Using Ant Colony Optimization to guide a CP search

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

[1] ILOG
[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
  $\rightsquigarrow$ may spend unacceptable time to solve some instances
  $\rightsquigarrow$ 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
  $\rightsquigarrow$ may spend unacceptable time to solve some instances
  $\rightsquigarrow$ 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],
$\mathcal{MAX} - \mathcal{MIN}$ 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

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

- initialize pheromone trails to τ_{max}
- repeat
  1. each ant builds a solution
  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 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

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

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

- 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)

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

- 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)

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

- 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)

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

- 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 / 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

$\rightsquigarrow$ 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

$\rightsquigarrow$ 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

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 X_i to v_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 w.r.t. min domain heuristic
- 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

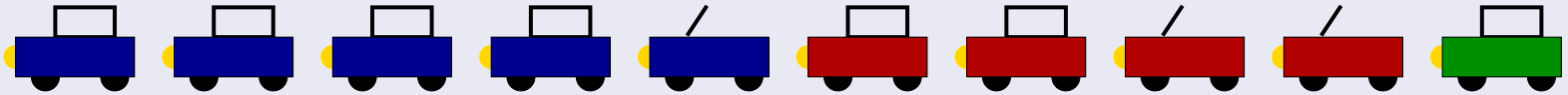
The car sequencing problem

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:



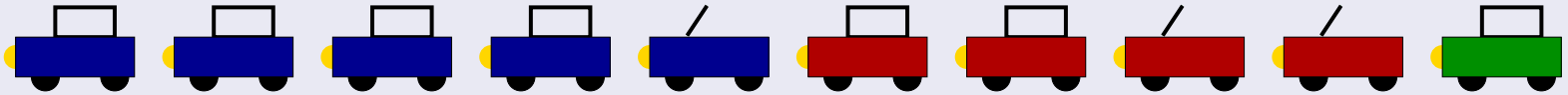
The car sequencing problem

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:

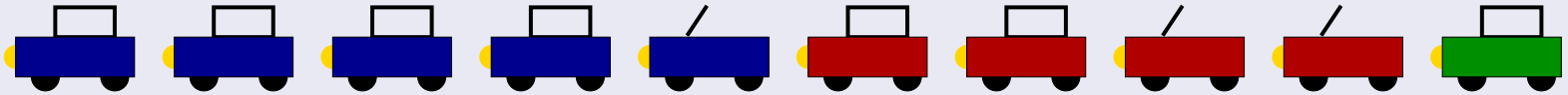
The car sequencing problem

Goal: Sequence cars along an assembly line

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

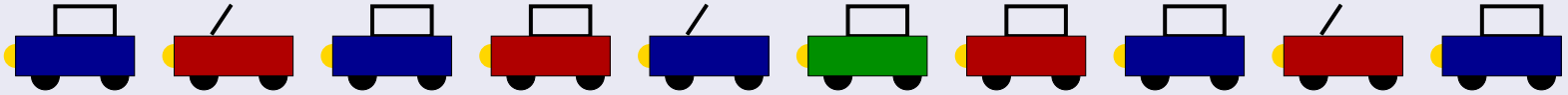
Example

Set of cars to be sequenced:



Sequencing contraints:

Solution:



CP model for the car sequencing problem

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 and $\forall$ subsequence s of q_j consecutive positions,
  $\sum_{i \in s} opt_{ij} \leq p_j$

Heuristic for the car-sequencing problem

Existing value ordering heuristic

- Heuristic proposed in [Smith 97] and used in [Régin & Puget 97]
  $\rightsquigarrow$ 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: $\eta(X_i, v_i) = 1$
  $\rightsquigarrow$ evaluate the influence of ACO on the search process
- Then, experiments with the heuristic of [Gottlieb, Puchta & Solnon 03]
  $\eta(X_i, v_i) = \sum_{o_k \text{ required by } v_i}$ 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 $\in \{60\%, 65\%, 70\%, 75\%, 80\%, 85\%, 90\%\}$

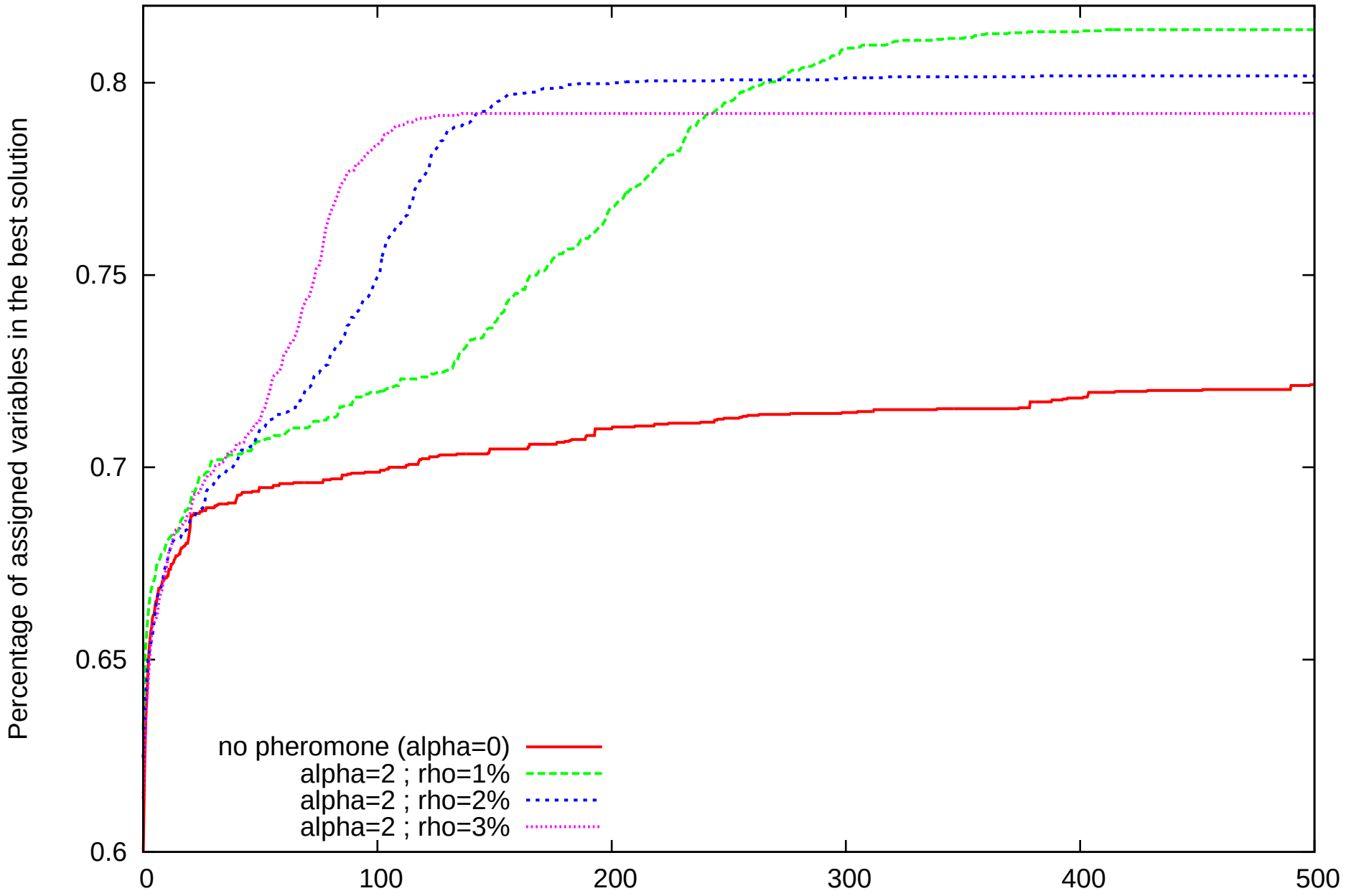
Table of contents

Parameter setting

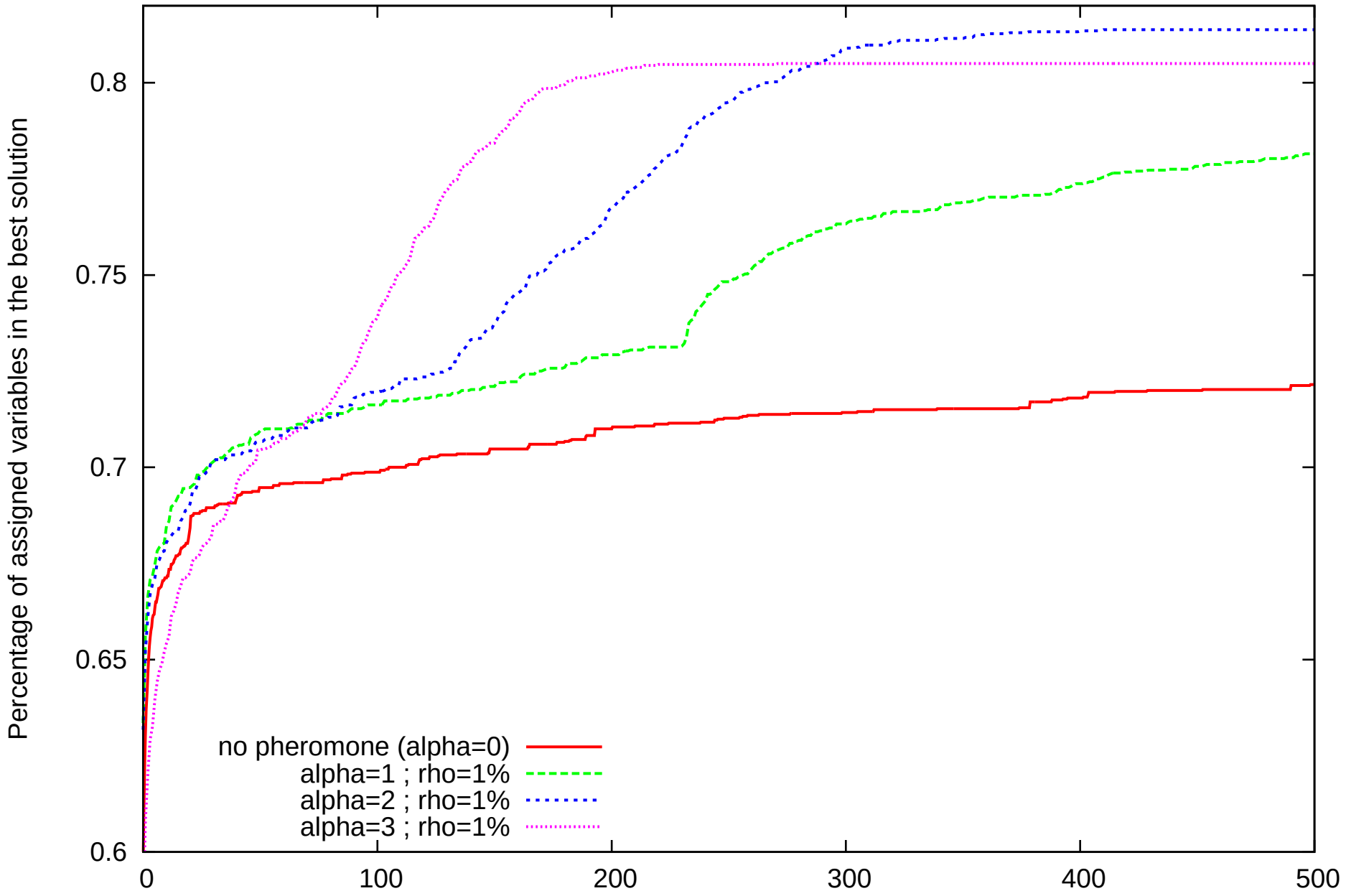
- Maximum number of cycles = 2500
- Number of ants = 30
- Heuristic factor weight β
  - When no heuristic is used, $\beta = 0$
  - When the heuristic based on utilisation rates is used, $\beta = 6$
- Pheromone bounds: $\tau_{min} = 0.01$ and $\tau_{max} = 10$
- Pheromone factor weight $\alpha \in \{1,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

Pheromone parameters influence the solution process

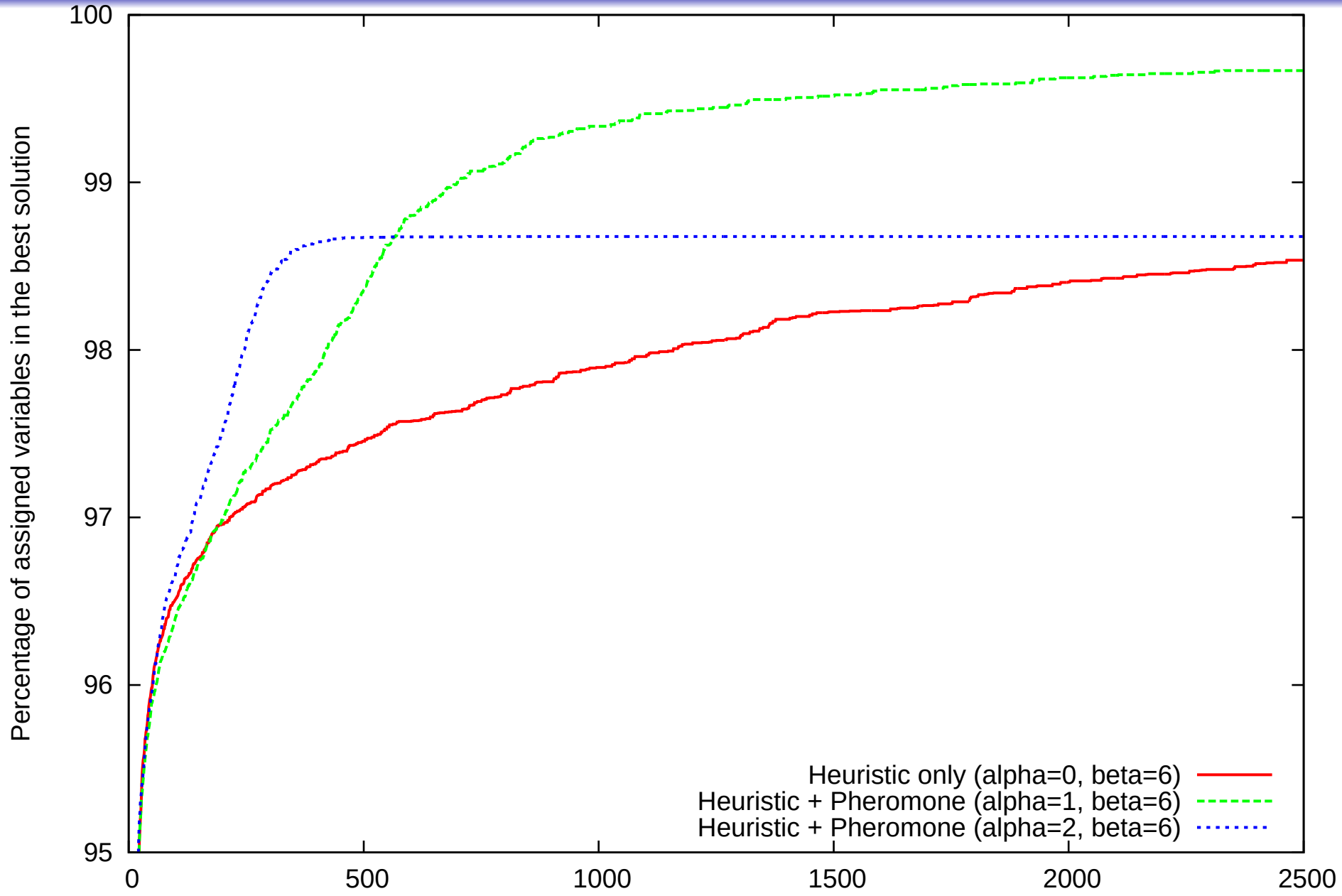
Influence of evaporation on the solution process (without heuristic)



Influence of α on the solution process (without heuristic)



Experimental results with the heuristic



Experimental results with the heuristic

Instance	Heuristic only ($\alpha = 0, \beta = 6$)			Heuristic + Pheromone ($\alpha = 1, \beta = 6$)		
	SR	#Cycles	Time	SR	#Cycles	Time
26-82	15%	1576	35.0	90%	679	18.9
16-81	45%	1030	33.0	72%	998	28.7
4-72	95%	706	15.9	100%	353	9.5
41-66	100%	11	0.2	100%	10	0.2
60-*	100%	1	0.1	100%	1	0.1
65-*	100%	1	0.1	100%	1	0.1
70-*	100%	2	0.1	100%	2	0.1
75-*	100%	2	0.1	100%	2	0.1
80-*	100%	2	0.1	100%	2	0.1
85-*	100%	3	0.1	100%	3	0.1
90-*	100%	3	0.2	100%	3	0.2

Table of contents

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

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 !

Influence of α and ρ on the resampling ratio

Number of cycles	250	500	1000	1500	2000
$\alpha = 1, \rho = 1\%$	0.000	0.000	0.000	0.000	0.000
$\alpha = 2, \rho = 1\%$	0.000	0.000	0.001	0.006	0.009
$\alpha = 2, \rho = 2\%$	0.028	0.453	0.699	0.780	0.821

Influence of α and ρ on the similarity ratio

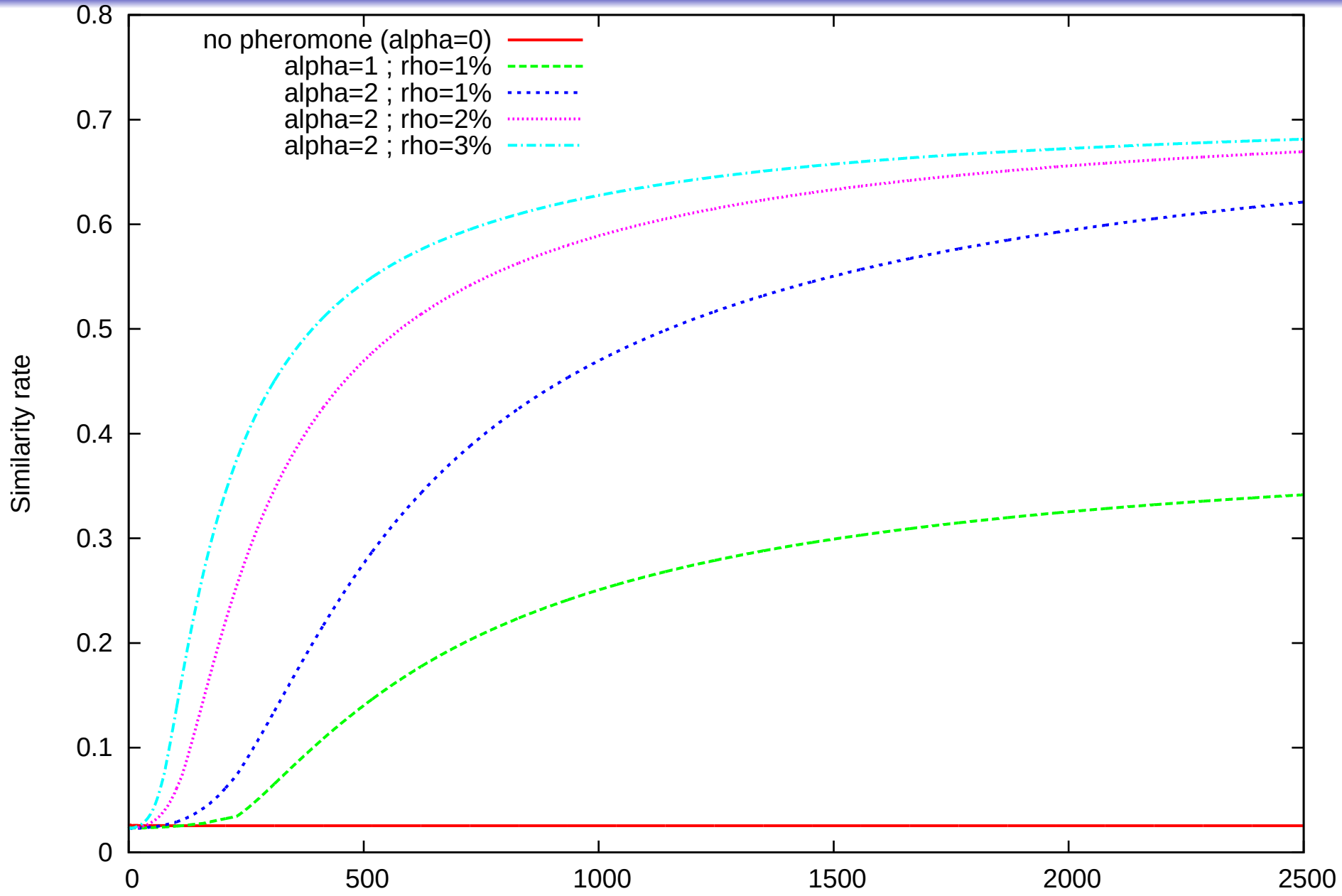


Table of contents

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 $\rightsquigarrow$ towards reactive ACO
  - use resampling and similarity ratio to dynamically tune parameters
- Learning from failures