CP with ACO

Madjid Khichane, Patrick Albert, Christine Solnon

CP with ACO

Madjid Khichane[1,2], 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, France
christine.solnon@liris.cnrs.fr

The Ant Colony Optimization (ACO) meta-heuristic [1] has proven its efficiency to solve hard combinatorial optimization problems. However most works have focused on designing efficient ACO algorithms for solving specific problems, but not on integrating ACO within declarative languages so that solving a new problem with ACO usually implies a lot of procedural programming. Our approach is thus to explore the tight integration of Constraint Programming (CP) with ACO. 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.

1 Description of Ant-CP

Some ACO algorithms have been previously proposed for solving Constraint Satisfaction Problems (CSPs), e.g., [2, 3]. 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 may be combined with the propagation engine of ILOG Solver in a very straightforward way. It can be viewed as a generalization of [4] to CSPs.

More precisely, our approach 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 propagated; this process is iterated until either all variables have been assigned or the propagation step detects a failure. Once every ant has constructed an assignment, pheromone trails are updated. The algorithm stops iterating either when an ant has found a solution, or when a maximum number of cycles has been performed.

Pheromone structure. The pheromone structure, denoted by Φ, is a parameter which defines the set of pheromone trails that are used to guide ants

Algorithm 1: Ant-CP procedure

Input: A CSP (X, D, C), a pheromone structure Φ, and a heuristic factor η
Output: A (partial) consistent assignment for (X, D, C)

```
1  Initialize all pheromone trails of Φ to τ_max
2  repeat
3      foreach k in 1..nbAnts do
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)
8              Add ⟨X_j, v⟩ to A_k
9              Propagate constraints
10         until var(A_k) = X or Failure ;
11     Update pheromone trails of Φ 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
```

during assignment constructions. The default pheromone structure associates a pheromone trail with every variable-value couple, i.e., $\Phi_{default} = \{\tau_{\langle X_i, v_i \rangle} / X_i \in X, v_i \in D(X_i)\}$. Each pheromone trail $\tau_{\langle X_i, v_i \rangle}$ represents the learnt desirability of assigning value v_i to variable X_i. For specific problems, one may design other pheromone structures and we shall propose and compare two other pheromone structures for the car sequencing problem in the next section.

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 in the context of backtrach search. These heuristics can be used as well in our context of greedy construction of assignments.

Choice of a value. Once a variable X_j has been selected, the value v to be assigned to this variable is randomly chosen within $D(X_j)$ with respect to a probability $p(X_j, v)$ which depends on a pheromone factor $\tau_{\mathcal{A}}(X_j, v)$ —which reflects the past experience of the colony regarding the addition of $\langle X_j, v \rangle$ to the partial assignment $\mathcal{A}$— and a heuristic factor $\eta_{\mathcal{A}}(X_j, v)$ —which is problem-dependent, i.e.,

$$p(X_j, v) = \frac{[\tau_{\mathcal{A}}(X_j, v)]^{\alpha} [\eta_{\mathcal{A}}(X_j, v)]^{\beta}}{\sum_{w \in D(X_j)} [\tau_{\mathcal{A}}(X_j, w)]^{\alpha} [\eta_{\mathcal{A}}(X_j, w)]^{\beta}} \quad (1)$$

where α and β are two parameters that determine the relative weights of pheromone and heuristic information. The definition of the pheromone factor depends on the pheromone structure. For the default structure $\Phi_{default}$, this pheromone factor is defined by $\tau_{\mathcal{A}}(X_j, v) = \tau_{\langle X_j, v \rangle}$.

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 detected.

Pheromone updating step. Once every ant has constructed an assignment, each pheromone trail of the pheromone structure Φ is decreased and then the best ants of the cycle deposit pheromone, i.e., $\forall \tau_i \in \Phi$,

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

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,
- $\Delta\tau(\mathcal{A}_k, \tau_i)$ is the quantity of pheromone deposited on the pheromone trail τ_i by the ant that has built assignment $\mathcal{A}_k$. This quantity depends on the chosen pheromone structure. For the default structure $\Phi_{default}$, this quantity is equal to zero if X_i is not assigned to v 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).

2 Using Ant-CP to solve the Car Sequencing Problem

The car sequencing problem involves scheduling cars along an assembly line in order to install options on them. Each car requires a set of options; all cars requiring a same subset of options are grouped in a same car class. For each option i, a capacity constraint p_i/q_i imposes that there are at most p_i cars requiring option i every q_i consecutive cars. We refer the reader to [5] for more details on this problem.

CP model and variable ordering heuristic. The CP model for the car sequencing problem has been defined with the CP modeling language of ILOG Solver and corresponds to the first model proposed in the user's manual of ILOG Solver. In our experiments, we have used a classical sequential variable ordering heuristic, which consists in assigning variables associated with positions in the sequence of cars in the order defined by the sequence.

Pheromone structures for the car sequencing problem. As pheromone is at the core of the efficiency of any ACO implementation, we explore the impact of its structure: besides the default pheromone structure $\Phi_{default}$, we propose and compare two structures called $\Phi_{classes}$ and Φ_{cars}. To run Ant-CP with a new pheromone structure, one basically has to define the set Φ of pheromone components, define the pheromone factor $\tau_{\mathcal{A}}(X_j, v)$ with respect to these pheromone

components, and define which pheromone components must be rewarded during the pheromone updating step. Due to lack of space, we do not describe into details the two pheromone structures $\Phi_{classes}$ and Φ_{cars}, but just give the intuition:

- $\Phi_{classes}$ is used in [6] and associates a pheromone trail $\tau_{(v,w)}$ with every couple of car classes (v, w); this pheromone trail represents the learnt desirability of sequencing a car of class w just after a car of class v;
- Φ_{cars} is used in [3] and associates a pheromone trail $\tau_{(v,i,w,j)}$ with every couple of classes (v, w) and every $i \in [1; \#v]$ and every $j \in [1; \#w]$ where $\#v$ and $\#w$ respectively are the number of cars in classes v and w. This pheromone trail represents the learnt desirability of sequencing the j^{th} car of class w just after the i^{th} car of class v.

Heuristic factors for the car sequencing problem. In the transition probability defined by eq. (1), the pheromone factor is combined with a heuristic factor $\eta_{\mathcal{A}}(X_j, v)$ which is problem-dependent. We have considered here the DSU heuristic of [7], which is based on the sum of the dynamic utilization rates of options: $\eta_{\mathcal{A}}(X_j, v) = \sum_{i \in \text{ options}(v)} \frac{n_i \cdot q_i}{N \cdot p_i}$, where n_i is the number of cars that are not yet sequenced and that require option i and N is the number of cars that are not yet sequenced.

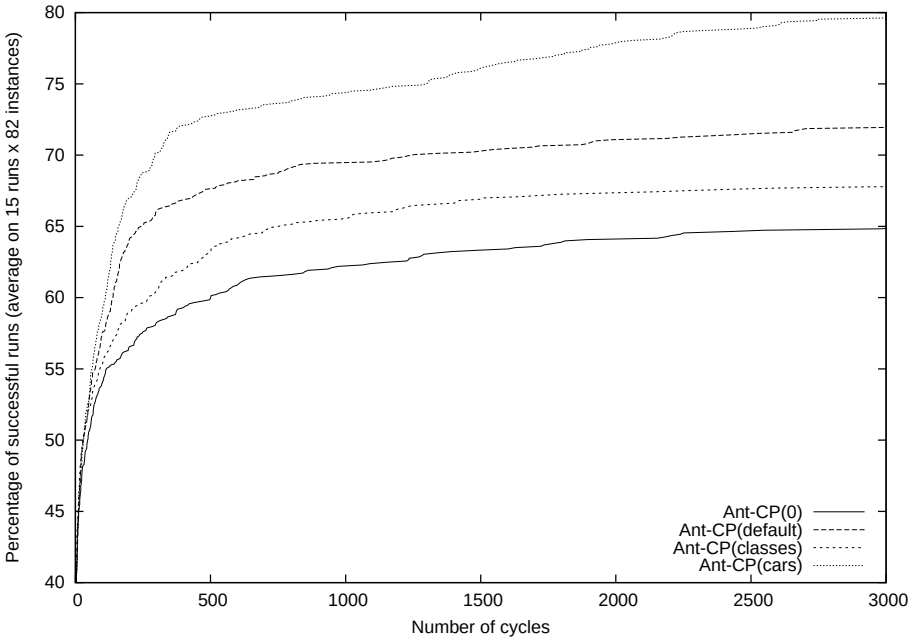

Fig. 1. Comparison of pheromone strategies.

Experimental results. We now experimentally compare Ant-CP($\Phi_{default}$), Ant-CP(Φ_{cars}), and Ant-CP($\Phi_{classes}$), which respectively use the pheromone structures $\Phi_{default}$, Φ_{cars} and $\Phi_{classes}$, with Ant-CP($\emptyset$) which ignores pheromone (i.e., $\Phi = \emptyset$ and the pheromone factor $\tau_{\mathcal{A}}(X_j, v)$ is set to 1). All experiments have been performed with the following parameter setting: $\tau_{min} = 0.01$, $\tau_{max} = 4$, $\alpha = 1$, $\beta = 6$, $\rho = 2\%$, $nbAnts = 30$, and $nbMaxCycles = 3000$.

Let us first note that the 70 instances of the test suite provided by Lee and available in CSPLib [8] are all very quickly solved by all instanciations of Ant-CP in a very few assignment constructions. It is worth mentionning here that some of these instances are still considered as difficult ones for complete branch-and-propagate based solvers [9, 10].

Fig. 1 displays the evolution of the percentage of successful runs with respect to the number of cycles on a more difficult benchmark provided by Perron and Shaw. This benchmark contains 82 instances that have between 100 and 500 cars to sequence. Fig. 1 shows that guiding the search with pheromone increases the success rate, after 3000 cycles, from 64.82% for Ant-CP($\emptyset$) to 79.62% for Ant-CP(Φ_{cars}), 71.86% for Ant-CP($\Phi_{default}$), and 67.77% for Ant-CP($\Phi_{classes}$).

These first results show that pheromone significantly improves the solution process, even when considering the default structure. Further works will mainly concern the validation of our approach on other CSPs and the integration of a reactive scheme in order to automatically adapt parameters during the search process.

References

1. Dorigo, M., Stuetzle, T.: Ant Colony Optimization. MIT Press (2004)
2. Solnon, C.: Ants can solve constraint satisfaction problems. IEEE Transactions on Evolutionary Computation **6**(4) (2002) 347–357
3. Solnon, C.: Combining two pheromone structures for solving the car sequencing problem with Ant Colony Optimization. EJOR (to appear) (2008)
4. Meyer, B., Ernst, A.: Integrating aco and constraint propagation. In: ANTS'04. Number 3172 in LNCS, Springer (2004) 166–177
5. 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. EJOR (to appear) (2008)
6. Gravel, M., Gagné, C., Price, W.: Review and comparison of three methods for the solution of the car-sequencing problem. JORS (2004)
7. Gottlieb, J., Puchta, M., Solnon, C.: A study of greedy, local search and aco approaches for car sequencing problems. In: EvoCOP. LNCS 2611, Springer (2003)
8. Gent, I., Walsh, T.: Csplib: a benchmark library for constraints. Technical report (1999) available from http://csplib.cs.strath.ac.uk/.
9. van Hoeve, W.J., Pesant, G., Rousseau, L.M., Sabharwal, A.: Revisiting the sequence constraint. In: CP 2006. LNCS 4204, Springer (2006) 620–634
10. Brand, S., Narodytska, N., Quimper, C.G., Stuckey, P.J., Walsh, T.: Encodings of the sequence constraint. In: CP. Volume 4741 of LNCS., Springer (2007) 210–224