Ant Search Strategies for Planning Optimization

M. Baioletti, A. Milani, V. Poggioni, F. Rossi
Dipartimento Matematica e Informatica
Universita di Perugia, ITALY
email: {baioletti, milani, poggioni, rossi}@dipmat.unipg.it

Abstract

In this paper a planning framework based on Ant Colony Optimization techniques is presented. It is well known that finding optimal solutions to planning problems is a very hard computational problem. Stochastic methods do not guarantee either optimality or completeness, but it has been proved that in many applications they are able to find very good, often optimal, solutions. We propose several approaches based both on backward and forward search over the state space, using several heuristics and testing different pheromone models in order to solve sequential optimization planning problems.

Introduction

Optimal sequential planning is a very hard task which consists on searching for solution plans with minimal length or minimal total cost. The cost of a plan is an important feature in many domains because high cost plans can be useless, almost like non executable plans. Several approaches to optimization planning have been recently proposed (Helmert, Do, and Refanidis 2008).

The main contribution of our work consists in investigating the application of the well known Ant Colony Optimization (ACO) meta–heuristic (Dorigo and Stuetzle 2004; Dorigo and Gambardella 1997) to optimal propositional planning. ACO is a meta–heuristic inspired by the behaviour of a natural ants colony that has been successfully applied to many *Combinatorial Optimization* problems. The first application of ACO to planning has been proposed very recently (Baioletti et al. 2009). The approach, although still preliminary, seems promising since it has been shown that stochastic approaches to planning can perform very well (Gerevini and Serina 2002). ACO seems suitable for planning because there is a strong analogy between the construction solution process of ACO and planning as progressive/regressive search in the state space. The basic idea is to use a probabilistic model of ants, in which each ant incrementally builds a solution to the problem by randomly selecting actions according to the pheromone values deposited by the previous generations of ants and to a heuristic function.

In this work we extend (Baioletti et al. 2009) by proposing several pheromone models for planning and different pheromone updating strategies. We also investigate an original extension of ACO-based backward techniques to planning. Moreover the integration of known planning heuristics in the ACO model are proposed. The forward ACO model employs the *Fast Forward (FF)* heuristic (Hoffmann and Nebel 2001), while in the backward version the heuristic h^2 in the h^m family (Haslum, Bonet, and Geffner 2005) is used. Implementations of proposed models have been tested in order to understand how the performance varies with respect to different domain classes.

Ant Colony Optimization

Ant Colony Optimization (ACO) is a meta–heuristic used to solve combinatorial optimization problems, introduced in the early 90s by (Dorigo and Stuetzle 2004), whose inspiration comes from the foraging behavior of natural ant colonies.

ACO uses a colony of artificial ants, which move on the search space and build solutions by composing discrete components. The construction of a solution is incremental: each ant randomly chooses a component to add to the partial solution built so far, according to the problem constraints. The random choice is biased by the pheromone value τ related to each component and by a heuristic function η. Both terms evaluate the appropriateness of the component. The probability that an ant will choose the component c is

$$p(c) = \frac{[\tau(c)]^\alpha [\eta(c)]^\beta}{\sum_x [\tau(x)]^\alpha [\eta(x)]^\beta} \quad (1)$$

where the sum on x ranges on all the components which can be chosen, and α and β are tuning parameters for pheromone and heuristic contributions.

The pheromone values represent a kind of memory shared by the ant colony and are subject to updating and evaporation. In particular, the pheromone can be updated at each construction step or for complete solutions (either all or the best ones) in the current iteration, possibly considering also the best solution of the previous iterations. The update phase is usually performed by adding to the pheromone values associated to components of the solution s an increment $\Delta(s)$

which depends on a solution quality function $F(s)$. [1]

The pheromone evaporation has the purpose of avoiding a premature convergence towards not optimal solutions and it is simulated by multiplying the pheromone value τ by a factor $1-\rho$, where $0 < \rho < 1$ is the evaporation rate.

The simulation of the ant colony is iterated until a satisfactory solution is found, an unsuccessful condition is met or after a given number of iterations.

ACO has been used to solve several combinatorial optimization problems, reaching in many cases state of the art performance, as shown in (Dorigo and Stuetzle 2004).

ACO and Planning

In this section we describe two different search algorithms which use ACO methodologies to solve optimization problems for propositional planning.

Forward Ants

In this first approach, ants are used to perform a forward search through the state space. The ants build a plan starting from the initial state s_0 executing actions step by step. Each ant draws the next action to execute in the current state from the set of executable actions. The choice is made by taking into account the pheromone value $\tau(a)$ and the heuristic value $\eta(a)$ for each candidate action a with a formula similar to (1). Once an action a is selected, the current state is updated by means of the effects of a.

Choosing always only executable actions means that the entire plan is executable in the initial state. The construction phase stops when a solution is found (the goals are true in the current state), a dead end is encountered (no action is executable) or an upper bound L_{max} for the the length of action sequence is reached.

Pheromone models The first choice is what is a solution component in terms of pheromone values. While it is clear that actions take surely part of components, it would be worth to add further information in order to perform a more informed choice. Therefore the function $\tau(a)$ can also depend on other data locally available to the ant.

However we have also studied a simple pheromone model, called *Action*, in which τ depends only on the action and nothing else.

We have designed and implemented four different pheromone models.

In the *State–Action* model τ also depends on the current state s. This is by far the most expensive pheromone model, because the number of possible components is exponential with respect to the problem size. On the other hand, the pheromone values have a straightforward interpretation as a policy (although in terms of a preference policy): $\tau(a, s)$ represents how much it is desirable (or better, it has been useful) to execute a in the state s.

In the *Level–Action* model τ also depends on the time step in which the action will be executed. This model is clearly more parsimonious then the state–action, but its interpretation is less clear. A drawback is that the desirability of an action at a time step t is unrelated to the values for close time steps, while if an action is desirable at time 2, i.e. at an early planning stage, it could also be desirable at time 1 and 3.

To solve this problem we introduce the *Fuzzy level–Action* model which is the fuzzified version of the previous model: the pheromone used for the action a to be executed at time t can be seen as the weighted average of the pheromone values $\tau(a, t')$ for a and for $t' = t - W, \ldots, t + W$ computed with the level–action model. The weights and the time window W can be tuned by experimental results.

In the *Action–Action* model the pheromone also depends on the last executed action. This is the simplest way in which the action choice can be directly influenced by the previous choices: using just the last choice is a sort of first order Markov property.

As we will discuss in the conclusion, this list of pheromone models is by no means exhaustive and many other pheromone models could be invented and tested.

Heuristic function The function $\eta(a)$ is computed as follows. First, we compute the state s' resulting from the execution of a in the current state s. Then the heuristic function $h_{FF}(s')$, defined in the FF planner (Hoffmann and Nebel 2001), is computed as a distance between the state s' and the goals. Therefore we set

$$\eta(a) = \frac{1}{h_{FF}(s')}$$

The computation of $h_{FF}(s')$ can also give a further information, a set H of *helpful actions*, i.e. actions which can be considered as more promising. For the actions in H we use a larger value of η, defining

$$\eta(a) = \frac{1}{(1-k)h_{FF}(s')}$$

where $k \in [0, 1]$.

Plan evaluation Plans are evaluated with respect to two different parameters. Let π a plan with length L and $\mathcal{S}_\pi =< s_0, s_1, \ldots, s_L >$ the sequence of all the states reached by π, then

$$h_{min}(\pi) = \min\{h_{FF}(s_i) \,:\, s_i \in \mathcal{S}_\pi\}$$

and

$$t_{min}(\pi) = \min\{i \,:\, h_{FF}(s_i) = h_{min}(\pi)\}$$

$h_{min}(\pi)$ is the distance (estimated by the heuristic function) between the goals and the state closest to the goals.

$t_{min}(\pi)$ is the time step at which the minimum distance is firstly obtained.

Note that, for a solution plan, $h_{min}(\pi) = 0$ and $t_{min}(\pi)$ corresponds to the length of π.

A possible synthesis of these two parameters is given by

$$P(\pi) = t_{min}(\pi) + W h_{min}(\pi)$$

where $W > 1$ is a weight which gives a large penality to non valid plans and then drives the search process towards valid plans.

[1] For minimization problem whose objective function is f, $F(s)$ is a decreasing function of f.

Pheromone updating We have implemented a Rank Ant System pheromone update method (Bullnheimer, Hartl, and Strauss 1999), which operates as follows.

The updating process is performed only at the end of each iteration. First, all the pheromone values are evaporated by a given rate ρ; then, the solutions in the current iteration are sorted in the decreasing order with respect to P.

Let $\pi_2, \ldots, \pi_\sigma$ the $\sigma-1$ best solutions found in the current iteration and let π_1 the best solution found so far, then the pheromone value $\tau(c)$ for the component c is increased by the value $\Delta(c) = \sum_{i=1}^{\sigma}(\sigma + 1 - i)\Delta_i(c)$ where

$$\Delta_i(c) = \begin{cases} \frac{1}{P(\pi_i)} & \text{if } c \text{ takes part in } \pi_i \\ 0 & \text{otherwise} \end{cases}$$

The algorithm The optimization process continues for a given number N of iterations. In each of them n_a ants build plans with a maximum number of actions L_{max}. c is the initial value for pheromone. The pseudo code of the resulting algorithm is shown in Algorithm 1.

Algorithm 1 The algorithm ACOPlan-F

```
1: π_best ← ∅
2: InitPheromone(c)
3: for g ← 1 to N do
4:     for m ← 1 to n_a do
5:         π_m ← ∅
6:         state ←initial state of the problem
7:         for i ← 1 to L_max and state is not final do
8:             A_i ← feasible actions on state
9:             a_k ← ChooseAction(A_i)
10:            extend π_m with a_k
11:            update state
12:        end for
13:    end for
14:    Update(π_best)
15:    Sort(π_1, ..., π_{n_a})
16:    UpdatePheromone(π_best, π_1, ..., π_{σ-1}, ρ)
17: end for
```

Backward Ants

The other possibility is to use backward ants. The construction phase operates in a backward direction: the ants start from the goals and, at each time step, they choose an action among those which reaches at least one current subgoal, without deleting any other subgoal. After having chosen an action a, the current subgoal set is updated by regression with respect to a.

In this way, the created sequences are always compatible with all the subgoals. The construction phase stops when a solution is found (all the current subgoals are true in the initial state), a dead end is encountered (no action is compatible with subgoals) or the upper bound L_{max} is reached.

Pheromone models The same considerations seen for the pheromone models of forward ants can also be reformulated for backward ants. We thus propose five different pheromone models. The models *Action*, *Level–Action*, *Fuzzy Level–Action* and *Action–Action* are similar to those defined for forward ants. The only difference is that the level number is counted from the goal, instead of the initial state.

While the *State–Action* model cannot be defined in this framework, as subgoals are not complete states, it is possible to define a new model, called *Goal–Action* in which τ also depends on the goal which is reached by a. This model has a nice interpretation in that $\tau(a, g)$ quantifies how much it is promising to reach the goal g by executing a.

Heuristic function The heuristic value for an action a is computed by means of the function h^2_{HSP}, described in (Haslum, Bonet, and Geffner 2005), which estimates the distance between the initial state and the subgoals g', obtained by regression of the current subgoals with respect to a.

Besides the fact that h^2_{HSP} is suited for a backward search, it is important to note that most of the operations needed to compute it are performed only once, at the beginning of the search process. Moreover it is worth to notice that h^2_{HSP} can be easily adapted to planning problems in which actions can have a non fixed cost.

Plan evaluation and pheromone update The plans are evaluated in a similar way as done for forward ants, except that h_{min} is defined in terms of the sequence $\mathcal{G}_\pi =< g_0, g_1, \ldots, g_L >$ of subgoal sets generated by π.

Also the pheromone update process is exactly the same. Note that, also in this case, $h_{min}(\pi) = 0$ if and only if π is a valid plan and thus, for valid plans, $P(\pi)$ reduces to the length of π.

The algorithm The algorithm, called *ACOPlan–B*, is very similar to *ACOPlan–F* and will not be reported. The only relevant difference is that the state is replaced by the subgoals set.

Experimental results

We chose to compare ACOPlan with the LPG system (Gerevini and Serina 2002). The choice is twice motivated: LPG is a stochastic planner and it is very efficient from both a computational and a solution quality point of view. In fact, when it runs with particular settings [2], it gives solution plans with, in general, a number of actions very close to the optimum (often it can find solutions with the optimum number of actions). The test are running in two parallel tracks, the first one for ACOPlan–F and the second one for ACOPlan–B. The focus of this section is on the first track, because the results for the second track are too preliminar to be presented and many aspects of the implementation are under revision. The first track evaluates the performance of ACOPlan–F with respect to the different pheromone models using some domains taken from the International Planning Competitions (IPC).

So far we have run a set of systematics tests over the domains *Rovers*, *Driverlog* and *Satellite* from IPC3 and the domains *Openstacks*, *Parcprinter* and *Pegsol* from IPC6. These domains have been chosen among the set of bench-

[2] In our experiments we used the *-n* modality with high values of n.

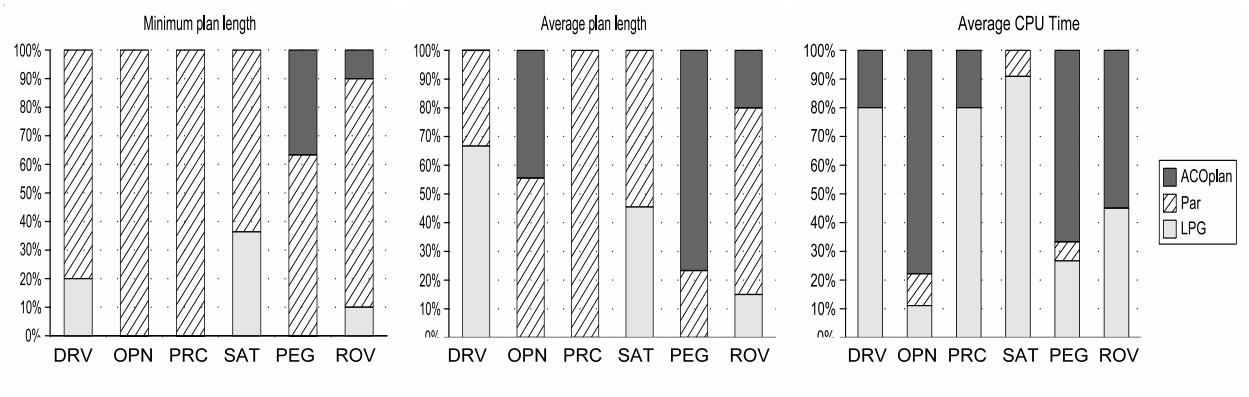

Figure 1: Results collected for the domains *DriverLog* (DRV),*Openstacks* (OPN), *ParcPrinter* (PRC), *Satellite* (SAT), *PegSol* (PEG) and *Rover* (RVR)

marks because they offer a good variety and the corresponding results allow us interesting comments.

ACOPlan–F has many parameters that have to be chosen. After systematic tests we decided to use this setting: 10 ants, 5000 iterations, $\alpha = 2$, $\beta = 5$, $\rho = 0.15$, $c = 1$, $k = 0.5$.

Since both systems are non deterministic planners, the results collected here are the mean values obtained over 100 runs. Both the systems ran with the time-limits of 600 secs.

In Fig.1 we summarize the global results along three dimensions: the minimum and the average values of the solution length, and the average CPU time spent to extract the solution plan with minimum length in each run. The histograms show the results for the comparison between ACOPlan and LPG. Each graph collects the results along one of the three dimensions (respectively minimum solution length, average solution length and average CPU time) for all the cited domains, representing with different coloration the cases when ACOPLan is the winner, the cases when LPG is the winner and the cases ended in a draw (represented by *Par* in the graph).
The results we collected show that ACOPLan–F is comparable with LPG both in terms of solution quality and in terms of the computation time.

Conclusions and Future Works

In this paper we have described a planning framework based on Ant Colony Optimization meta–heuristic to solve optimal sequential planning problems. The preliminary empirical tests have shown encouraging results and that this approach is a viable method for optimization in classical planning. For these reasons we are thinking to improve and extend this work in several directions.
We are planning to perform extensive sets of experiments in order to compare the different pheromone models and to contrast the difference between forward and backward ants. Another possibility could be to define other pheromone models and using different ACO techniques.
The heuristic function used for forward ants should be replaced with a heuristic function which can handle action costs, following the approach proposed in the FF(h_a) system (Helmert, Do, and Refanidis 2008).

Finally, we are considering to apply ACO techniques also to other types of planning. The extensions of classical planning which appear to be appealing for ACO are planning with numerical fluents and planning with preferences, which are optimization problems as well.

References

Baioletti, M.; Milani, A.; Poggioni, V.; and Rossi, F. 2009. An aco approach to planning. In *In Proc of the 9th European Conference on Evolutionary Computation in Combinatorial Optimisation, EVOCOP 2009*.

Bullnheimer, B.; Hartl, R. F.; and Strauss, C. 1999. A new rank based version of the ant system - a computational study. *Central European Journal for Operations Research and Economics* 7:25–38.

Dorigo, M., and Gambardella, L. M. 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1):53–66.

Dorigo, M., and Stuetzle, T. 2004. *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press,.

Gerevini, A., and Serina, I. 2002. Lpg: a planner based on local search for planning graphs. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS'02), AAAI Press, Toulouse, France*.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proc. of AAAI 2005)*, 1163–1168.

Helmert, M.; Do, M.; and Refanidis, I. 2008. International planning competition ipc-2008, the deterministic part. http://ipc.icaps-conference.org/.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253 – 302.