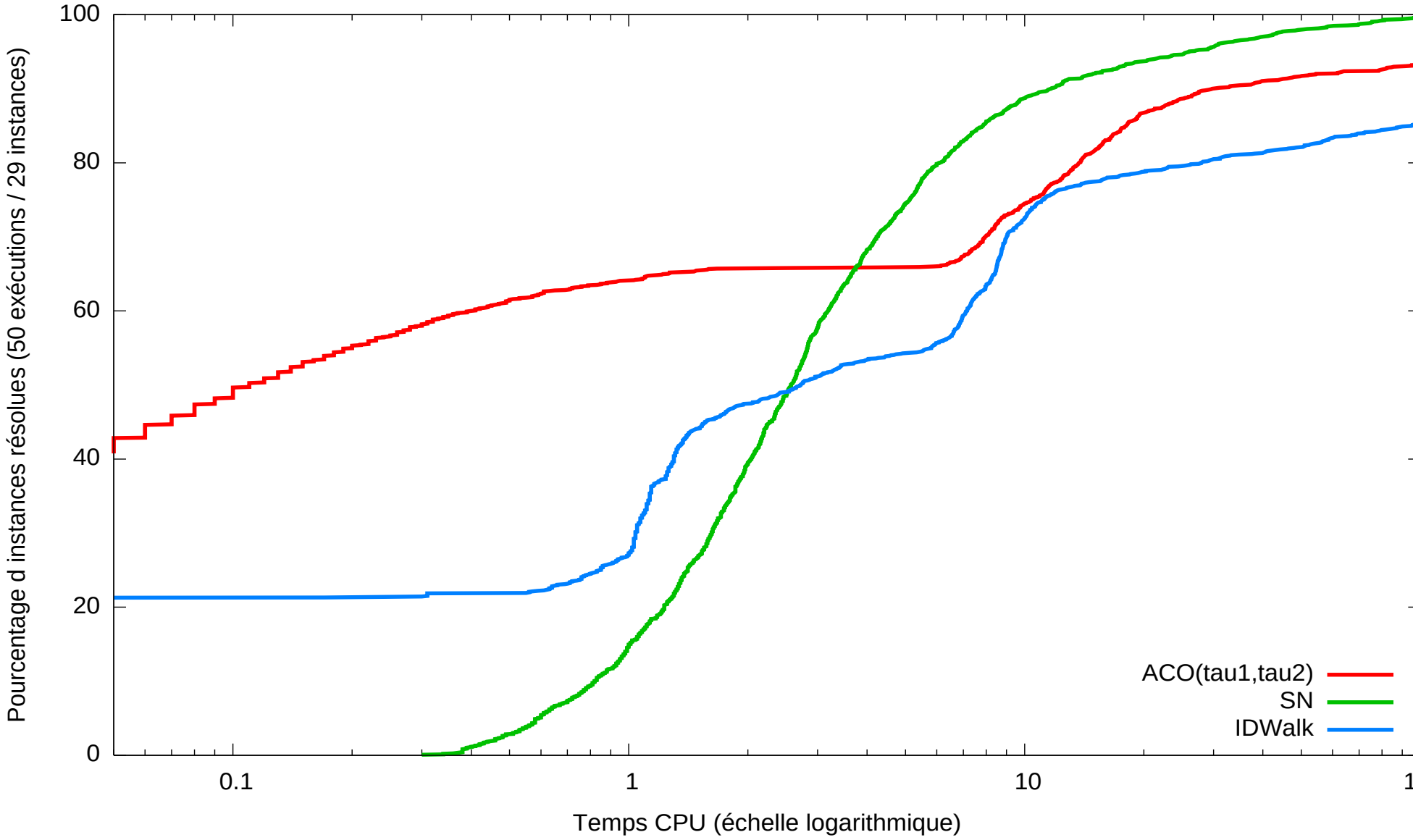


Table of contents

Application of ACO to binary CSPs

Construction graph associated with a CSP (X, D, C)

Complete non directed graph $G = (V, E)$ s.t.

$$V = \{\langle X_i, v_i \rangle : X_i \in X, v_i \in D(X_i)\}$$

Two different pheromone strategies

- Vertex strategy: pheromone is laid on vertices

  $\tau_{\langle X_i, v_i \rangle} \rightsquigarrow$ desirability of assigning X_i to v_i
- Edge strategy: pheromone is laid on edges

  $\tau_{\langle X_i, v_i \rangle, \langle X_j, v_j \rangle} \rightsquigarrow$ desir. of assigning together X_i to v_i and X_j to v_j

Application of ACO to CSP

At each cycle, each ant constructs a complete assignment A

Given a partial assignment A:

- Choose a non assigned variable X_i w.r.t. min domain heuristic
- Choose a value $v_i \in D(X_i)$ with probability

$$p(v_i) = \frac{\tau_{factor}(v_i)^{\alpha} \cdot \eta_{factor}(v_i)^{\beta}}{\sum_{v_k \in D(X_i)} \tau_{factor}(v_k)^{\alpha} \cdot \eta_{factor}(v_k)^{\beta}}$$

Application of ACO to CSP

At each cycle, each ant constructs a complete assignment A

Given a partial assignment A:

- Choose a non assigned variable X_i w.r.t. min domain heuristic
- Choose a value $v_i \in D(X_i)$ with probability

$$p(v_i) = \frac{\tau_{factor}(v_i)^{\alpha} \cdot \eta_{factor}(v_i)^{\beta}}{\sum_{v_k \in D(X_i)} \tau_{factor}(v_k)^{\alpha} \cdot \eta_{factor}(v_k)^{\beta}}$$

- Pheromone factor depends on the strategy:
  - Vertex strategy $\rightsquigarrow \tau_{factor}(v_i) = \tau_{\langle X_i, v_i \rangle}$
  - Edge strategy $\rightsquigarrow \tau_{factor}(v_i) = \sum_{\langle X_k, v_k \rangle \in A} \tau_{\langle X_i, v_i \rangle, \langle X_k, v_k \rangle}$
- Heuristic factor: $\eta_{factor}(v_i) = 1/(1 + \text{new violations})$

Application of ACO to CSP

At each cycle, each ant constructs a complete assignment A

Given a partial assignment A:

- Choose a non assigned variable X_i w.r.t. min domain heuristic
- Choose a value $v_i \in D(X_i)$ with probability

$$p(v_i) = \frac{\tau_{factor}(v_i)^{\alpha} \cdot \eta_{factor}(v_i)^{\beta}}{\sum_{v_k \in D(X_i)} \tau_{factor}(v_k)^{\alpha} \cdot \eta_{factor}(v_k)^{\beta}}$$

At the end of each cycle, update pheromone

- Evaporate pheromone
  - Vertex strategy: on every vertex $\langle X_i, v_i \rangle$
  - Edge strategy: on every edge $(\langle X_i, v_i \rangle, \langle X_k, v_k \rangle)$
- Add pheromone on the best assignment A_{best} of the cycle
  - Vertex strategy: add pheromone on every $\langle X_i, v_i \rangle \in A_{best}$
  - Edge strategy: add pheromone on every $(\langle X_i, v_i \rangle, \langle X_j, v_j \rangle) \in A_{best} \times A_{best}$

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

$\rightsquigarrow$ 230 instances from 6 benchmarks

Experimental results: ACO experimental setting

Hybridization with local search

At the end of each assignment construction, perform local search

- Move = change the value of one variable
- Accept the first non tabu and non deteriorating move
- Tabu list of infinite length

Parameter setting

- Choice of a setting that favors a quick convergence:
  $\alpha = 2$, $\beta = 8$, $\rho = 0.02$, 15 ants, $\tau_{min} = 0.01$, $\tau_{max} = \delta_{avg}/\rho$
  (where δ_{avg} = average qty of pheromone laid at each cycle)
- Exploitation of the resampling ratio to prevent from stagnation
  - When a solution is resampled, decrease pheromone trails on its components (multiplication by 0.1)
  - Restart if more than 1000 resamplings

Experimental results: considered solvers

- 23 solvers of the competition, all based on complete approaches
  ⇝ CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO (Vertex and Edge)
  ⇝ 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

Solver	#solved
buggy_2_5_s	207
buggy_2_5	207
Abscon 109 AC	207
VALCSP 3.1	206
VALCSP 3.0	206
Abscon 109 ESAC	204
...	...

6 best complete solvers of the competition in 2006

Experimental results: considered solvers

- 23 solvers of the competition, all based on complete approaches
  ⇝ CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO (Vertex and Edge)
  ⇝ 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

Solver	#solved
buggy_2_5_s	207
buggy_2_5	207
Abscon 109 AC	207
VALCSP 3.1	206
VALCSP 3.0	206
Abscon 109 ESAC	204
...	...
ACO(Vertex)	218
ACO(Edge)	214

6 best complete solvers of the competition in 2006

Experimental results: considered solvers

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

Solver	#solved
buggy_2_5_s	207
buggy_2_5	207
Abscon 109 AC	207
VALCSP 3.1	206
VALCSP 3.0	206
Abscon 109 ESAC	204
...	...
ACO(Vertex)	218
ACO(Edge)	214
Tabu	226

6 best complete solvers of the competition in 2006

Experimental results

Easy benchmarks (all instances solved in less than 1800 s.)

- composed-25-10-20 : 10 instances, $|X| = 105$, $|D| = 10$
- marc : 5 instances, $|X| \in \{80, 84, 88, 92, 96\}$, $|D| = |X|$
- rand-2-k : 23 instances, $|X| = k$, $|D| = k$, $k \in \{23, 24, 25, 26, 27\}$

Benchmark	Best complete solver		Vertex	Edge	Tabu
comp-25-10-20	**0.06**	(VALCSP 3.1)	1.31	8.03	0.32
marc	**19.95**	(Abscon 109 AC)	26.96	27.05	28.21
rand-2-23	14.82	(VALCSP 3.0)	**1.42**	3.15	5.76
rand-2-24	9.94	(VALCSP 3.1)	**3.22**	7.43	8.25
rand-2-25	52.69	(buggy_2_5_s)	**3.20**	7.27	6.71
rand-2-26	180.86	(VALCSP 3.1)	**3.26**	87.71	21.13
rand-2-27	248.71	(VALCSP 3.1)	**23.45**	193.88	26.35

(average CPU time, in seconds, for each class of instances)

Experimental results

More difficult benchmarks

- rand-2-n-k-fcd : 60 instances, $|X| = n$, $|D| = k$
- geom : 92 instances, $|X|$=50, $|D| = 20$

	Best complete solver			ACO(Vertex)		ACO(Edge)		Tabu	
	nb	time	solver	nb	time	nb	time	nb	time
2-30-15	**20**	**0.2**	(VALCSP 3.0)	**20**	0.4	**20**	0.8	**20**	0.5
2-40-19	**20**	25.3	(VALCSP 3.1)	**20**	**7.0**	**20**	98.3	**20**	8.2
2-50-23	14	829.7	(buggy_2_5)	**20**	81.9	17	347.6	**20**	**31.0**
geom	**92**	**1.0**	(VALCSP 3.0)	91	11.0	91	50.2	**92**	3.6

nb = number of instances solved in less than 1800 s.

time = average CPU time for the solved instances

Experimental results

Really hard benchmark

- frb$n$-k [Xu et al. / IJCAI'05] : 40 instances, $|X| = n$, $|D| = k$

	Best solver			ACO(Vertex)		ACO(Edge)		Tabu	
	nb	time	solver	nb	time	nb	time	nb	time
30-15	**5**	**0.2**	(VALCSP 3.1)	**5**	0.4	**5**	1.3	**5**	0.5
35-17	**5**	2.1	(VALCSP 3.1)	**5**	3.0	**5**	5.0	**5**	**0.9**
40-19	**5**	9.4	(VALCSP 3.0)	**5**	**7.0**	**5**	103.5	**5**	9.1
45-21	**5**	123.4	(VALCSP 3.1)	**5**	467.7	**5**	354.1	**5**	**43.4**
50-23	3	327.2	(VALCSP 3.1)	3	430.4	3	680.5	**4**	**9.9**
53-24	1	74.0	(Abs. 109 AC)	3	105.7	3	530.8	**4**	**291.6**
56-25	0	-	-	2	535.8	2	170.2	**4**	**329.3**
59-26	1	413.9	(Abs. 109 AC)	1	63.6	0	-	**4**	**523.7**

nb = number of instances solved in less than 1800 s.

time = average CPU time for the solved instances

Table of contents

From CSP solving to CP

Ants can solve CSPs !!!

- Competitive results...
- ...obtained with *ad hoc* ACO algorithms

Next step: CP with ACO

- Get rid of parameter tuning $\rightsquigarrow$ reactive ACO
  use diversification measures to adaptively tune parameters
- Integration within a CP language
  Investigation of two directions:
  - Use ACO to guide a CP search within Ilog solver
  - Design a Comet-based modelling language for ACO

From CSP solving to CP

Ants can solve CSPs !!!

- Competitive results...
- ...obtained with *ad hoc* ACO algorithms

Next step: CP with ACO

- Get rid of parameter tuning $\rightsquigarrow$ reactive ACO
  use diversification measures to adaptively tune parameters
- Integration within a CP language
  Investigation of two directions:
  - Use ACO to guide a CP search within Ilog solver
  - Design a Comet-based modelling language for ACO

Next step: CP with ACO

Using ACO to guide a CP search

Joint project with Ilog (Patrick Albert and Madjid Khichane)

- Use a CP modeling language to describe the problem
- Use ACO to guide the search engine

  Iterative construction of partial consistent assignments

  - ACO $\rightsquigarrow$ heuristic to choose values
  - CP solver $\rightsquigarrow$ constraint propagation and verification

First promising results on the car sequencing problem...
...see First Workshop on Autonomous Search at CP'2007

Next step: CP with ACO

A Comet-based modelling language for ACO

Joint work with Yves Deville and Pascal Van Hentenryck

- Enrich Comet with new abstractions to ease the design of ACO algorithms
- First investigations on 2 classes of problems:
  - Hamiltonian path finding problems
    (Traveling Salesman Problem, Car sequencing Problem, ...)
  - Subset selection problems
    (Multidimensional Knapsack Problem, Maximum Clique Problem, ...)

Some references

- On ACO and the MAX-MIN Ant System
  - M. Dorigo & T. Stützle: Ant Colony Optimization. MIT Press. 2004
  - http://iridia.ulb.ac.be/∼mdorigo/ACO
  - T. Stützle & H. H. Hoos: MAX-MIN Ant System. Future Generation Computer Systems. 16(8):889–914. 2000
- On solving the car sequencing problem with ACO
  - C. Solnon: Combining two ACO algorithms for solving the car sequencing problem. EJOR (to appear)
  - J. Gottlieb, M. Puchta & C. Solnon: A Study of Greedy, Local Search and ACO Approaches for Car Sequencing Problems. EvoCOP 2003.
- On solving binary CSPs with ACO
  - C. Solnon: Ants Can Solve CSPs. IEEE Transactions on Evolutionary Computation. 6(4):347–357. 2002
  - F. Tarrant & D. Bridge: When Ants Attack: Ant Algorithms for CSP. Artificial Intelligence Review. 24(3-4):455-476. 2005
  - C. Solnon & D. Bridge: An ACO Meta-Heuristic for Subset Selection Problems. Systems Engineering using Particle Swarm Optimisation. 2006.

Generalization to Subset Selection Problems

Definition

- An SS-problem is defined by (S, S_C, f) where
  - S = set of candidate objects
  - $S_C \subseteq \mathcal{P}(S)$ = set of consistant subsets
  - $f : S_c \rightarrow \mathbb{R}$ = objective funtion
- Goal: find $s^* \in S_C$ such that $f(s^*)$ is maximal (or minimal)

Examples

- Maximum clique problem
- Multidimensionnal knapsack problem
- Set covering problem
- ...
- Constraint satisfaction problems

Solving SS-problems with ACO

Vertex strategy: identify good objects

Pheromone is laid on objects: $\tau(i) \rightsquigarrow$ interest of selecting i

- Knapsack [Leguizamon & Michalewicz 99]
- Set covering [Hadji & al 00]
- Minimum weight K-trees [Blum 02]
- CSP [Solnon & Bridge 06]
- Maximum clique [Solnon & Fenet 06]

Edge strategy: identify good pairs of objects

Pheromone is laid on pairs: $\tau(i, j) \rightsquigarrow$ interest of selecting i and j

- CSP [Solnon 02]
- Minimum weight K-trees [Blum 02]
- Maximum clique [Fenet & Solnon 03]
- Knapsack [Alaya, Solnon & Ghédira 04]
- Graph matching [Sammoud, Solnon & Ghédira 05]

Generic ACO algorithm for SS-problems

Algorithm parameterized by:

$\rightsquigarrow$ an SS-problem (S, S_C, f)
$\rightsquigarrow$ a pheromone strategy $\Phi \in \{Vertex, Edge\}$

At each cycle, each ant builds a subset

Probability of adding object $o_i \in Candidats$ to subset S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

Generic ACO algorithm for SS-problems

Algorithm parameterized by:

$\rightsquigarrow$ an SS-problem (S, S_C, f)
$\rightsquigarrow$ a pheromone strategy $\Phi \in \{Vertex, Edge\}$

At each cycle, each ant builds a subset

Probability of adding object $o_i \in Candidats$ to subset S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

Generic ACO algorithm for SS-problems

Algorithm parameterized by:

$\rightsquigarrow$ an SS-problem (S, S_C, f)
$\rightsquigarrow$ a pheromone strategy $\Phi \in \{Vertex, Edge\}$

At each cycle, each ant builds a subset

Probability of adding object $o_i \in Candidats$ to subset S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

$\tau_{factor}(o_i, S_k) \rightsquigarrow$ depends on Φ

If Φ = *Vertex*:
$\tau_{factor}(o_i, S_k) = \tau(o_i)$

If Φ = *Edge*:
$\tau_{factor}(o_i, S_k) = \sum_{o_j \in S_k} \tau(o_j, o_i)$

Generic ACO algorithm for SS-problems

Algorithm parameterized by:

$\rightsquigarrow$ an SS-problem (S, S_C, f)
$\rightsquigarrow$ a pheromone strategy $\Phi \in \{Vertex, Edge\}$

At each cycle, each ant builds a subset

Probability of adding object $o_i \in Candidats$ to subset S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

$\eta_{factor}(o_i, S_k) \rightsquigarrow$ depends on the SS-problem

Generic ACO algorithm for SS-problems

Algorithm parameterized by:

$\rightsquigarrow$ an SS-problem (S, S_C, f)
$\rightsquigarrow$ a pheromone strategy $\Phi \in \{Vertex, Edge\}$

At each cycle, each ant builds a subset

Probability of adding object $o_i \in Candidats$ to subset S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

At the end of each cycle, reward best subsets.

The components of S_k to be rewarded depend on Φ

If Φ = *Vertex*:
pheromone laying on vertices

If Φ = *Edge* :
pheromone laying on clique edges