Experimental evaluation of pheromone models in ACOPlan

M.Baioletti, A.Milani, V.Poggioni, and F.Rossi

Department of Mathematics and Computer Science
University of Perugia
Via Vanvitelli 1, 06123 Perugia, ITALY
{baioletti,milani,poggioni,rossi}@dmi.unipg.it

Abstract

ACOplan is a planner based on the ant colony optimization framework. Using the ACO framework to solve planning optimization problems, one of the main issues to address is the choice of informative and easy to compute pheromone models. In this paper we present and discuss an experimental evaluation of the several pheromone models implemented in ACOPlan. The experiments have been run solving the problems used in the last planning competition. The results suggest that fuzzy pheromone models represents a suitable tradeoff between performance and cost.

1 Introduction

A planning problem is usually defined as a search problem: finding a solution plan which satisfies the problem constraints. An important improvement of this definition is to consider planning as an optimization problem, with respect to a given metric.

Propositional planning are usually optimized with respect to the plan length or to the number of time steps (if parallel actions are allowed). For instance, HSP [6], using an admissible heuristic function, is able to provide the former kind of optimization, while many algorithms, including GraphPlan [5], BlackBox [10] and others, can optimize in the latter form, by using an iterative deepening search method.

However the planning frameworks in which optimization seems to be more natural are planning with *action costs* and planning with *numerical fluents*. In the first, the actions are associated to a real number representing their execution cost, while in the second actions can handle and modify numerical variables and an objective function can be easily expressed in terms of some target variables (for instance fuel as in *Logistics* domain).

The main idea of this paper is to use the well known *Ant Colony Optimization* meta–heuristic (henceforth *ACO*) to solve planning problems with the aim of optimizing the quality of the solution plans. ACO is a meta–heuristic inspired by the behavior of natural ants colony which has been successfully applied to many *Combinatorial Optimization* problems. Clearly,

Proceedings of the 17th International RCRA workshop (RCRA 2010):
Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion
Bologna, Italy, June 10–11, 2010

being ACO a stochastic algorithm, there is no guarantee that optimal solutions are ever found, but we hope that, as shown in many other applications of ACO, this method can produce excellent or optimal solutions.

To test if ACO can be effectively used in optimization planning problem we have implemented, as a first step, an ACO–based propositional planner *ACOPlan* which tries to optimize plans in terms of plan length. This system has been presented in [1, 2, 3, 4]. After the good results shown in these works, we decided to extend ACOPlan to more complex planning frameworks, like planning with action costs and numerical planning. In this work we present the extension to planning with action costs and an experimental evaluation of this system by using the most recent planning benchmarks for planning with action costs [8].

The paper is structured as follows. A short introduction to Ant Colony Optimization is provided, then the ACOPlan algorithm is presented in a recent version applied to the framework of planning with action costs. Four pheromone models are also introduced and motivated. The experimental evaluation of the proposed models on the above cited benchmark is presented and discussed. The paper concludes with a discussion of lessons learned and future works and extensions.

1.1 Ant Colony Optimization

Ant Colony Optimization (ACO) is a meta–heuristic used to solve combinatorial optimization problems, introduced since early 90s by Dorigo et al. [7], whose inspiration comes from the foraging behavior of natural ant colonies.

ACO uses a colony of artificial ants, which move in 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 each 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 update 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 values 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 art performance, as shown in [7]. More recently ACO has also been applied to automated planning[1, 2, 3, 4].

2 The ACOPlan planner

Automated planning is a well known AI problem deeply studied in the last years. Many different kinds of problems defined over several planning models have been proposed. Our aim is to solve planning problems optimizing the cost of the solution plans, where the cost of a plan is defined in terms of the sum of the execution costs of its actions.

In the standard reference "classical" model for planning [12], world states are described in terms of a finite set $\mathcal{F}$ of propositional variables under a closed world assumption (CWA), i.e. a state s is represented with a subset of $\mathcal{F}$, containing all the variables which are true in s.

A Planning Problem is defined by a triple $(\mathcal{I}, \mathcal{G}, \mathcal{A})$, in which $\mathcal{I}$ is the initial state, $\mathcal{G}$ denotes the goal, and $\mathcal{A}$ is a finite set of actions.

An action $a \in \mathcal{A}$ is described by a triple $(pre(a), add(a), del(a))$, where $pre(a)$, $add(a)$, $del(a) \subseteq \mathcal{F}$. $pre(a)$ is the set of preconditions: a is executable in a state s if and only if $pre(a) \subseteq s$. $add(a)$ and $del(a)$ are, respectively, the sets of positive and negative effects: if an action a is executable in a state s, the state resulting after its execution is

$$Res(s, a) = (s \cup add(a)) \setminus del(a). \tag{2}$$

Otherwise $Res(s, a)$ is undefined.

A linear sequence of actions $(a_1, a_2, \ldots, a_n)$ is a plan for a problem $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ if a_1 is executable in $\mathcal{I}$, a_2 is executable in $s_1 = Res(\mathcal{I}, a_1)$, a_3 is executable in $s_2 = Res(s_1, a_2)$, and so on. A plan $\pi = (a_1, a_2, \ldots, a_n)$ is a (classical) solution for $(\mathcal{I}, \mathcal{G}, \mathcal{A})$ if $\mathcal{G} \subseteq s_n$.

Several extensions have been proposed to the classical model, by the adding notions such as time, cost, resource consumption, etc., and increasing the complexity of the domain and the constraints.

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

In a planning model with action execution costs, the problem consists in finding an optimal plan which achieve the goals in $\mathcal{G}$. A real constant, defined as action execution cost $c(a)$, is associated to each action $a \in \mathcal{A}$, and the *plan execution cost* is defined accordingly by $c(\pi) = \sum_{i=1}^{n} c(a_i)$. A plan $\pi = (a_1, a_2, \ldots, a_n)$ is an optimal solution plan if there is no other solution plans π' such that $c(\pi) > c(\pi')$.

The idea is to use an ACO approach to solve this kind of optimization problem, by "ants" which perform a forward search in the state space building a plan starting from the initial state s_0 and executing actions step by step. Each ant extracts the next action to execute in the current state from the set of candidate executable actions. The choice is made by a randomized weighted extraction which takes into account the pheromone value τ associated to each solution component and the heuristic value $\eta(a)$ for each candidate action a with a formula similar to (1). Once an action a has been selected, the current state is updated by means of the effects of a.

By choosing always only executable actions is possible to guarantee that the entire plan is executable in the initial state. The construction phase stops when (1) a solution is found (the goals are true in the current state), (2) when a dead end is encountered (no action is executable) or (3)when an upper bound L_{max} for the the length of action sequence is reached. Better solution plans will strengthen the pheromone they deposit.

2.1 The algorithm

The general algorithm of ACOPlan [1, 2, 3, 4], was originary proposed for optmizing the plan length. 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 limited to L_{max}. The pseudo code of the resulting algorithm is shown in Algorithm 1 where c is the initial value for pheromone.

2.2 The Pheromone Models

The performance of the general algorithm described above, strongly depends on the choice of the next action, i.e. on the pheromone model which is used. A pheromone model consists of a set of solution *components* loaded with pheromone, where a component is a structure characterizing the context in which an ant has chosen a specific action. The more successful a component is, the more pheromone it will receive. The pheromone amount on the alternative components represent the distribution of τ value in (1). A good component should be enough informative to distinguish the context of most successful choices from the worst ones, in other word in order to represent the local strategies of successful ants, moreover components should be easy to represent and calculate. The following four pheromone models has been considered and experimented with the ACOPlan algorithm.

Algorithm 1 The algorithm ACOPlan

```
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
```

State–State (SS) In this case a component is defined by a pair (s_i, s_j) where s_i is the current state and s_j is the state reachable by the execution of one of the actions executable in s_i.

In this case only state information are shared among ants: state transitions which appear in more successful solutions will have their pheromone levels increased. In the other three models, instead, also information about actions is considered in the components.

State–Action (SA) In the *State–Action* model, τ depends on the current state s and on the action a we would execute. This is one of the most expensive pheromone model by far, 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.

Fuzzy level–Action (FLA) The basic idea is to associate the action which is evaluated with the plan step, i.e. the level, where it is executed. Since the limited number of levels, such a model has a more tractable representation with respect to the state–action one. A drawback is that strictly associating the preference policy of an action to a time step t makes it unrelated to the values for close time steps, on the other hand it is likely that an action desirable at time 2, i.e. at an

early planning stage, it will also be desirable at time 1 and 3, i.e. a stage close to an early ones. To solve this problem we introduce the *Fuzzy level–Action* model which is the fuzzified version of the combination level–action: the pheromone used for 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.

Action–Action (AA) In the *Action–Action* model a notion of *local history* is introduced: the pheromone depends both on the action under evaluation and on the last executed action. This is the simplest way in which the action choice can be directly influenced by the previous choices. Considering only the last choice allows to have a manageable representation, and it results in a sort of local first order Markov property.

2.3 The Action_Cost heuristic

The *Action_Cost* heuristic, i.e. the estimated cost to reach the goals starting from a state s, is computed on the delete relaxation P^+ of the problem by extracting a relaxed plan π^+ in a *FF*–style [9] from the relaxed planning graph. The planning graph [5] is a leveled graph with two kinds of nodes: facts and actions. The initial fact level F_0 is the set of facts that are true in the initial state s_0, while the levels A_k and F_{k+1} at a generic step k are respectively computed considering the actions executable using the facts at the level F_k and the positive effects of these actions.
The computation of the heuristic is defined by the cost of π^+:

$$c(\pi^+) = \sum_{a \in \pi^+} c(a) \tag{3}$$

where a is an action belonging to π^+. The extraction process of π^+ is similar to the *FF* one, except that the *difficulty* of an action, taken into account when next actions to add to π^+ at the level k have to be chosen (see [9] for details), is the action cost at the level $k - 1$, which is recursively estimated as follows:
Action costs.

$$c_k(a) = c(a) + \sum_{f \in pre(a)} c_k(f). \tag{4}$$

Fluent costs.

$$c_k(f) = \min\{c(a) + \sum_{b \in pre(a)} c_{k-1}(b) \mid a \in A_{k-1}, f \in \mathit{eff}(a)\}. \tag{5}$$

where $c_k(f)$ is the cost we have to spend to reach f at the level F_k, $c(a)$ is the cost assigned to the action a, A_{k-1} is the set of feasible actions in the facts level F_{k-1}. Moreover, we have $c_0(f) = 0,\ \forall f \in F_0$.
Notice that the cost of a fluent (and of an action too) may be different depending on the level in the planning graph.

This heuristic can be related to the ones presented in [11].

3 Experimental Evaluation

This section presents and discusses the results obtained by the different pheromone models for ACOplan previously described. The optimization problem is to minimize the overall plan execution cost. *State-State*, *State-Action*, *Action-Action* and *Fuzzy-Level-Action* models have been experimented with the benchmark domains of the planning competition IPC-6 [8].

A preliminary phase of parameters tuning has been held by systematic tests in order to establish the ACOPlan general setting which are common to all the four pheromone models, then the phase of actual experiments has been conducted with ACOPlan settings: 10 ants, 5000 iterations, $\alpha = 2$, $\beta = 5$, $\rho = 0.15$, $c = 1$, $k = 0.5$. Moreover since ACOPlan is a not deterministic planner, the results collected for each single problem instance are the average values obtained over 50 runs. The experiments were run on a Linux machine with a dual core processor and 2 GB of RAM.

In the following, the experimental results for ACOPlan over the domains [8] *Elevators*, *Openstacks*, *Parcprinter*, *Pegsol*, *Transport* and *Woodworking* are presented.

In Fig.1 it is shown, for each domain, the percentage of problems solved in 900 seconds. A first point which can be noted is that the best performing pheromone model is the *Action-Action* model, while the *State-Action* model is the worst performing one. This confirms the results obtained by the first version of ACOPlan (optimizing the plan length), presented in [13], where the model *Action-Action* has been introduced.

On the other hand a very encouraging new result has been obtained: ACOPlan with *Fuzzy-Level-Action* pheromone model performs very close to the *Action-Action* model, considering also that *Fuzzy-Level-Action* is much more tractable.

Due to the high variability of the absolute values of the solution costs, in order to compare the different solutions for all the problems in each domain, we have scaled and normalized all the quality values in the interval $[0, 1]$. First of all we have taken the cost of the best solution found by any model; then, for each problem and for each program version, we have computed the ratio between the best value and the average value obtained. The results are shown in the group of figures Fig.2 - Fig.7.

	AA	FLA	SA	SS
Elevators	89%	83%	81%	83%
Openstacks	100%	100%	100%	100%
Parcprinter	80%	80%	72%	74%
Pegsol	100%	100%	100%	100%
Transport	82%	84%	82%	82%
Woodworking	100%	100%	100%	100%

Figure 1: Percentages of problems solved in 900 secs for each IPC-6 domains.

We can note that all the pheromone models perform equally good for the first and the easiest problem instances in each domain; this is due to the fact that the problem instances in benchmark domain are ordered with respect to increasing size, which roughly correspond to an increasing optimization and problem solving difficulty.

A general comment on the overall plan quality result is that the prevalence of *Action-Action* ACOPlan and the bad performance of *State-Action* observed on the number of solved problems, see Fig.1, is generally confirmed for the quality found. In other words the problem solving ability and the optimization ability of all the pheromone models seems to follow a similar pattern. On the other hand even the worst performing pheromone models are within the 80 percent of the best quality found so far in 3 domains out of 6. The reason for the correspondence between quality and solving ability in ACOPlan is that a more successful strategy, in a randomized algorithms, tends to find many diverse solutions, while a successful deterministic strategy tends to find fast solutions, which are very close to each other, i.e. with close quality values.

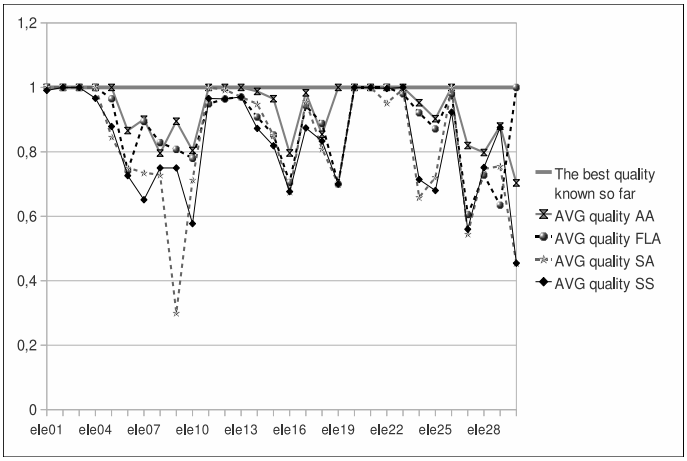

Figure 2: Best quality *elevators* domain,

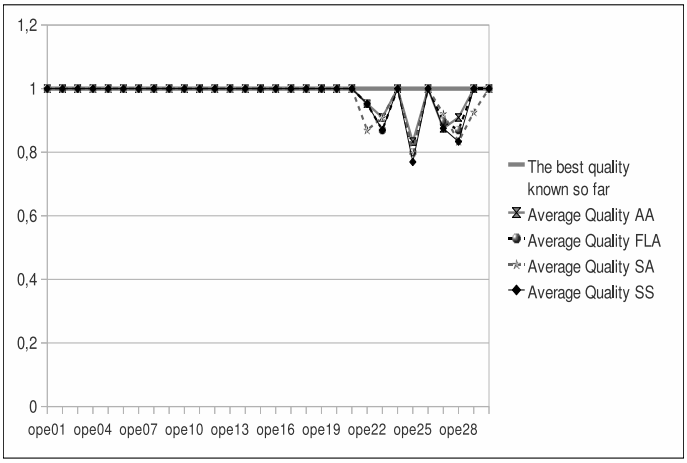

Figure 3: Best quality *openstacks* domain.

Domains which are considered "difficult", such as *elevators*, *transport* and *woodworking* show a prevalent bad performance for component models *State-State* and *State-Action*. The intuition would suggest instead that *State-State* and *State-Action* characterize the context where a choice is operated by ants, since a pair of states represents the most detailed description of a local context, the best performance of more simplified contexts such as *Action-Action* pair and *Fuzzy-Level-Action* seems to be counterintuitive. A careful analysis has been done of pheromone distribution for SS and SA on the worst performing problem instances, such as *elevator9*, *parcprinter19*, *transportation15*. The analysis has revealed that the difficult seems to lie exactly in a too high level of details, i.e. in the proliferation of components. In other words if each different state originates one (or more) different component, similar successful states, which could differ for just one fluent value, do not accumulate enough pheromone, a fact which prevent them to be preferred in ants choices.

We have observed that in situations where SS and SA are performing bad, the pheromone is spread over too many components, which are not likely to appear again in different runs of the ants, even if the plans found in the run slightly differ from the previous ones. On the other hand a less detailed description of the context, like in the *Action-Action* model, allows the accumulation of pheromone over components which appear more easily in different successful and similar runs.

Let consider for example two successful plans $\pi_1 = \{A_0, \ldots, A_i, A_{i+1}, \ldots, A_n\}$ and $\pi_2 = \{A_0, \ldots, A_{i+1}, A_i, \ldots, A_n\}$ which differs only for two consecutive switched actions A_i and A_{i+1}. The two plans in the *Action-Action* model will only differ *locally* for the three components (A_{i-1}, A_i) , (A_i, A_{i+1}) and (A_i, A_{i+1}), while in the *State-State* (or in the *State-Action*) models they will potentially differ on all the components starting from the state produced by

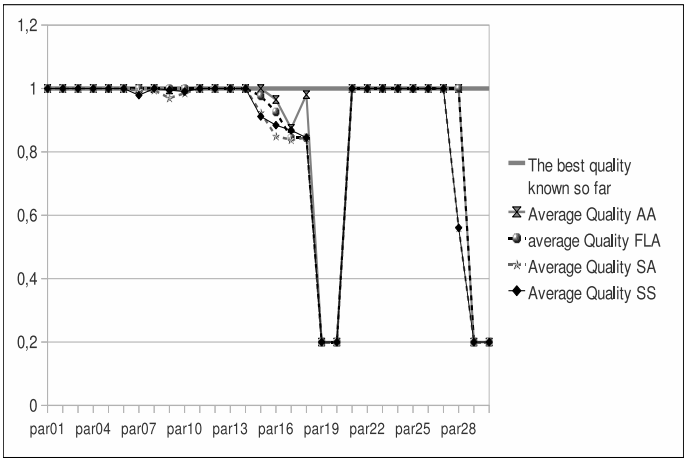

Figure 4: Best quality *parcprinter* domain.

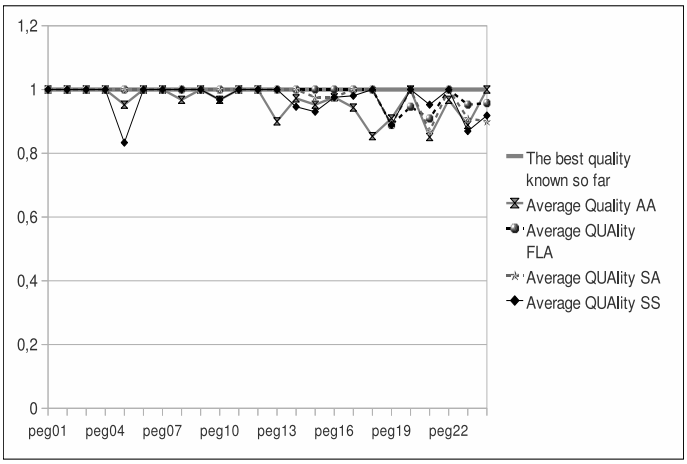

Figure 5: Best quality *pegsol* domain.

action A_{i-1} onward, till the end of the plans. It is apparent that the earlier a different state is produced the greatest is the impact on the component differences among plans which are similar.

Considering the *global plan quality* shown in Fig.8 the *Fuzzy-Level-Action* still confirm its proximity to *Action-Action*, i.e. FLA is always performing second and it is the best performing one also in the only domain (see *pegsol* domain) where AA is not. The global quality results for *Openstacks* also point out that it is in general an easy problem domain not significant to address quality performance comparisons.

The performance of *Fuzzy-Level-Action* (FLA) can be explained by reasons similar to the ones supporting the better behavior of *Action-Action*(AA) with respect to *state based* components. In fact in *Fuzzy-Level-Action* (FLA)

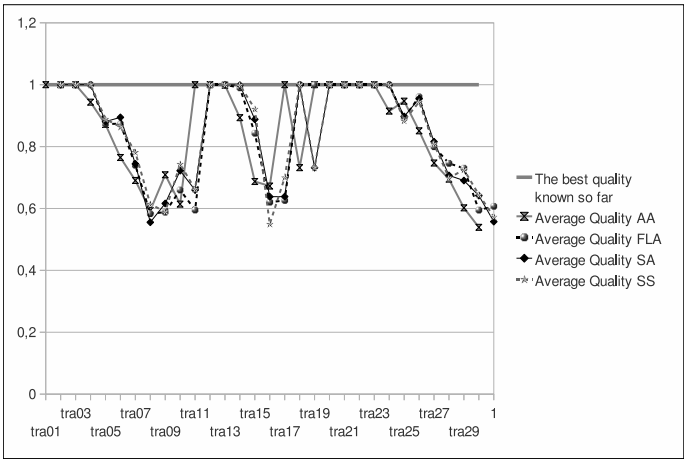

Figure 6: Best quality *transport* domain.

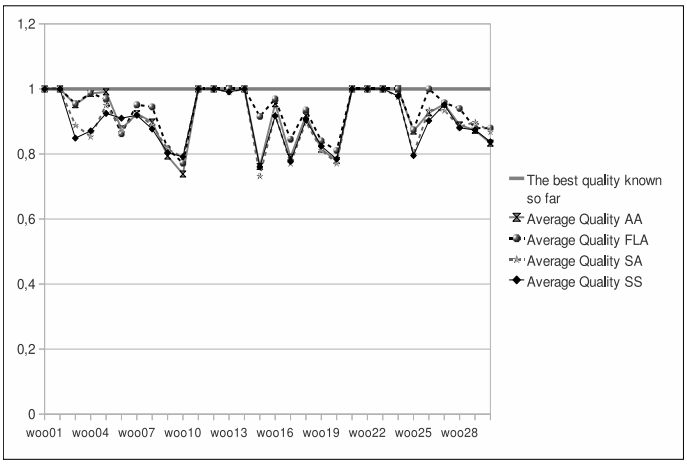

Figure 7: Best quality *woodworking* domain.

two similar plans, in the sense of the previous π_1 and π_2, receive pheromone in similar components: the model is *robust* with respect to local differences such as consecutive action switching or action insertion in a plan.

Moreover the FLA model is more flexible with respect to AA. In AA the pheromone for plan π_1 is given, for instance, to (A_i, A_{i+1}) whereas pheromone for plan π_2 is put on component (A_{i+1}, A_i). In FLA instead, switching actions A_i and A_{i+1} will not prevent the two components (A_i, i) and $(A_i, i+1)$ to receive pheromone from both plans, although they will receive a fuzzy different amount. *Fuzzy-Level-Action* is then a worthwhile compromise to obtain a good performance close to *Action-Action* without the associated cost.

In general FLA tends to learn in which time step (or nearby) is useful

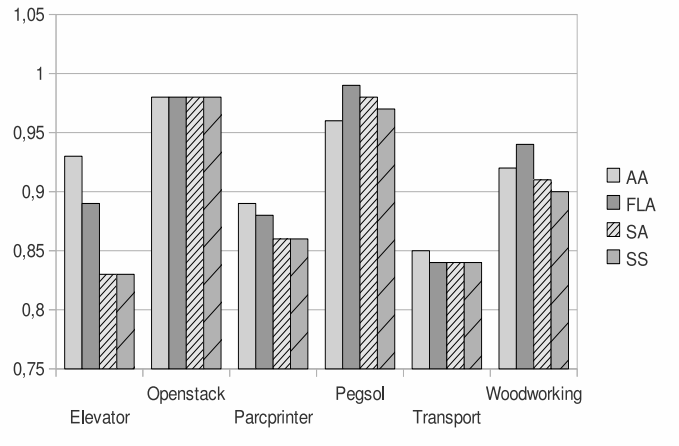

Figure 8: For each domain, each column represents a synthesis of the average quality found by means of each model. AA(Action-Action), FLA(Fuzzy-Level-Action), SA(State-Action), SS(State-State).

to apply an action, while AA tends to learn which action is useful to apply after a given one. It is worth noticing that in specific domains like *pegsol* the better performance of all the pheromone models with respect to AA is basically due to the fact that the domain is easy to solve and to optimize for an ACOPlan solver, i.e. all the models solve the *pegsol* domain problems finding solutions within 95 percent of the optimum. The ability of a detailed distinction of the context, for instance applying an action at a certain precise step, is then crucial in finding and refining the optimal value, i.e. to improve the local search.

4 Conclusions

The experiments of ACOPlan in the framework of planning with action costs, suggest that action based pheromone models outperforms state based ones, while the proposed novel pheromone model based on *Fuzzy-Level-Action* components has been proved to be a promising and effective tradeoff between performance (both for solving and optimization capabilities) and space cost.

Further experiments are needed to confirm the behavior of FLA model in more general optimization contexts such as multiple costs and cost metrics involving resource consumption.

More general lessons have been learned by a deeper understanding the role of component models in the problem solving and optimization capability of ACOPlan, which can be a useful guideline for further development of ACOplan and at some extent in the broader context of ACO algorithms:

- in order to produce effective search strategies which *distinguish a use-*

ful situation, the components should capture the context in which an action is applicable, or a decision is taken (*maximum context details*);

- the component should not proliferate too much, for performance reason, and more importantly in order to *allow the pheromone deposits* and concentrates on frequent components, in other words the components should not be too much descriptive in order to avoid to spread the pheromone on too many components which seldom will be found again in the search space (*minimum context details*)

- the model components should include *elements of the solution plans* (actions and precedence constraints, time steps or plan history etc.) instead of properties of entities which can greatly differ in similar solution plans (e.g. a complete state description)

- the model components should verify the property that *two similar solution should produce similar components*, according to a given notion of similarity, i.e. a minimal perturbation on a solution producing a solution with the same optimality value should deposit a similar amount of pheromone on similar components.

This latter property has been shown to hold for AA and more strongly for FLA, allowing the pheromone to deposit on most of the components of perturbed solutions.

Particularly promising will be investigating the extension of the technique of *component fuzzification* introduced with FLA. According to this principle, given a metric distance D among components, a pheromone model can be fuzzified providing that: each component C which is similar to a given one C_{sol} appearing in a solution, will receive a pheromone amount in proportion inverse to the distance $D(C, C_{sol})$. The technique has a general aim and it can also be applied to *state* based pheromone models, although these latter ones still present space complexity problems.

Another line of future research will be the investigation of ACOPlan with multiple pheromone models where the transition probabilities are computed by using a normalized weighted combination of different pheromone types.

References

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

[2] M. Baioletti, A. Milani, V. Poggioni, and F. Rossi. Ant search strategies for planning optimization. In *Proc of the International Conference on Planning and Scheduling, ICAPS 2009*, 2009.

[3] M. Baioletti, A. Milani, V. Poggioni, and F. Rossi. Optimal planning with ACO. In *Proc of AI*IA 2009, LNCS 5883*, pages 212–221, 2009.

[4] M. Baioletti, A. Milani, V. Poggioni, and F. Rossi. PlACO: Planning with Ants. In *Proc of The 22nd International FLAIRS Conference. AAAI Press*, 2009.

[5] A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, (90):281–300, 1997.

[6] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129((1-2)), 2001.

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

[8] M. Helmert, M. Do, and I. Refanidis. International Planning Competition IPC-2008, The Deterministic Part. http://ipc.icaps-conference.org/, 2008.

[9] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14:253 – 302, 2001.

[10] H. Kautz and B. Selman. Unifying sat-based and graph-based planning. In *Proc. of IJCAI-99*, 1999.

[11] E. Keyder and H. Geffner. Heuristics for planning with action costs revisited. In *Proceedings of ECAI 2008*, pages 588–592, 2008.

[12] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[13] F. Rossi. *An ACO approach to Planning*. PhD thesis, PhD Thesis, Mathematics and Computer Science Dept., University of Perugia, Italy, 2009.