Using ACO to guide a CP search

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

[1] ILOG SA, 9 rue de Verdun, 94253 Gentilly cedex, France
{mkhichane,palbert}@ilog.fr
[2] LIRIS CNRS UMR 5205, University of Lyon I
Nautibus, 43 Bd du 11 novembre, 69622 Villeurbanne cedex, France
christine.solnon@liris.cnrs.fr

Abstract. Despite its doubtless success, Constraint Programming (CP) has to face two main challenges to develop its success: solving harder and harder problems, and being accessible to users with little to no CP expertise. To address these challenges, we are exploring the use of Ant Colony Optimization (ACO) to solve problems expressed in a regular CP Solver: ILOG Solver. Basically, ants iteratively build partial solutions, trying to assign as many variables as possible, until a complete solution is found. Pheromone trails are used to guide the construction process with respect to past experiments in an adaptive way. So far the approach has been tested successfully on the car sequencing problem.

1 Introduction

Motivation

Constraint Programming (CP) is great to express optimization problems in a declarative way. However, as the problems tend to be harder and harder, the historical propagate-and-backtrack approach tends to reach its limits. To deal with this problem, most CP solvers give access to the search procedure: a user can program the search, thus specifying problem specific search methods and heuristics for exploiting problem properties to guide the solver toward the best solution. However, programming the search requires a lot of background knowledge and optimization expertise. This is a tough brake on the expansion of CP in business software.

Stochastic methods have proved their adequacy to solve efficiently complex problems. However most works have focused on the optimization algorithms leaving aside the ease of description of the problem itself, so that most stochastic methods rely on a lot of procedural programming.

Our approach is thus to explore the tight integration of CP with some stochastic search. If we reach the goal, the user will focus on modeling the problem using a high-level declarative language and let the engine search for the (best) solution. We acknowledge that the approach might not give always the best results, but it will allow more people, thus more applications, to benefit from the optimization power that CP can deliver.

Ant Colony Optimization

We have chosen the Ant Colony Optimization (ACO) meta-heuristic for guiding the search process. A main reason for this choice is that ACO is a constructive approach which may be integrated with constraint propagation in a very straightforward way.

The basic idea of ACO [1–3] is to model the problem to solve as the search for a best path in a graph, and to use artificial ants to search for good paths. The behavior of artificial ants mimics the behavior of real ants: artificial ants lay pheromone on components (edges and/or vertices) of the graph and each ant chooses its path with respect to probabilities that depend on pheromone trails that have been previously laid by the colony; these pheromone trails progressively decrease by evaporation. Intuitively, this indirect stigmergic communication means aims at giving information about the quality of path components in order to attract ants, in the following iterations, towards the corresponding areas of the search space.

Artificial ants also have some extra-features that do not find their counterpart in real ants. In particular, they are usually associated with data structures that contain the memory of their previous actions, and they may apply some daemon procedures, such as local search, to improve the quality of computed paths. In many cases, pheromone is updated only after having constructed a complete path, and not during the walk, and the amount of pheromone deposited is usually a function of the quality of the complete path. Finally, the probability for an artificial ant to choose a component often depends not only on pheromones, but also on some problem-specific local heuristics.

The first ant algorithm to be applied to a discrete optimization problem has been proposed by Dorigo in [4]. The problem chosen for the first experiments was the Traveling Salesman Problem and, since then, this problem has been widely used to investigate the solving capabilities of ants [5, 6]. The ACO meta-heuristic, described in [3], is a generalization of these first ant based algorithms, and has been successfully applied to different hard combinatorial optimization problems such as quadratic assignment problems [7], vehicle routing problems [8], or maximum clique problems [9].

Basic principles of our approach and outline of the paper

In a first step, we focus on pure Constraint Satisfaction Problems (CSPs), the goal of which is to find an assignment to a number of constrained variables. We want to deal with optimization problems —CSPs with a minimization or maximization objective function— in a next step.

Our research is based upon ILOG Solver, and we use its modeling language and its propagation engine, but the search is guided by ACO. This approach has the benefit of reusing all the work done at the modeling level as well as the code dedicated to constraint propagation and verification. We can as well test different variations of our ideas on a large benchmark library —this is an invaluable asset.

Some ACO algorithms have been previously proposed for solving Constraint Satisfaction Problems: [10, 11] describe ACO algorithms for solving binary CSPs; [12] describes ACO algorithms for solving the car sequencing problem. In these algorithms, ants iteratively build complete assignments (that assign a value to every variable) that may violate constraints, and their goal is to minimize the number of constraint violations; a solution is found when the number of constraint violations is null.

In this paper, we investigate a new ACO framework for solving CSPs: ants iteratively build partial assignments (such that some variables may not be assigned to a value) that do not violate constraints, and their goal is to maximize the number of assigned variables; a solution is found when all variables are assigned. This new ACO framework for solving CSP may be combined with the propagation engine of ILOG Solver in a very straightforward way.

The paper is organized as follows. In the next section, we briefly recall some definitions and terminology about CSPs. Section 3 describes the basic ACO algorithm for solving CSPs. Section 4 shows how this algorithm may be used to solve a classical CP benchmark, i.e., the car sequencing problem, and Section 5 gives some preliminary experimental results on this problem.

2 Background

A *CSP* [13] is defined by a triple (X, D, C) such that X is a finite set of variables, D is a function that maps every variable $X_i \in X$ to its domain $D(X_i)$, that is, the finite set of values that can be assigned to X_i, and C is a set of constraints, that is, relations between some variables which restrict the set of values that can be assigned simultaneously to these variables.

An *assignment* is a set of variable-value pairs, noted $\langle X_i, v_i \rangle$ and corresponding to the assignment of a value $v_i \in D(X_i)$ to a variable X_i. The variables assigned in an assignment $\mathcal{A}$ are denoted by $var(\mathcal{A})$. An assignment $\mathcal{A}$ is *partial* if some variables are not assigned in $\mathcal{A}$, i.e., $var(\mathcal{A}) \subset X$; it is *complete* if all variables are assigned, i.e., $var(\mathcal{A}) = X$. An assignment $\mathcal{A}$ is *consistent* if it does not violate any constraint. A *solution of a CSP* (X, D, C) is a complete and consistent assignment.

3 Description of Ant-CP

The proposed algorithm for solving CSPs, called Ant-CP, follows the *MAX-MIN* Ant System algorithmic scheme [14], and is sketched in Algorithm 1. First, pheromone trails are initialized to some given value τ_{max}. Then, at each cycle (lines 2-12), each ant k constructs a consistent assignment $\mathcal{A}_k$ (lines 4-10): starting from an empty assignment, the ant iteratively chooses a variable which is not yet assigned and a value to assign to this variable; this variable assignment is added to $\mathcal{A}_k$, and constraints are triggered which might in turn narrow the domains of non assigned variables, trigger new assignments, or detect a failure;

Algorithm 1: Ant-CP procedure

Input: A CSP (X, D, C) and a set of parameters $\{\alpha, \beta, \rho, \tau_{min}, \tau_{max}, nbAnts\}$
Output: A consistent assignment for (X, D, C)

```
1  Initialize all pheromone trails to τ_max
2  repeat
3      foreach k in 1..nbAnts do
           /* Construction of a consistent assignment A_k                    */
4          A_k ← ∅
5          repeat
6              Select a variable X_j ∈ X so that X_j ∉ var(A_k)
7              Choose a value v∈D(X_j) with probability p(X_j, v)
8              A_k ← A_k ∪ {⟨X_j, v⟩}
9              trigger((X, D, C), ⟨X_j, v⟩, A_k, Failure)
10         until var(A_k) = X or Failure ;
11     Update pheromone trails using {A_1, ..., A_nbAnts}
12 until var(A_i) = X for some i ∈ {1..nbAnts} or max cycles reached ;
13 return the largest constructed assignment
```

this process is iterated until either all variables have been assigned (i.e., a solution has been found) or the propagation step detects a failure. Once every ant has constructed an assignment, pheromone trails are updated (line 11). The algorithm stops iterating either when an ant has found a solution, or when a maximum number of cycles has been performed.

In the next paragraphs, we define the pheromone structure on which artificial ants lay pheromone trails, and we describe the variable selection step, the transition probability function used for choosing values, the propagation step and the pheromone updating step.

Pheromone structure. Pheromone trails are used to guide ants during their assignment constructions: the goal is to learn from previous experiments which variable-value assignments have allowed ants to build good assignments, and to use this information to bias further assignment constructions.

Hence we associate a pheromone component with every variable-value couple $\langle X_i, v_i \rangle$ such that $X_i \in X$ and $v_i \in D(X_i)$. The amount of pheromone associated with a couple $\langle X_i, v_i \rangle$ is noted $\tau_{\langle X_i, v_i \rangle}$. Intuitively, this amount of pheromone represents the learnt desirability of assigning value v_i to variable X_i.

Selection of a variable. When constructing an assignment, the order in which the variables are assigned is rather important and variable ordering heuristics have been studied widely [13, 15]. Some commonly used variable-orderings are, e.g., *most-constraining-first* ordering, which selects an unassigned variable that is connected (by a constraint) to the largest number of unassigned variables; *most-constrained-first* ordering, which selects an unassigned variable that is connected (by a constraint) to the largest number of assigned variables; or *smallest-domain*

ordering, which selects an unassigned variable that has the smallest number of consistent values with respect to the partial assignment already built.

To solve a CSP with Ant-CP, one has to choose a variable selection procedure which, given a partial assignment, returns the next variable to be assigned. In experiments reported in this paper, we have considered the smallest-domain ordering (further ties are broken randomly).

Choice of a value. Once a variable has been selected, ants have to choose a value for it. The goal is to choose the most promising value for the variable, that is, the one that will allow the ant to assign the largest number of variables. However, it is difficult to anticipate the consequences of the choice of a value on the further assignments of the remaining unassigned variables. Hence, there are very few heuristics for guiding this choice and these heuristics usually are problem-dependent, so they cannot be applied to general CSPs (e.g., [15] for the car-sequencing problem).

The main contribution of ACO for solving CSPs is to provide a generic approach for choosing values: this choice is made randomly with a probability that depends on a pheromone factor which evaluates the learnt desirability of the value, and a heuristic factor which is problem-dependent. More formally, the probability for an ant to select a value v for a variable X_j is defined by

$$p(X_j, v) = \frac{[\tau_{\langle X_j, v\rangle}]^\alpha [\eta(X_j, v)]^\beta}{\sum_{w \in D(X_j)} [\tau_{\langle X_j, w\rangle}]^\alpha [\eta(X_j, w)]^\beta}$$

where $\tau_{\langle X_j, v\rangle}$ is the pheromone trail associated with the couple $\langle X_j, v\rangle$, $\eta(X_j, v)$ is a problem-dependent heuristic, and α and β are two parameters that determine the relative weights of pheromone and heuristic information.

Constraint propagation. Each time a variable is assigned to a value, a propagation algorithm is called. This algorithm narrows the domains of the variables that are not yet assigned. If the domain of a variable becomes a singleton, then the partial assignment $\mathcal{A}_k$ is completed by the assignment of this variable and the propagation process is continued. At the end of the propagation process, if the domain of a variable becomes empty or if some inconsistency is detected, then *Failure* is assigned to *True*; otherwise, it is assigned to *False*.

One may consider different propagation algorithms, that ensure different partial consistencies, e.g., node-consistency, arc-consistency, or bound-arc-consistency. In our premiminary experiments, we have used the default propagation algorithm integrated to ILOG solver.

Pheromone updating step. Once every ant has constructed an assignment, pheromone trails are updated according to ACO: all pheromone trails are decreased uniformly, in order to simulate evaporation and allow ants to forget bad

assignments, and then the best ants of the cycle deposit pheromone. More formally, at the end of each cycle, the quantity of pheromone on each couple $\langle X_i, v_i \rangle$ is updated as follows:

$$\tau_{\langle X_i,v_i \rangle} \leftarrow (1-\rho) \cdot \tau_{\langle X_i,v_i \rangle} + \sum_{\mathcal{A}_k \in BestOfCycle} \Delta\tau(\mathcal{A}_k, X_i, v_i)$$
$$\text{if } \tau_{\langle X_i,v_i \rangle} < \tau_{min} \text{ then } \tau_{\langle X_i,v_i \rangle} \leftarrow \tau_{min}$$
$$\text{if } \tau_{\langle X_i,v_i \rangle} > \tau_{max} \text{ then } \tau_{\langle X_i,v_i \rangle} \leftarrow \tau_{max}$$

where

- ρ is the evaporation parameter, such that $0 \leq \rho \leq 1$,
- τ_{min} and τ_{max} are two parameters for bounding pheromone trails,
- *BestOfCycle* is the set of the best assignments constructed during the cycle, that is, $BestOfCycle = \{\mathcal{A}_i \in \{\mathcal{A}_1, \ldots, \mathcal{A}_{nbAnts}\} \mid |var(\mathcal{A}_i)| \text{ is maximal}\}$
- $\Delta\tau(\mathcal{A}_k, X_i, v_i)$ is the quantity of pheromone deposited on the couple $\langle X_i, v_i \rangle$ by the ant that has built assignment $\mathcal{A}_k$. This quantity is equal to zero if X_i is not assigned to v_i in $\mathcal{A}_k$; otherwise, it is proportionally inverse to the gap of sizes between $\mathcal{A}_k$ and the largest assignment $\mathcal{A}_{best}$ built since the beginning of the search (including the current cycle), i.e.,

$$\Delta\tau(\mathcal{A}_k, X_i, v_i) = 0 \qquad \text{if } \langle X_i, v_i \rangle \notin \mathcal{A}_k$$
$$\Delta\tau(\mathcal{A}_k, X_i, v_i) = 1/(1 + |\mathcal{A}_{best}| - |\mathcal{A}_k|) \quad \text{if } \langle X_i, v_i \rangle \in \mathcal{A}_k$$

4 Using Ant-CP to solve the Car Sequencing Problem

Problem description. The car sequencing problem involves scheduling cars along an assembly line in order to install options (e.g., sun-roof or air-conditioning) on them. Each option is installed by a different station, designed to handle at most a certain percentage of the cars passing along the assembly line, and the cars requiring this option must be spaced so that the capacity of the station is never exceeded.

This problem is NP-hard [16]. It has been formulated as a constraint satisfaction problem (CSP), and is a classical benchmark for CP solvers [17]. Dedicated filtering algorithms have been proposed for handling the sequence constraint [18, 19]. These filtering algorithms, when integrated within a branch and propagate exhaustive search, are very effective to solve some highly constrained feasible instances, or to prove infeasibility of some over-constrained instances. However, on some other instances, it cannot reduce domains enough to make complete search tractable.

Hence, different incomplete approaches have been proposed, that leave out exhaustivity, trying to quickly find approximately optimal solutions in an opportunistic way, e.g., local search [20], large neighbourhood search [21], IDWalk [22], or Ant Colony Optimization [23].

A more general problem –which introduces paint batching constraints and priority levels for capacity constraints– has been proposed by Renault for the ROADEF challenge in 2005 [24].

Test suite. We have considered the four feasible instances used in [18] and available in the benchmark library CSPLib [25], i.e., `16-81`, `26-82`, `41-66`, and `6-76`. These instances have 100 cars to sequence and 5 options.

CP model. Two different CP models have been proposed to model the car sequencing problem with ILOG solver [26]. In experiments reported here, we have considered the second model which associates a variable with each car to be sequenced and uses a global *distribute* constraint for constraining the number of cars required by each configuration of options, and a global *sequence* constraint [18] to define constraints on sequences of cars.

Then, the only thing to do is to define the heuristic function η that must be used for this problem. Experiments reported in section 5 have been done with no heuristic, i.e., when defining $\eta(X_j, v) = 1$. Our goal is to explore the benefit of using ACO to guide a CP search, and to quantify the influence of pheromone parameters on the solution process. The known heuristics for the car sequencing problem are quite efficient and would seriously bias our study. Indeed, we have introduced in [20] different heuristic functions dedicated to the car sequencing problem. These heuristics are mainly based on option utilization rates and aim at favoring the choice of cars requiring options that have high demands with respect to capacities [15]. We have shown in [20] that a simple randomized greedy algorithm –that randomly selects cars with respect to probabilities that only depend on utilization rate heuristics– is able to solve the four instances of our test suite.

5 Experimental results

When solving a combinatorial optimization problem with a (meta-)heuristic approach such as local search, evolutionary computation, or ACO, one usually has to find a compromise between two dual goals. On the one hand, one has to *intensify* the search around the most promising areas, that are usually close to the best solutions found so far. On the other hand, one has to *diversify* the search and favor exploration in order to discover new, and hopefully more successful, areas of the search space. The behavior of ants with respect to this intensification/diversification duality may be influenced by modifying parameter values.

In this section, we first discuss the role of the different parameters with respect to this intensification/diversification duality; then we introduce in section 5.2 two measures for quantifying intensification/diversification.

5.1 Influence of ACO parameters on the solution process

Influence of τ_{min} and τ_{max}. As pointed out in [14], the goal of bounding pheromone trails within an interval $[\tau_{min}, \tau_{max}]$ is to avoid premature stagnation of search, i.e., a situation where all ants construct the same solution over and over again so that no better solutions can be found anymore. Indeed, by imposing explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone trails,

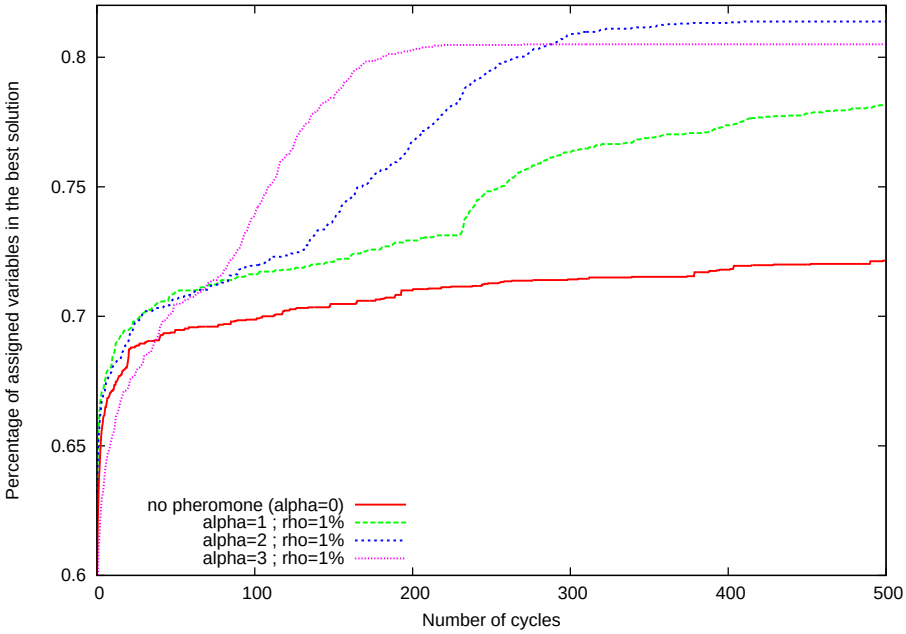

Fig. 1. Influence of pheromone on the solution process of `Ant-CP`: each curve plots the evolution of the number of assigned variables in the best constructed assignment when the number of cycles increases, for a given setting of α (average on 10 runs on four instances of the car sequencing problem). The other parameters have been set to $\beta = 0$, $\rho = 1\%$, $nbAnts = 30$, $\tau_{min} = 0.01$, and $\tau_{max} = 10$.

one ensures that relative differences between pheromone trails cannot become too extreme. Therefore, the probability of choosing a value cannot become too small and stagnation situations are avoided. Furthermore, by initializing pheromone trails to τ_{max} at the beginning of the search, one ensures that during early cycles the relative difference between pheromone trails is rather small (after i cycles, it is bounded by a ratio of $(1 - \rho)^i$). Hence, exploration is emphasized at the beginning of the search.

In experiments reported here, we have set τ_{min} to 0.01 and τ_{max} to 10.

Influence of α and ρ. The two pheromone parameters α and ρ have a great influence on the solution process. Indeed, diversification can be emphasized either by decreasing the value of the pheromone factor weight α —so that ants become less sensitive to pheromone trails— or by decreasing the value of the pheromone evaporation rate ρ. When increasing the exploratory ability of ants in this way, one usually finds better solutions, but as a counterpart it takes longer time to find them. This is illustrated in Figures 1 and 2 on the car sequencing problem.

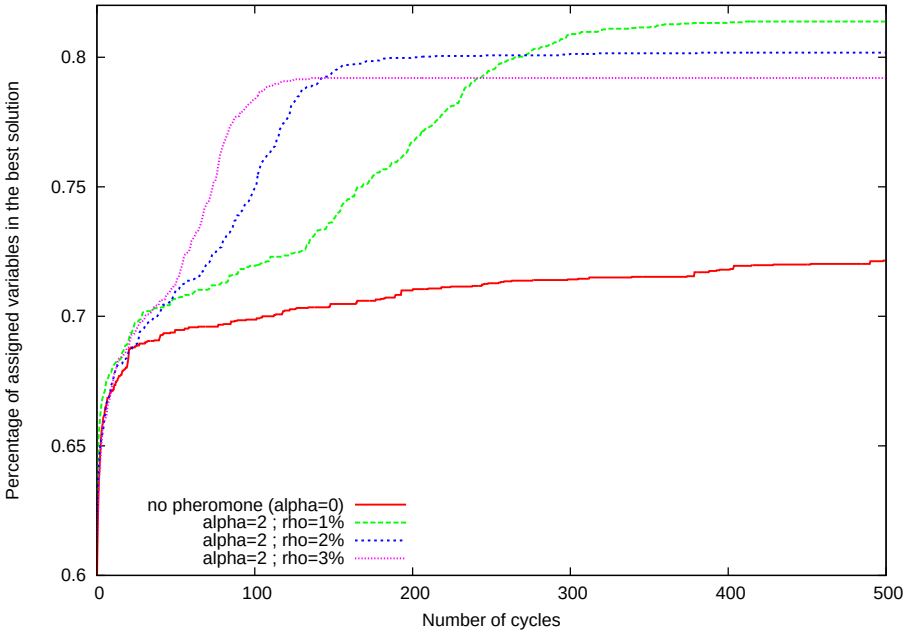

Fig. 2. Influence of pheromone on the solution process of `Ant-CP`: each curve plots the evolution of the number of assigned variables in the best constructed assignment when the number of cycles increases, for a given setting of ρ (average on 10 runs on four instances of the car sequencing problem). The other parameters have been set to $\alpha = 2$, $\beta = 0$, $nbAnts = 30$, $\tau_{min} = 0.01$, and $\tau_{max} = 10$.

Note that results displayed in these Figures have been obtained with $\beta = 0$ so that values are selected with respect to the pheromone factor only.

On these figures, one can first note that when $\alpha = 0$, the best constructed assignments are much smaller: in this case, pheromone is totally ignored and the resulting search process performs as a random one so that after 500 cycles, the size of the best assignment nearly stops increasing, and hardly reaches 72% of the variables. This shows that pheromone actually improves the solution process with respect to a pure random algorithm.

Figures 1 and 2 also show us that increasing α or ρ implies a quicker feedback so that ants find better solutions at the beginning of the solution process; however, this also implies quicker stagnation so that ants have converged toward worse solutions at the end of the solution process. Hence, the setting of α and ρ let us balance between two main tendencies. On the one hand, when limiting the influence of pheromone with low pheromone factor and evaporation rates, the quality of the final solution is better, but the time needed to converge on this solution is also higher. On the other hand, when increasing the influence of pheromone with higher pheromone factor and evaporation rates, ants find better

solutions during the first cycles, but after a few hundred or so cycles, they are no longer able to find better solutions.

On this test suite of the car sequencing problem, the best results are obtained when setting α to 2 and ρ to 1%.

Influence of *nbAnts*. To emphasize diversification and avoid premature stagnation, one can also increase the number of ants so that more states are explored at each cycle.

Influence of β. This parameter determines the sensitivity of ants to the heuristic factor in the state transition rule. This heuristic factor is problem-dependant. Its relevancy depends on the considered problem, and therefore the setting of β is different from one problem to another.

5.2 Measuring intensification/diversification

To measure the diversification effort of an ACO search process, we propose to use the re-sampling and the diversification ratio.

Re-sampling ratio. This measure has been used, e.g., in [27, 9], in order to get insight into how effective algorithms are in sampling the search space: if we define *nbDiff* as the number of unique candidate solutions generated by an algorithm over a whole run and *nbTot* as the total number of generated candidate solutions, then the re-sampling ratio is defined as (*nbTot* - *nbDiff*)/*nbTot*. Values close to 0 correspond to an effective search, i.e., not many duplicate candidate solutions are generated, whereas values close to 1 indicate a stagnation of the search process around a small set of solutions.

Note that the re-sampling ratio may be very quickly computed by using a hashing table and counting the number of collisions. To reduce time and space complexities, we do not store assignments in the hashing table, but a second hashing key. In this case, we obtain an approximation of the re-sampling ratio as it may happen that two different assignments share the same hashing keys. However, the probability that this happens is equal to $n/(p1 \cdot p2)$ where n is the number of different computed assignments, and $p1$ and $p2$ are the maximal values of the two hashing keys (provided that hashing keys are uniformely distributed). Hence, by choosing large enough values for $p1$ and $p2$ we compute a very good approximation of the re-sampling ratio.

Table 1 provides an insight into `Ant-CP`'s performance by means of this re-sampling ratio. This table shows us that the search is very well diversified when $\alpha = 1$ and $\rho = 1\%$, whereas it is not much diversified when $\alpha = 2$ and $\rho = 2\%$.

The re-sampling ratio allows one to quantify the size of the searched space. However, it gives no information about the distribution of the computed solutions within the whole search space. Actually, a pure random search (nearly) never re-samples twice a same solution but, as it does not intensify the search around promising solutions, it usually is not able to find good solutions.

Table 1. Evolution of the re-sampling ratio of `Ant-CP`. Each row successively displays the setting of α and ρ, and the re-sampling ratio after 250, 500, 1000, 1500, and 2000 cycles (average over 10 runs for four instances). The other parameters have been set to $\beta = 0$, $nbAnts = 10$, $\tau_{min} = 0.01$, $\tau_{max} = 10$.

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

Similarity ratio. The similarity ratio provides a complementary insight into algorithm performance by indicating how much the computed solutions are similar, i.e., how much the search is intensified.

The similarity ratio corresponds to the pair-wise population diversity measure, introduced for genetic approaches, e.g., in [28]. More precisely, we define the similarity of two assignments $\mathcal{A}_i$ and $\mathcal{A}_j$ by

$$sim(\mathcal{A}_i, \mathcal{A}_j) = \frac{2 \cdot |\mathcal{A}_i \cap \mathcal{A}_j|}{|\mathcal{A}_i| + |\mathcal{A}_j|}$$

and we define the similarity ratio of a set of assignments S by the average similarity of every pair of different assignments in S.

Figure 3 plots the evolution of the similarity ratio of the assignments computed every cycle, thus giving an information about the distribution of these assignments. For example, let us consider the curve plotting the evolution of the similarity ratio when α is set to 2 and ρ to 1%. The similarity increases from less than 5% at the beginning of the solution process to 28% at cycle 500. This shows that ants progressively focus on a sub-region of the search space. When considering this together with the fact that the re-sampling ratio is very low (smaller than 1%), one can conclude that in this case Ant-CP reaches a good compromise between diversification —as it nearly never re-computes twice a same assignment— and intensification —as the similarity of the computed assignments is increased.

Figure 3 also shows that, when α or ρ increase, the similarity ratio both increases sooner and rises more steeply.

Note that the similarity ratio may be computed very quickly by maintaining an array `freq` such that, for every variable-value couple $\langle X_i, v_i \rangle$, `freq[`$\langle X_i, v_i \rangle$`]` is equal to the number of assignments of S which contain $\langle X_i, v_i \rangle$. In this case, the similarity ratio of S is equal to

$$\frac{\sum_{X_i \in X, v_i \in D(X_i)} (\texttt{freq}[\langle \texttt{X}_\texttt{i}, \texttt{v}_\texttt{i} \rangle] \cdot (\texttt{freq}[\langle \texttt{X}_\texttt{i}, \texttt{v}_\texttt{i} \rangle] - 1))}{(\mid S \mid -1) \cdot \sum_{\mathcal{A}_k \in S} \mid \mathcal{A}_k \mid}$$

and it can be easily computed in an incremental way while constructing assignments (see [28] for more details).

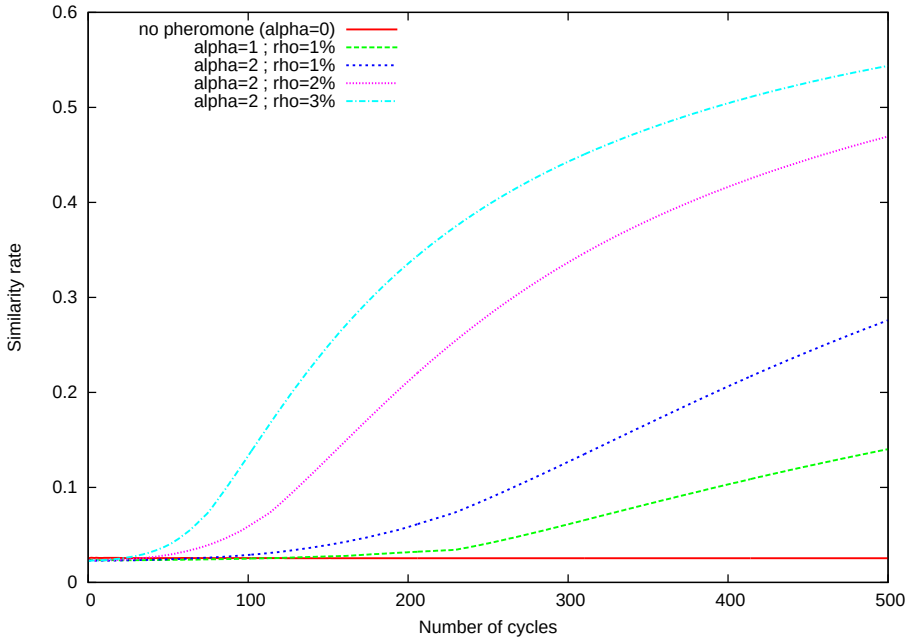

Fig. 3. Evolution of the similarity ratio for different settings of α and ρ (average on 10 runs on four different instances of the car sequencing problem). The other parameters have been set to $\beta = 0$, $nbAnts = 30$, $\tau_{min} = 0.01$, and $\tau_{max} = 10$.

6 Conclusion

This paper reports a preliminary work on guiding a CP search with the ACO meta-heuristic. The problem to solve is defined in a declarative way by means of constraints, thanks to ILOG solver's modelling language. The ILOG solver's search process is substituted by an ACO search process which consists in iteratively constructing consistent partial assignments until a solution is found. During assignment constructions, pheromone is used to dynamically adapt the value selection procedure while the constraint propagation engine of ILOG solver is used to filter variable domains.

This work may be related to [29] that uses evolutionary algorithms to generate robust CP search strategies, and to COMET [30] which is a high level language for modelling problems by means of constraints, and solving them with local search

First experiments on the car sequencing problem have demonstrated the feasibility of this framework, eventhough results are not competitive with state-of-the-art approaches dedicated to the car sequencing problem. However, one should notice that these results have been obtained without integrating any problem-dependent heuristic (that are known to be very efficient on this prob-

lem). Indeed, our goal is not (yet) to deliver the best results, but rather (i) to evaluate the expected complementarity of CP and ACO, (ii) to understand and to model the effect of the pheromones on the search process and (iii) to provide an insight into the search process by means of intensification/diversification measures.

The two intensification/diversification measures –re-sampling and similarity ratio– provide a simple synthetic information on the search process. We are thus starting working on exploring the design of simple methods that will use these two measures to control intensification/diversification in an adaptive way. A tiny supervisor embedded in the ACO implementation maintains a model of the search process and adequately adapts the α and ρ parameters in order to focus more rapidly on the exploitation of the interesting zones of the search space, while being able to escape gracefully from local optima that seems to have been enough explored.

To get the full power of ACO, one needs to equip it with an heuristic. Thus our second direction is the design of a method that progessively learns heuristic values from the search of each ant. First experiments on the car sequencing problem tend to show that very fruitful information can be learnt from the failures detected by the CP propagator. Such learnt information is very complementary to the pheromones that are learnt from partial success of ants.

References

1. Dorigo, M., Di Caro, G.: The Ant Colony Optimization meta-heuristic. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw Hill, UK (1999) 11–32
2. Dorigo, M., Caro, G.D., Gambardella, L.: Ant algorithms for discrete optimization. Artificial Life **5**(2) (1999) 137–172
3. Dorigo, M., Stuetzle, T.: Ant Colony Optimization. MIT Press (2004)
4. Dorigo, M.: Optimization, Learning and Natural Algorithms *(in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (1992)
5. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics **26**(1) (1996) 29–41
6. Dorigo, M., Gambardella, L.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 53–66
7. Gambardella, L., Taillard, E., Dorigo, M.: Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society **50** (1999) 167–176
8. Bullnheimer, B., Hartl, R., Strauss, C.: An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research **89** (1999) 319–328
9. Solnon, C., Fenet, S.: A study of aco capabilities for solving the maximum clique problem. Journal of Heuristics **12**(3) (2006) 155–180
10. Solnon, C.: Ants can solve constraint satisfaction problems. IEEE Transactions on Evolutionary Computation **6**(4) (2002) 347–357
11. Solnon, C., Bridge, D.: An ant colony optimization meta-heuristic for subset selection problems. In Nedjah, N., Mourelle, L., eds.: System Engineering using Particle Swarm Optimization, Nova Science (2006) 7–29

12. Solnon, C.: Combining two ant colony optimization algorithms for solving the car sequencing problem. to appear in European Journal of Operational Research (EJOR) (2007)
13. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London, UK (1993)
14. Stützle, T., Hoos, H.: MAX-MIN Ant System. Journal of Future Generation Computer Systems, special issue on Ant Algorithms **16** (2000) 889–914
15. Smith, B.: Succeed-first or fail-first: A case study in variable and value ordering heuristics. In: third Conference on the Practical Applications of Constraint Technology PACT'97. (1996) 321–330
16. Kis, T.: On the complexity of the car sequencing problem. Operations Research Letters **32** (2004) 331–335
17. Dincbas, M., Simonis, H., van Hentenryck, P.: Solving the car-sequencing problem in constraint logic programming. In Kodratoff, Y., ed.: Proceedings of ECAI-88. (1988) 290–295
18. Regin, J.C., Puget, J.F.: A filtering algorithm for global sequencing constraints. In: CP97. Volume 1330 of LNCS. Springer-Verlag (1997) 32–46
19. van Hoeve, W.J., Pesant, G., Rousseau, L.M., Sabharwal, A.: Revisiting the sequence constraint. In: 12th International Conference on Principles and Practice of Constraint Programming - CP 2006. Volume 4204 of Lecture Notes in Computer Science., Springer (2006) 620–634
20. Gottlieb, J., Puchta, M., Solnon, C.: A study of greedy, local search and ant colony optimization approaches for car sequencing problems. In: Applications of evolutionary computing. Volume 2611 of LNCS., Springer (2003) 246–257
21. Perron, L., Shaw, P.: Combining forces to solve the car sequencing problem. In: Proceedings of CP-AI-OR'2004. Volume 3011 of LNCS., Springer (2004) 225–239
22. Neveu, B., Trombettoni, G., Glover, F.: Id walk: A candidate list strategy with a simple diversification device. In: Proceedings of CP'2004. Volume 3258 of LNCS., Springer Verlag (2004) 423–437
23. Solnon, C.: Solving permutation constraint satisfaction problems with artificial ants. In: Proceedings of ECAI'2000, IOS Press, Amsterdam, The Netherlands. (2000) 118–122
24. Solnon, C., Cung, V., Nguyen, A., Artigues, C.: The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. to appear in European Journal of Operational Research (EJOR) (2007)
25. Gent, I., Walsh, T.: Csplib: a benchmark library for constraints. Technical report, APES-09-1999 (1999) available from http://csplib.cs.strath.ac.uk/. A shorter version appears in CP99.
26. ILOG: Ilog solver user's manual. Technical report, ILOG (1998)
27. van Hemert, J., Solnon, C.: A study into ant colony optimization, evolutionary computation and constraint programming on binary constraint satisfaction problems. In: Evolutionary Computation in Combinatorial Optimization (EvoCOP 2004). Volume 3004 of LNCS., Springer-Verlag (2004) 114–123
28. Morrison, R., Jong, K.D.: Measurement of population diversity. In: 5th International Conference EA 2001. Volume 2310 of LNCS., Springer-Verlag (2001) 31–41
29. Dumeur, R., Puget, J.F., Shaw:, P.: Applying evolutionary search to generate robust constraint programming search strategies. In: Workshop on Combination of metaheuristic and local search with Constraint Programming techniques. (2005)

30. Michel, L., Hentenryck, P.V.: A constraint-based architecture for local search. In: OOPSLA '02: Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, NY, USA, ACM Press (2002) 83–100