# Updating Strategies of Policies for Coordinating Agent Swarm in Dynamic Environments

Richardson Ribeiro[1], Adriano F. Ronszcka[1], Marco A. C Barbosa[1],
Fábio Favarim[1] and Fabrício Enembreck[2]
[1]*Department of Informatic, Federal University of Technology - Parana, Pato Branco, Brazil*
[2]*Pos-Graduate Program in Computer Science, Pontificial Catholical University - Parana, Curitiba, Brazil*

Keywords: Swarm Intelligence, Ant-Colony Algorithms, Dynamic Environments.

Abstract: This paper proposes strategies for updating action policies in dynamic environments, and discusses the influence of learning parameters in algorithms based on swarm behavior. It is shown that inappropriate choices for learning parameters may cause delays in the learning process, or lead the convergence to an unacceptable solution. Such problems are aggravated in dynamic environments, since the fit of algorithm parameter values that use rewards is not enough to guarantee a satisfactory convergence. In this context, strategy-updating policies are proposed to modify reward values, thereby improving coordination between agents operating within dynamic environments. A framework has been developed which iteratively demonstrates the influence of parameters and updating strategies. Experimental results are reported which show that it is possible to accelerate convergence to a consistent global policy, improving the results achieved by classical approaches using algorithms based on swarm behavior.

## 1 INTRODUCTION

When properly applied, coordination between agents can help to make the execution of complex tasks more efficient. Proper coordination can help to avoid such complications as finding redundant solutions to a sub-problem, inconsistencies of execution (such as up-dating obsolete sub-problems), loss of resources, and deadlocks (waiting for events which will probably not occur) (Wooldridge, 2002).

Real-world activities which require coordinated action include traffic environments (Ribeiro et al., 2012), sensor networks (Mihaylov et al., 2009), supply chain management (Chaharsooghi et al., 2008), environmental management, structural modeling, and dealing with the consequences of natural disasters. In such applications agents act in uncertain environments which change dynamically. Thus, autonomous decisions must be taken by agents themselves in the light of what they perceive locally.

In such applications, agents must decide upon courses of action that take into account the activities of other agents, based on knowledge of the environment, limitations of resources and restrictions on communication. Methods of coordination must be used to manage consequences that result when agents have inter-related objectives: in other words, when agents share the same environment or share common resources.

The paradigm for coordination based on swarm intelligence has been extensively studied by a number of researchers (Dorigo, 1992), (Kennedy et al., 2001), (Ribeiro and Enembreck, 2012), (Sudholt, 2011). It is inspired by the behavior of colonies of social insects, with computational systems reproducing their behavior exhibited when solving collective problems: typically the colonies are those of ants, bees, woodlice or wasps. Such colonies have desirable characteristics (adaptation and coordination) which find solutions to computational problems needing concerted activity. Earlier research on the organization of social insect colonies and its applications for the organization of multi-agent systems has shown good results for complex problems, such as combinatorial optimization (Dorigo and Gambardella, 1996).

However, one of the main difficulties with such algorithms is the time required to achieve convergence, which can be quite expensive for many real-world applications. In such applications, there is no

Ribeiro R., F. Ronszcka A., A. C. Barbosa M., Favarim F. and Enembreck F..
Updating Strategies of Policies for Coordinating Agent Swarm in Dynamic Environments.
DOI: 10.5220/0004443703450356
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 345-356
ISBN: 978-989-8565-59-4
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

guarantee that reward-based algorithms will converge, since it is well known that they were initially developed and used to cope with static problems where the objective function is invariant over time. However, few real-world problems are static in which changes of priorities for resources do not occur, goals do not change, or where there are tasks that are no longer needed. Where changes are needed through time, the environment in which agents operate is dynamic.

The use of methods based on insect behavior, of ant colonies in particular, has drawn the attention of researchers who reproduce sophisticated exploration strategies which are both general and robust (Dorigo and Gambardella, 1996); but in most cases such approaches are not able to improve coordination between agents in dynamic conditions due to the need to provide adequate knowledge of changes in environment.

The work reported here re-examines and extends principles presented in (Ribeiro et al., 2012) and (Ribeiro et al., 2008), which discussed approaches for updating policies in dynamic environments and analysed the effects of strengthening learning-algorithm parameters in dynamic optimization problems. To integrate such approaches into updating strategies, as proposed in this paper, a test framework was developed to iteratively demonstrate the influence of parameters in the *Ant-Q* algorithm (Gambardella and Dorigo, 1995), whilst agents respond to the system using updating strategies, further discussed in Section 3.

One important aspect is to investigate whether policies learned can be used to find a solution rapidly after the environment has been modified. The strategies proposed in this paper respond to changes in the environment. We cite as one example of dynamic alteration the physical movement of the vertex in a graph (the environment): i.e., the change in vertex coordinates, in a solution generated with a Hamiltonian cycle. Such strategies are based on rewards (pheromones) from past policies that are used to steer agents towards new solutions. Experiments were used to compare the utilities of policies generated by agents using strategies proposed. The experiments were run using benchmark: eil51 e eil76, found in the online library TSPLIB (Reinelt, 1991).

The paper is organized as follows: Section 2 describes some approaches related to optimization and ant-colony algorithms. Section 3 then describes policy up-dating strategies based on ant-colonies, and the framework developed for evaluating the proposed approaches. Experimental results are set out in Section 4, and the final Section 5 lists conclusions and discusses ideas for future work.

## 2 OPTIMIZATION AND ANT-COLONY ALGORITHMS

Ant colony optimization (ACO) (Dorigo, 1992) is a successful population-based approach inspired by the behavior of real ant colonies: in particular, by their foraging behavior. One of the main ideas underlying this approach is the indirect communication among the individuals of a colony of agent, called (*artificial*) *ants*, based on an analogy with pheromone trails that real ants use for communication (pheromones are an odorous, chemical substance). The (artificial) pheromone trails are a kind of distributed numeric information that is modified by the ants to reflect their accumulated experience while solving a particular problem.

Ant-colony algorithms such as *Ant System* (Dorigo, 1992), *Ant Colony System* (Dorigo and Gambardella, 1996) and *Ant-Q* (Gambardella and Dorigo, 1995) have been applied with some success to combinatorial optimization problems, including the traveling salesman problem, graph coloring and vehicle routing. Such algorithms are based on the foraging behavior of ants ("*agents*") which follow a decision pattern based on probability distribution (Dorigo et al., 1996).

Gambardella and Dorigo (1995) developed the algorithm *Ant-Q,* inspired by the earlier algorithm *Q-learning* of Watkin and Dayan (1992). In *Ant-Q,* the pheromone is denoted by *AQ-value* (*AQ*(*i*,*j*)). The aim of *Ant-Q* is to estimate *AQ*(*i*,*j*) as a way to find solutions favoring collectivity. Agents select their actions based on transition rules, as given in equations 1 and 2:

$$s = \begin{cases} \arg\max\limits_{j \in N_k(t)} \left\{ [AQ(i,j)]^{\delta} \times [HE(i,j)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \tag{1}$$

where the parameters $\delta$ and $\beta$ represent the weight (influence) of the pheromone *AQ*(*i*,*j*) and of the heuristic *HE*(*i*,*j*) respectively; *q* is a value selected at random with probability distribution [0,..,1]: the larger the value of $q_0$, the smaller is the probability of the random selection; *S* is a random variable drawn from the probability function *AQ*(*i*,*j*); and the *HE*(*i*,*j*) are heuristic values associated with the link (*i*,*j*) (edge) which helps in the selection of adjacent states (vertices). In the case of the traveling salesman problem, it is taken as the reciprocal of the Euclidean distance (Gambardella and Dorigo, 1995).

Three different rules were used to choose the random variable *S*: i) *pseudo-random*, where *S* is a state selected at random from the set $N_k(t)$, following a uniform distribution; ii) *pseudo-random-proportional*, in which *S* is selected from the distribution given by Equation 2 and; iii) *random-proportional*, such that, if *q* had the value 0 in Equation 1, then the next state is drawn at random from the distribution given by Equation 2.

$$S = \left\{ \frac{[AQ(i,j)]^{\delta} \times [HE(i,j)]^{\beta}}{\sum_{a \in N_k(t)} [AQ(i,a)]^{\delta} \times [HE(i,a)]^{\beta}} \right. \tag{2}$$

Gambardella and Dorigo (1995) showed that a good rule for choosing actions with the *Ant-Q* algorithm is based on *pseudo-random-proportional*. The *AQ*(*i*,*j*) is then estimated by using the updating rule in Equation 3, similar to the *Q-learning* algorithm:

$$\begin{aligned} AQ(i,j) &= (1-\alpha) \times AQ(i,j) \ + \\ &\alpha\left( \Delta AQ(i,j) + \gamma \times \max_{a \in S(j)} AQ(j,a) \right) \end{aligned} \tag{3}$$

where the parameters γ and α are the discount factor and learning rate respectively.

When the updating rule is local, the updated *AQ*(*i*,*j*) is applied after the state *s* has been selected, setting *ΔAQ*(*i*,*j*) to zero. The effect is that *AQ*(*i*,*j*) associated with the link (*i*,*j*) is reduced by a factor γ each time that this link appears in the candidate solution. As in the *Q-learning* algorithm, therefore, this approach tends to avoid exploration of states with lower probability (pheromone concentration), making the algorithm unsuitable for situations where the present solution must be altered significantly as a consequence of unexpected environmental change.

Other methods based on ant-colony behavior have been proposed for improving the efficiency of exploration algorithms in dynamic environments. Guntsch and Middendorf (2003) propounded a method for improving the solution when there are changes in environment, using local search procedures to find new solutions. Alternatively, altered states are eliminated from the solution, connecting the previous state and the successor to the excluded state. Thus, new states are brought into the solution. The new state is inserted at the position where the cost is least or where the highest cost in the environment is reduced, depending on the objective. Sim and Sun (2002) used multiple ant-colonies, such that one colony is repelled by the pheromone of the others, favoring exploration when the environment is altered. Other methods for dealing with a dynamic environment change the updating rule of the pheromone to enhance exploration. Li and Gong (2003), for example, modified local and global updating rules in the *Ant Colony System* algorithm. Their updating rule was altered as shown in Equation 4:

$$\tau_{ij}(t+1) = (1 - p(\tau_{ij}(t)))\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{4}$$

where $p_1(\tau_{ij})$ is a function of $\tau_{ij}$ at time *t*, with $\theta > 0$; for example:

$$p_1(\tau_{iJ}) = \frac{1}{1 + \mathrm{e}^{-(\tau_{iJ} + \theta)}} \tag{5}$$

with $\theta > 0$.

Such methods can be used as alternatives for finding solutions where the environment is changing. By using probabilistic transition rules, the ant-colony algorithm widens the exploration of the state-space. In this way a random transition decision is used and some parameters are modified, with new heuristic information influencing the selection of the more desirable links.

High pheromone values are reduced by introducing a dynamic evaporation process. Thus when the environment is altered and the solution is not optimal, pheromone concentration in the corresponding links is diminished over time. Global updating proceeds in the same way, except that only the best and worst global solutions are considered; i.e.:

$$\tau_{ij}(t+1) = (1-\rho_2(\tau_{ij}(t)))\tau_{ij}(t) + \gamma_{ij}\Delta\tau_{ij}(t) \tag{6}$$

where:

$$\gamma_{i,j} = \begin{cases} +1 & \textit{if } (i,j) \textit{ is the best global solution} \\ -1 & \textit{if } (i,j) \textit{ is the worst global solution} \\ 0 & \textit{otherwise} \end{cases} \tag{7}$$

A similar global updating rule was used by Lee et al. (2011). Other strategies for changing the pheromone value have been proposed to compensate the occurrence of stagnation in ant-colony algorithms. Gambardella et al. (1997) proposed a method for re-adjusting pheromone values with the values initially distributed. In another strategy, Stutzle and Hoos (1997) suggested proportionally increasing the pheromone value according to the difference between it and its maximum value.

Thus, a number of methods based on ant-colony algorithms have been developed for improving efficiency of algorithm exploration in dynamic environments. They can be used as alternatives for improving the solution when the environment is changed. The proposed approaches are based on procedures that use strategies to improve exploration using the probabilistic transition of the *Ant Colony System* algorithm to widen exploration of the state space. Thus, the most random transition decision is used, varying some parameters where the new heuristic information influences the selection of the more desirable links.

Some papers apply updating rules to links of a solution, including an evaporation component similar to the updating rule of *Ant Colony System*. Thus the pheromone concentration diminishes through time, with the result that less favorable states are less likely to be explored in future episodes. For this purpose, one alternative would be to re-initialize the pheromone value after observing the changes to the environment, maintaining a reference to the best solutions found. If the altered region of the environment is identified, the pheromone of adjacent states is re-initialized, making them more attractive. If a state is unsatisfactory, rewards can be made smaller (generally proportional to the quality of the solution), thus becoming less attractive over time because of loss of pheromone by evaporation.

It can be seen that most of the works mentioned concentrate their efforts on improving transition rules using sophisticated strategies to obtain convergence. However, experimental results shown that such methods do not yield satisfactory results in environments that are highly dynamic and where the magnitude of the space to be searched is not known. In Section 3 we present strategies developed for updating policies generated by rewards (pheromones) in dynamic environments.

In these problems, it is only possible to find the optimum solution if the state space is explored completely, so that the computational cost increases exponentially as the state-space increases.

## 3 STRATEGIES FOR UPDATING POLICIES GENERATED BY ALGORITHMS WHICH SIMULATE ANT-COLONIES

In Ribeiro and Enembreck (2010) it was found that algorithms based on rewards are efficient when the learning parameters are satisfactorily estimated and when modifications to the environment do not occur which might change the optimal policy. An action policy is a function mapping states to actions by estimating a probability that a state *e'* can be reached after taking action *a* in state *e*. In dynamic environments, however, there is no guarantee that the *Ant-Q* algorithm will converge to an acceptable policy. Before setting out the strategies for updating policies, we give a summary of the field of application, using a combinatorial optimization problem frequently used in computation to demonstrate problems that are difficult to solve: namely the Traveling Salesman Problem (TSP). In general terms, the TSP is defined as a closed graph $A=(E,L)$ (representing the agent's environment) with $n$ states (vertices) $E=\{e_1,...,e_n\}$, in which $L$ is the set of all linkages (edges) between pairs of states $i$ and $j$, where $l_{ij} = l_{ji}$ under symmetry. The goal is to find the shortest *Hamiltonian* cycle which visits every state, returning to the point of origin (Schrijver 2003). It is typically assumed that the distance function is a metric (e.g., Euclidean distance).

One approach to solving the TSP is to test all possible permutations, using exhaustive search to find the shortest Hamiltonian cycle. However given that the number of permutations is $(n - 1)!$, this approach becomes impracticable in the majority of cases. Unlike such exhaustive methods, heuristic algorithms such as *Ant-Q* therefore seek feasible solutions in less computing time. Even without guaranteeing the best solution (the optimal policy), the computational gain is favorable to finding an acceptable solution.

The convergence of ant-colony algorithms would occur if there were exhaustive exploration of the state space, but this would require a very lengthy learning process before convergence was achieved. In addition, agents in dynamic environments can adopt policies which delay the learning process or which generate sub-optimal policies. Even so, the acceleration to convergence of swarm-based algorithms can be accelerated by using adaptive policies which avoid unsatisfactory updating. The following paragraphs therefore set out strategies for estimating current policy which improve convergence of agents under conditions of environmental change.

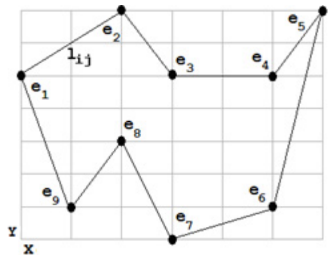
These strategies change pheromone values so as to improve coordination between agents and to allow convergence even when there changes in the Cartesian position of environmental states. The objective of the strategies is to find the optimum equilibrium of policy reformulation which allows new solutions to be explored using the information from past policies. Giving a new equilibrium to the pheromone value is equivalent to adjusting information in the linkages, giving flexibility to the search procedure which enables it to find a new solution when the environment changes, thereby modifying the influence of past policies to construct new solutions.

One updating strategy that has been developed is inspired by the approaches set out in (Guntsch and Middendorf, 2001) and (Lee et al., 2001), in which pheromone values are reset locally when environmental changes have been identified. This is termed the *global mean* strategy. It allocates the mean of all pheromone values of the best policy to all adjacent linkages in the altered states. The *global mean* strategy is limited because it fails to take in account the intensity of environmental change. For example, good solutions when states are altered can often make the solution less acceptable since it is only necessary to update part of action policy. The *global distance* strategy updates the pheromone concentrations of states by comparing the Euclidean distances between all states, before and after the environmental change. If the cost of the policy increases with increasing rate of environmental change, the pheromone value is decreased proportionately; otherwise it is increased. The *local distance* strategy is similar to the *global distance* strategy, but updating the pheromone is proportional to the difference in Euclidean distance of states that were altered.

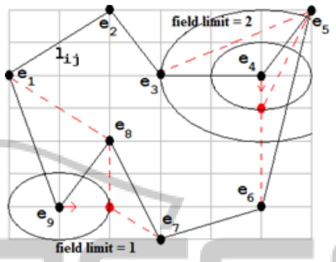
Before discussing how strategies allocate values to the current policy in greater detail, we discuss how environmental changes are occurring. Environmental states can be altered by factors such as scarcity of resources, change in objectives or in the nature of tasks, such that states can be inserted, excluded, or simply moved within the environment. Such characteristics are found in many different applications such as traffic management, sensor networks, management of supply chains, and mobile communication networks.

Figure 1 gives a simplified representation of a scenario with 9 states in a Cartesian plane. Figurea shows the scenario before alteration; Figureb shows the scenario after altering positions of states. The configuration of the scenario is shown in Table 1.

It can be seen that altering the positions of states $e_4$ and $e_9$ will add six new linkages to the current policy. The changes to the environment were made arbitrarily by altering the Cartesian positions of states whilst restricting them to lie within the field limit: i.e., adjacent to a Cartesian position. Field limit is used to restrict changes in addition to adjacent states.



(a) Position of states before alteration (A)



(b) Position of states after alteration (A')

Figure 1: Changing environmental states.

Table 1: Linkages between states before and after alterations.

| **Before alterations (A)** | | **After alterations (A')** | |
|---|---|---|---|
| states | Linkages | states | linkages |
| $e_1(0,5)$ | 1→2, 1→9 | $e_1(0,5)$ | 1→2, **1→8** |
| $e_2(2,7)$ | 2→3, 2→1 | $e_2(2,7)$ | 2→3, 2→1 |
| $e_3(3,5)$ | 3→2, 3→4 | $e_3(3,5)$ | **3→5**, 3→2 |
| $e_4(5,5)$ | 4→3, 4→5 | **$e_4$(5,4)** | **4→6**, 4→5 |
| $e_5(6,7)$ | 5→4, 5→6 | $e_5(6,7)$ | 5→4, 5→3 |
| $e_6(5,1)$ | 6→5, 6→7 | $e_6(5,1)$ | **6→4**, 6→7 |
| $e_7(3,0)$ | 7→6, 7→8 | $e_7(3,0)$ | **7→9**, 7→6 |
| $e_8(2,3)$ | 8→9, 8→7 | $e_8(2,3)$ | 8→9, **8→1** |
| $e_9(1,1)$ | 9→1, 9→8 | **$e_9$(2,1)** | **9→7**, 9→8 |

Thus, introducing environmental change can modify the position of a state, which can introduce differences between the current and the optimal policies giving rise, temporarily, to undesirable policies and errors. The strategies must update the pheromone values of linkages between the altered states, according to the characteristics of each.

### *A. Mean Global Strategy*

The *mean global* strategy takes no account of the intensity of environmental change, whilst detecting that states have been altered. The mean pheromone value of all linkages to the current best policy Q is attributed to linkages to the modified states. In contrast to other reports where the pheromone was reinitiated without taking account of the value learned, the *mean global* strategy re-uses values from past policies to estimate updated values. Equation 8 shows how the values are computed for this strategy:

$$mean\_global = \frac{\sum_{l \in Q} AQ(l)}{n_l} \tag{8}$$

where $n_l$ is the number of linkages and $AQ(l)$ is the pheromone value of the $l$ linkages.

***B. Global Distance Strategy***

The *global distance* strategy calculates the distance between all states and the result is compared with the distance between states in the modified environment. This strategy therefore takes into account the total intensity of environmental change. If the distance between states increases, the updated pheromone value is inversely proportional to this distance. If the cost of the distance between states is reduced, the pheromone value is increased by the same proportion. Equation 9 is used to estimate the updated values for linkages between states in the modified environment A'.

$$global\_distance = \frac{\sum_{i=1}^{n_e} \sum_{j=i+1}^{n_e} d_A(l_{ij})}{\sum_{i=1}^{n_e} \sum_{j=i+1}^{n_e} d_{A'}(l_{ij})} \times AQ(l_{ij}) \tag{9}$$

where ne is the number of states, A' is the environment after change and d is the Euclidean distance between the states.

***C. Local Distance Strategy***

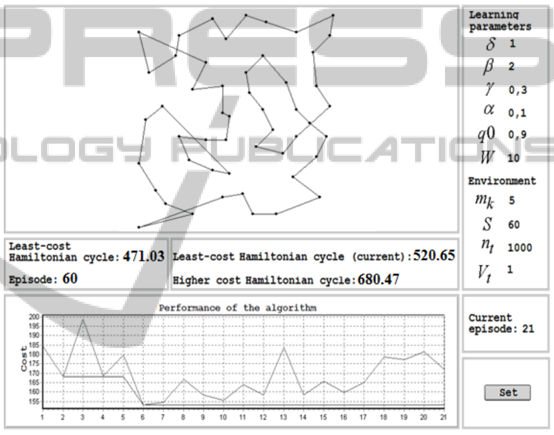
The *local distance* strategy is similar to the *global distance* strategy, except that only the pheromone of linkages to the modified states is updated. Each linkage is updated in proportion to the distance to adjacent states that were modified so that updating is localized in this strategy, thereby improving convergence when there are few changes to the environment. Equation 10 is used to compute updated values for the linkages:

$$local\_distance = \frac{d_A(l_{ij})}{d_{A'}(l_{ij})} \times AQ(l_{ij}) \tag{10}$$

The next sub-section gives the *framework* and the algorithm for the strategies mentioned above.

## 3.1 FANTS - *Framework for Ants*

FANTS was developed to simulate the *Ant-Q* algorithm to include the strategies outlined above. Figure 2 gives an overall picture of FANTS and its main components. The figure shows a graph in which the thicker, bolder line represents the best policy (i.e., shortest Hamiltonian cycle in $t_i$) of the episode $t_i$ as revealed by the *Ant-Q* algorithm. The graph immediately below shows the algorithm's convergence from one episode to those following it. An episode $t$ corresponds to a sequence of actions which determines the states visited by the agents. An episode $t_i$ ends when agents return to their original state after visiting all the others. The dimension Y of the graph is the cost of the policy in each episode (i.e., the cost of a Hamiltonian cycle). Also the dimension X corresponds to the number of episodes. The line in the graph which varies most gives the least-cost Hamiltonian cycle in each episode *t*.


Learning parameters
$\delta$ 1
$\beta$ 2
$\gamma$ 0,3
$\alpha$ 0,1
$q0$ 0,9
$W$ 10
Environment
$m_k$ 5
$S$ 60
$n_t$ 1000
$V_t$ 1

Least-cost Hamiltonian cycle: 471.03
Episode: 60
Least-cost Hamiltonian cycle (current): 520.65
Higher cost Hamiltonian cycle: 680.47

Current episode: 21

Figure 2: FANTS.

The columns to the right of Figure 2 show the parameters used by the algorithm and environment, where $\delta$ and $\beta$ are the parameters of the transition rule, and $\gamma$ and $\alpha$ are the algorithm's learning parameters. The variables $m_k$, $S$ and $t$ are the number of agents, the number of states and the number of episodes respectively. The parameter $t$ is used as a stopping criterion for the algorithm. The internal structures of the *framework* are expressed by equations 8-13 which make up the algorithm *Ant-Q* presented as Pseudocode 1.

The initial pheromone value is calculated from Equation 11:

$$\frac{1}{avg \times n} \tag{11}$$

$$d_{(ij)} = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \tag{12}$$

where *avg* is the average of the Euclidean distances between state pairs (*i,j*) calculated from Equation 12, and *n* is the number of agents in the system. Having calculated the pheromone initial value, this value is attributed to the linkages which constitute the graph. This procedure is used only before starting the first episode, allowing the agents to select states using both the pheromone values and the heuristic values.

An important aspect of the algorithm in Pseudocode 1 is the method for updating the learning table, which can occur either globally or locally. Global updating occurs at the end of each episode, when the least-cost policy is identified and the state values are updated using the reward parameter.

```
Algorithm FANT()
Require:
Learning table AQ(i,j);
Environment E;
#Changes, $t_w$ = 100;
Number of agents $m_k$;
Number of states S;
Number of episodes $t_n$;
Learning parameters:{$\alpha,\beta,\gamma,q_0,\delta,W$};
Updating strategies = {mean_global,
global_distance,local_distance}

Ensure:
Randomize the states in E;
Use equation 11 to compute the
initial value of the pheromone
and assign it to AQ(i,j);
For all episode Do:
  Set the initial position of the
  agents in the states;
  While there are states to be
  visited Do: // lista tabu <> $\phi$
  For all agent Do:
   if (q(rand(0..1) <= $q_0$) Then
     Choose an action according
     to equation 1;
    Else
     Choose an action according
     to equation 2;
   end if
    Update AQ(i,j) using the rule
    in place upgrade Equation 3);
  end for
  end while
 Compute the cost of the best
 policy of the episode $t_x$;
 Compute the global update, using
 equations 3 and 13;
 If #changes are supposed to
 occur Then
 For all linkage (i,j) of altered
  states Do:
   Switch (strategy):
    Case mean_global strategy:
    value=strategyA();//equation 8
    Case global_distance strategy:
    value=strategyB();//equation 9
    Case local_distance strategy:
    value=strategyC();//equation 10
  end for
  end if
 For all linkage (i,j) incident
 to the altered state Do:
   AQ(i,j) = value;
 end for
 Otherwise continue()
 end for
 Return(.,.)
```

Pseudocode 1: FANTS algorithm with strategies.

Equation 13 is used to calculate the value of $\Delta AQ(i,j)$, the reward for global updating.

$$\Delta AQ(i,j) = \left\{ \frac{W}{L_{best}} \right. \qquad (13)$$

where $W$ is a parameterized variable with value 10 and $L_{\text{best}}$ is the total cost of the shortest Hamiltonian cycle in the current episode. Local updating occurs at agent action, the value of $\Delta AQ(i,j)$ being zero in this case.

## 4 EXPERIMENTAL RESULTS

Experiments are reported here which evaluate the strategies discussed in Section 3 and the effects of the learning parameters on *Ant-Q* performance. These experiments evaluate algorithm efficiency in terms of: (i) variations in learning rate; (ii) discount factor; (iii) exploration rate; (iv) transition rules; (v) number of agents in the system; and (vi) the proposed updating strategies. Results and discussions are given in sub-sections 4.1 and 4.2.

The experiments were run using benchmark: eil51 e eil76, found in the online library TSPLIB[1] (Reinelt, 1991). The datasets eil51 and eil76 have 51 and 76 states respectively and were constructed by Christofides and Eilon (1969). Such sets have im-

[1]www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

portant characteristics for simulating problems of combinatorial optimization, such as, for example, the number of states and the presence of neighboring states separated by similar distances. They were also used by Dorigo (1992), Gambardella and Dorigo (1995), Bianchi et al. (2002), and Ribeiro and Enembreck (2010). Figure 3 shows the distribution of states in a plane, using a 2D Euclidean coordinate system.

Learning by the algorithm in each set of instances was repeated 15 times, since it was found that doing experiments in one environment alone, using the same inputs, could result in variation between results computed by the algorithm. This occurs because agent actions are probabilistic and values generated during learning are stochastic variables. The action policy determined by an agent can therefore vary from one experiment to another. The efficiency presented in this section is therefore the mean of all experiments generated in each set of instances. This number of replications was enough to evaluate the algorithm's efficiency, since the quality of policies did not change significantly (± 2.4%).

The learning parameters were initially given the following values: $\delta$=1; $\beta$=2; $\gamma$=0.3; $\alpha$=0.1; $q_0$=0.9 and $W$=10. The number of agents in the environment is equal to the number of states. Stopping criteria were taken as 400 episodes ($t$=400). It should be noted that because of the number of states and the complexity of the problems, the number of episodes are not enough for the best policy to be determined. However the purpose of the experiments was to evaluate the effects of parameters on the algorithm *Ant-Q* and on the utility of the final solution from the strategies given in Section 3.

To evaluate the performance of a technique, a number of different measures could be used, such as time of execution, the number of episodes giving the best policy, or a consideration only of the best policies identified. To limit the number of experiments, the utility of policies found after a given number of episodes was used, taking the minimum-cost policy at the end of the learning phase.

Preliminary results discussed in subsection 4.1 are for the original version of the *Ant-Q* algorithm, whilst experiments with dynamic environments and updating strategies are given in subsection 4.2.

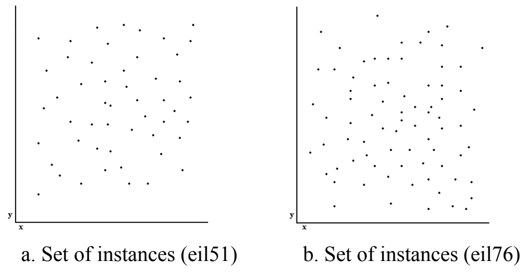


Figure 3: State space: Set of instances used in the simulations, with states given as points in a 2D Euclidean coordinate system.

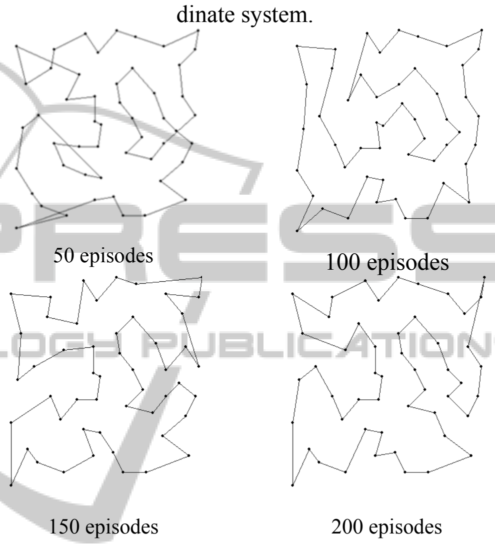


Figure 4: Policy evolution after each 50 episodes.

## 4.1 Preliminary Discussion of the Learning Parameters

Initial experiments were generated to evaluate the impact of the learning parameters and consequently were adjusted to the proposed strategies. Preliminary discussions are related in sub subsections A to E.

### A. Learning Rate

The learning rate α shows the importance of the pheromone value when a state has been selected. To find the best values for α, experiments were conducted in the set of instances for values of α between 0 and 1. Best results were found for α between 0.2 and 0.3. For larger values, agents tend to no longer make other searches to find lower-cost trajectories once they have established a good course of action in a given environmental state. For lower values, learning is not given the importance that it requires, so that agents tend to not select different paths from those in the current policy. The best α-value for policy was 0.2, and this was used in the other expe iments. It was also seen that the lower the rate of learning, the lower is the variation in policy.

**B. discount Factor**

The discount factor determines the time weight relative to the rewards received. The best values for the discount factor were between 0.2 and 0.3. Smaller values led to inefficient convergence, having little relevance to agent learning. Values greater than 0.3 the discount factor receives too much weight, leading agents to local optima.

**C. Exploration Rate**

The exploration rate, denoted by the parameter $q_0$, gives the probability that an agent selects a given state. Experiments showed that the best values lay between 0.8 and 1. As the parameter value approaches zero, agent actions become increasingly random, leading to unsatisfactory solutions.

The best value found for $q_0$ was 0.9. Agents then selected leading to lower-cost trajectories and higher pheromone concentrations. With $q_0$= 0.9 the probability of choosing linkages with lower pheromone values was 10%.

**D. Transition Rule**

The factors $\delta$ and $\beta$ measure the importance of the pheromone and of the heuristic (distance) when choosing a state. The influence of the heuristic parameter $\beta$ is evident. To achieve best results, the value of $\beta$ must be at least 60% lower than the value of $\delta$.

**E. Number of Agents**

To evaluate the effect of number of agents in the system, 26 to 101 agents were used. The best policies were found when the number of states is equal to the number of agents in the system. When the number of agents exceeded the number of states, good solutions were not found resulting in stagnation. Thus, having found a solution, agents tend to cease to look at other states, having found a local maximum. When the number of agents is lower than the number of states, the number of episodes must increase exponentially in order to achieve better results.

## 4.2 Performance of Agents with Updating Strategies

To evaluate the strategies set out in Section 3, dynamic environments were generated in the set of instances eil76. Agent performance was evaluated in terms of the percentage change (percent of changes (10% and 20%) in environment for a window *$t_w$=100*) generated in the environment after each 100 episodes. This time window ($t_w$=100) was used because past studies have shown that the algorithm converged well in environments in around 70 states (Ribeiro and Enembreck 2010).

Change was introduced as follows: at each 100 episodes, the environment produces a set of alterations. The changes were made arbitrarily in a way that simulated alterations in regions that were partially-known or subject to noise. Thus, environments with 51 states had 10 states altered when 20% change occurred. Moreover, alterations were then simulated for the space with limiting field of depth 1 and 2, so that change in state positions was restricted, thus simulating the gradual dynamically changing problems of the real world. Equation 14 is used to calculate the number of altered states in $t_w$=100.

$$\text{changes}_{t_w=100} = \frac{\#\text{states}}{100} \times \#\text{percent} \qquad (14)$$

The results of the experiments compare the three strategies with the policy found using the original *Ant-Q* algorithm. The learning parameters used in simulation were the best of those reported in subsection 4.1. In most cases, each strategy required a smaller number of episodes, since the combination of rewards led to better values by which agents reached convergence when policies were updated. Figures 5, 6, 7 and 8 show how the algorithm converged in the set of instances eil51. The X-axis in these figures shows the $t_i$ episodes; the Y-axis shows policy costs (Hamiltonian cycle as a percentage) obtained in each episode, which 100% refers to the best policy compute (optimal policy).

Figures 5, 6, 7 and 8 show that the global policy obtained when the strategies are used is better than that of the original *Ant-Q*. The *mean global* strategy is seen to be most adequate for environments where changes are greater (Figures 6 and 8). This is because this strategy uses all the reward values within the environment. However, agents reach convergence only slowly when the environment is little changed, since altered states will have lower rewards in their linkages than the linkages that define the current best solution. Nevertheless the *global distance* strategy was also more robust in environments with few changes (Figures 5 and 7). When the environment is altered, the strategy seeks to modify rewards in proportion to the amount of environmental change. Thus, the effect of updating reduces the impact resulting from change, causing agents to converge uniformly. The *local distance* strategy only takes account of local changes, so that updating of policies by means of this strategy works best when the reward values are larger, as in later episodes.

In general, the strategies succeed in improving policy using fewer episodes. They update global policy, and accumulate good reward values, when the number of episodes is sufficiently large. When learning begins, policy is less sensitive to the strategies, so that policy performance is improved after updating. Some strategies can estimate values that are inappropriate for current policy, mainly after many episodes and environmental changes result in local maxima.

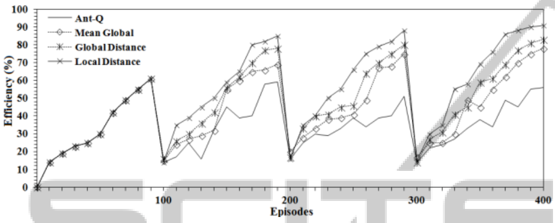


Figure 5: Limiting field = **1**; Change = 10%.

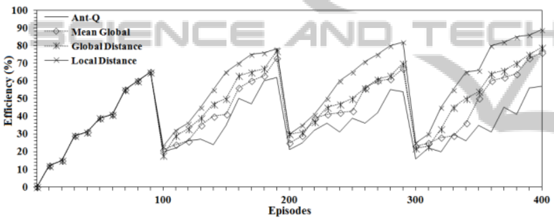


Figure 6: Limiting field = **1**; Change = 20%.

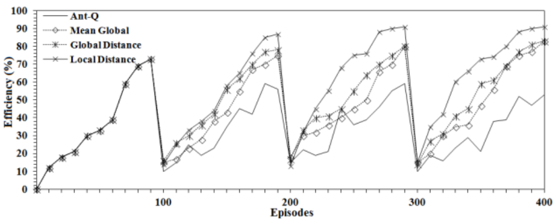


Figure 7: Limiting field = **2**; Change = 10%.

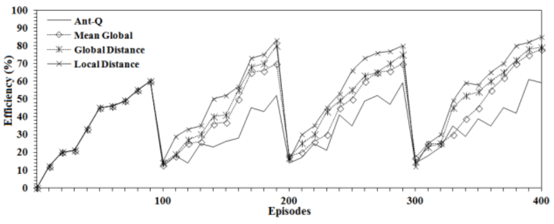


Figure 8: Limiting field = **2**; Change = 20%.

One point concerns the effect of the limiting field (adjacent to the Cartesian position) on strategies. Even with the limiting field restricted, the strategies improve the algorithm's convergence. In other experiments where the limiting field was set to 5, the efficiency of the *Ant-Q* algorithm is lower (19%) when compared with the best strategy (Figures 9 and 10).

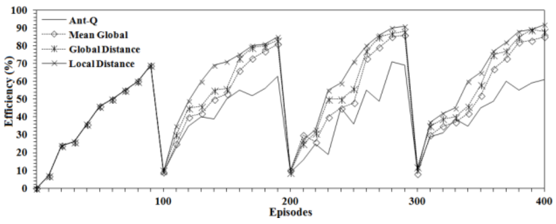


Figure 9: Limiting field = **5**; Change = 10%.

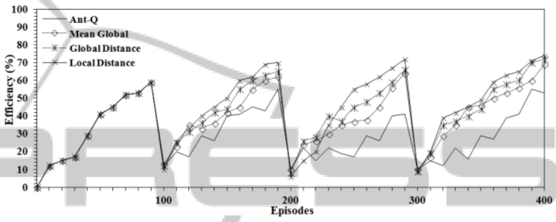


Figure 10: Limiting field = **5**; Change = 20%.

The *mean global* strategy is better when the limiting field is less than 5 (as in Figures 5 to 8). Since updating uses the mean of all pheromone values, the value for linkages between altered states is the same. The *global distance* and *local distance* strategies converge rapidly when the limiting field is 5 (Figures 9 and 10). This is because updating is proportional to the length of each linkage connected to an altered state. Thus linkages which are not part of the best policy have their pheromone values reduced.

We also generate experiments in others environments of different dimensions, 35, 45 and 55 states. Note that a number of states *S* can generate a long solution space, in which the number of possible policy is $|A||^{s}|$. The quality of policies in such environments did not change significantly (± 1.9%) and the efficiency of best strategy compared with the results of the set of instances eil76 is lower (14%).

## 5 CONCLUSIONS AND DISCUSSIONS

Methods for coordination based on learning by rewards have been the subject of recent research by a number of researchers, who have reported various applications using intelligent agents (Ribeiro et al., 2008), (Tesauro, 1995) and (Watkins and Dayan, 1992). In this scheme, learning occurs by trial and error when an agent interacts with the surrounding environment, or with its neighbors. The source of learning is the agent's own experience, which contributes to defining a policy of action which maximizes overall performance.

Adequate coordination between agents that use learning algorithms depends on the values of fitted parameters if best solutions are to be found. Swarm-based optimization techniques therefore use rewards (pheromone) that influence how agents behave, generating policies that improve coordination and the system's global behavior.

Applying learning agents to the problem of coordinating multi-agent systems is being used more and more frequently. This is because it is generally necessary for models of coordination to adapt in complex problems, eliminating and/or reducing deficiencies in traditional coordinating mechanisms (Enembreck et al., 2009). For this purpose the paper has presented FANTS, a solution-generating test *framework* for analysing performance of agents with the algorithm *Ant-Q* and for describing how *Ant-Q* behaves in different scenarios, and with different parameters and updating strategies of policies in dynamic environments. The *framework* presented is capable of demonstrating interactively the effects of varying parameter values and the number of agents, helping to identify appropriate parameter values for *Ant-Q* as well as the strategies that lead to solution.

Results obtained when the updating strategies for policies in dynamic environments are used show that performance of the *Ant-Q* algorithm is superior to its performance at discovering best global policy in the absence of such strategies. Although individual characteristics vary from one strategy to another, the agents succeed in improving policy through global and local updating, confirming that the strategies can be used where environments are changing over time.

Experiments using the proposed strategies show that, although their computational cost is greater, their results are satisfactory because better solutions are found in a smaller number of episodes. However further experiments are needed to answer questions that remain open. For example, coordination could be achieved using only the more significant parameters. A heuristic function could be used to accelerate *Ant-Q*, to indicate the choice of action taken and to limit the space searched within the system. Updating the policy could be achieved by using other coordination procedures, avoiding stagnation and local maxima. Some of these strategies are found in (Ribeiro et al., 2008) and (Ribeiro et al., 2012). A further question is concerned with evaluating the algorithm under scenarios with more states and other characteristics. These hypotheses and issues will be explored in future research.

## ACKNOWLEDGEMENTS

We thank anonymous reviewers for their comments. This research is supported by the Program for Research Support of UTFPR - campus Pato Branco, DIRPPG (Directorate of Research and Post-Graduation) and *Fundação Araucária* (Araucaria Foundation of Parana State).

## REFERENCES

Chaharsooghi, S. K., Heydari, J., Zegordi, S. H., 2008. *A reinforcement learning model for supply chain ordering management: An application to the beer game*. Journal Decision Support Systems. Vol. 45 Issue 4, pp. 949-959.

Dorigo, M., 1992. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Itália.

Dorigo, M., Gambardella, L. M., 1996. *A Study of Some Properties of Ant-Q*. In Proceedings of PPSN Fourth International Conference on Parallel Problem solving From Nature, pp. 656-665.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. *Ant System: Optimization by a Colony of Cooperting Agents*. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41.

Enembreck, F., Ávila, B. C., Scalabrin, E. E., Barthes, J. P., 2009. *Distributed Constraint Optimization for Scheduling in CSCWD*. In: Int. Conf. on Computer Supported Cooperative Work in Design, Santiago, v. 1. pp. 252-257.

Gambardella, L. M., Dorigo, M., 1995. *Ant-Q: A Reinforcement Learning Approach to the TSP*. In proc. of ML-95, Twelfth Int. Conf. on Machine Learning, p. 252-260.

Gambardella, L. M., Taillard, E. D., Dorigo, M., 1997. *Ant Colonies for the QAP*. Technical report, IDSIA, Lugano, Switzerland.

Guntsch, M., Middendorf, M., 2001. *Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP*. In Proc. of the Workshop on Applications of Evolutionary Computing, pp. 213-222.

Guntsch, M., Middendorf, M., 2003. *Applying Population Based ACO to Dynamic Optimization Problems*. In Proc. of Third Int. Workshop ANTS, pp. 111-122.

Kennedy, J., Eberhart, R. C., Shi, Y., 2001. *Swarm Intelligence*. Morgan Kaufmann/Academic Press.

Lee, S. G., Jung, T. U., Chung, T. C., 2001. *Improved Ant Agents System by the Dynamic Parameter Decision*. In Proc. of the IEEE Int. Conf. on Fuzzy Systems, pp. 666-669.

Li, Y., Gong, S., 2003. *Dynamic Ant Colony Optimization for TSP*. International Journal of Advanced Manufacturing Technology, 22(7-8):528-533.

Mihaylov, M., Tuyls, K., Nowé, A., 2009. *Decentralized Learning in Wireless Sensor Networks*. Proc. of the Second international conference on Adaptive and

Learning Agents (ALA'09), Hungary, pp. 60-73.

Reinelt, G., 1991. *TSPLIB - A traveling salesman problem library*. ORSA Journal on Computing, 3, 376 - 384, 1991.

Ribeiro, R., Enembreck, F., 2012. *A Sociologically Inspired Heuristic for Optimization Algorithms: a case study on Ant Systems*. Expert Systems with Applications. Expert Systems with Applications, v.40, Issue 5, pp. 1814-1826.

Ribeiro, R., Favarim F., Barbosa, M. A. C., Borges, A. P, Dordal, B. O., Koerich, A. L., Enembreck, F., 2012. *Unified algorithm to improve reinforcement learning in dynamic environments: An Instance-Based Approach*. In 14th International Conference on Enterprise Information Systems (ICEIS'12), Wroclaw, Poland, pp. 229-238.

Ribeiro, R., Enembreck, F., 2010. *Análise da Teoria das Redes Sociais em Técnicas de Otimização e Aprendizagem Multiagente Baseadas em Recompensas*. Post-Graduate Program on Informatics (PPGIa), Pontifical Catholic University of Paraná (PUCPR), Doctoral Thesis, Curitiba - Pr.

Ribeiro, R., Borges, A. P., Enembreck, F., 2008. *Interaction Models for Multiagent Reinforcement Learning*. Int. Conf. on Computational Intelligence for Modelling Control and Automation - CIMCA08, Vienna, Austria, pp. 1-6.

Schrijver, A., 2003. *Combinatorial Optimization*. volume 2 of Algorithms and Combinatorics. Springer.

Sim, K. M., Sun, W. H., 2002. *Multiple Ant-Colony Optimization for Network Routing*. In Proc. of the First Int. Symposium on Cyber Worlds, pp. 277-281.

Stutzle, T., Hoos, H., 1997. *MAX-MIN Ant System and Local Search for The Traveling Salesman Problem*. In Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 309-314.

Sudholt, D., 2011. *Theory of swarm intelligence*. Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO '11). ACM New York, NY, USA, pp. 1381-1410.

Tesauro, G., 1995. *Temporal difference learning and TD-Gammon*. Communications of the ACM, vol. 38 (3), pp. 58-68.

Watkins, C. J. C. H., Dayan, P., 1992. *Q-Learning*. Machine Learning, vol.8(3), pp.279-292.

Wooldridge, M. J., 2002. *An Introduction to MultiAgent Systems*. John Wiley and Sons.