Intelligent Automation and Soft Computing, Vol. 17, No. 1, pp. 41-59, 2011

META-TACS: A TRUST MODEL DEMONSTRATION OF ROBUSTNESS THROUGH A GENETIC ALGORITHM

FÉLIX GÓMEZ MÁRMOL *, GREGORIO MARTÍNEZ PÉREZ AND JAVIER G. MARÍN-BLÁZQUEZ

Departamento de Ingeniería de la Información y las Comunicaciones
Facultad de Informática, Campus de Espinardo, s/n
University of Murcia, 30.071 Murcia, Spain
{felixgm, gregorio, jgmarin}@um.es

ABSTRACT—Ensuring trust and confidence in virtual communities' transactions is a critical issue nowadays. But even more important can become the use of robust and accurate trust models allowing an entity to decide which other entity to interact with. This paper aims to study the robustness of TACS (Trust Ant Colony System), a previously proposed bio-inspired P2P trust model, when applying a genetic algorithm in order to find the range of values of its working parameters that provides the best TACS performance. The optimization of those parameters has been carried out using the CHC genetic algorithm. Experiments seems to demonstrate that TACS can achieve high performance ratios due to the enhancement provided by META-TACS, and to achieve them for a wide range of working parameters, hence showing a remarkable robustness.

Key Words: Robust Trust Model, CHC Genetic Algorithm, TACS, Trust Ant Colony System, Trust management, P2P Networks, Meta-heuristics

1. INTRODUCTION

Different research works have been done so far in order to improve security in P2P networks by ensuring a minimum level of confidence between every two interacting nodes. However, it is not definitively solved since none of the studied models has become a de facto standard in this field.

Moreover, there is a wide variety of trust and reputation models depending on their scope (P2P, Ad-hoc, Wireless Sensor Networks, multi-agent systems, etc.), their type (trust model, reputation one or even both), and the technique they use (fuzzy logic, Bayesian networks, etc).

Nevertheless, they are mainly focused on the way the trust and/or reputation values are computed and they do not manage neither how the node to have a transaction with is finally selected, nor which is the path leading to that certain node, nor how robust the model is.

In this paper we will first review a resilient trust model for P2P networks [24, 25] where some nodes offer some services or goods and other ones are requesting those services. The former will be always looking for the best self profit, while the latter will be demanding the best services with respect to some quality characteristics, such as the prize or delay, for instance.

* Corresponding author; phone number: +34 868 887866; fax number: +34 868 884151

The main feature of our model is the use of a bio-inspired Ant Colony System (ACS) algorithm [1] allowing to determine the route (sequence of nodes) leading to the most trustworthy server offering a particular service all over the network.

Ant Colony Optimization (ACO) [1, 2, 22, 23] is widely accepted as one of the most promising soft-computing algorithms when solving some kinds of optimization problems such as the traveling salesman problem.

In this type of optimization algorithm ants travel along the network searching the optimum path fulfilling certain conditions (for instance, the optimum path leading to a node offering certain service). While they are traveling they leave a trace of pheromone, called τ, which is used to guide other ants (the more pheromone trace a path has, the more probability it has to be chosen). Finally, the path with the strongest trace of pheromone will be selected as the optimum one.

But the main objective of this work is to propose and study the behavior of META-TACS, an optimizer of the working parameters of TACS via evolutionary computation. In previous work [24,25] TACS was proposed as a bio-inspired P2P trust model including some preliminary results. In the present work, a CHC (Cross generational elitist selection, Heterogeneous recombination, Cataclysmic mutation) genetic algorithm [26] was employed to implement META-TACS. META-TACS allows studying the performance and the robustness of TACS against a wide range of values for its working parameters. One of the results of the study was that TACS remains obtaining good outcomes almost regardless the combination of values of its working parameters, which suggests that it is both resilient and easily configurable. CHC was chosen because of its rather small population size (since evaluating a set of parameters, that is, an individual, is comparatively costly) while keeping a good balance between exploration and convergence. As we will see later, META-TACS provided us with a bounded range of possible values for each parameter of TACS. The long term objective was to provide TACS with mechanisms to self-adapt to the current network conditions, although the results show a remarkable resilience to diverse typical situations.

The rest of the paper is organized as follows: section 2 presents a set of related works in this area. In section 3 our model TACS is described. Then, section 4 presents a genetic algorithm designed to optimize TACS parameters, called META-TACS, whose experiments and results are described and analyzed in section 5. Finally, section 6 exposes some conclusions and future work.

2. RELATED WORK

A number of models have arisen in the field of trust and reputation recently. That is the case, for instance, of [3-8], where several trust and reputation models are proposed for a multi-agent system [9] in which agents interact (competition, coordination, cooperation...) in order to get the greatest self profit. In [10-16, 31], however, authors develop trust and reputation models mainly for P2P networks, although some of them could be used in a multi-agent system or even in an ad-hoc network. Ad-hoc and Wireless Sensor Network trust and reputation models are exposed, for example, in [17-19, 32].

Some of these models are based on fuzzy logic in order to represent trust and reputation values. Others rely on Bayesian networks and a posteriori probabilities. There are also models based on social networks. And some others just give analytic expressions to compute trust and reputation.

Among all the studied and analyzed works, we have just found two models where the bio-inspired ant colony optimization is used. They are [20, 21], but none of them face the problem as we do. AntRep [20], for instance, uses the ACS in order to distribute reputation information, while TDTM [21] requires the existence of a Public Key Infrastructure in the network. In our

opinion this can become a strong restriction since not all the P2P networks will be composed of devices with enough capabilities to support such cryptographic infrastructure.

All the cited models just give a trust and/or reputation value for each entity in the network, but any of them tells how to reach that entity. That is, any of them provide the path to follow in order to go to that certain node.

Many works have been done in order to formalize trust and categorize trust models [30] but, as far as we know, this is one of the first papers where a trust model optimization is presented. In this way, we decided to use a genetic algorithm (in particular, the CHC one, given its specific GA features) [26, 27] since it has been proved that evolutionary computation (EC) seems an intelligent option when dealing with optimization problems where there is not much knowledge available about the search space structure while, at the same time, EC is still able to obtain high quality solutions.

3. TACS OVERVIEW

TACS (Trust Ant Colony System) [24, 25] is a Trust model for P2P networks based on the bio-inspired algorithm ACS (Ant Colony System [1,2]) where the pheromone traces, $\tau \in [0,1]$, are identified with the confidence a client requesting a certain service has on finding a trustworthy server through a specific route. The heuristic component $\eta \in [0,1]$ (also involved in the decision of which path to choose) is identified with the similarity between the service requested by the client and the service actually offered by a certain server. And if a server does not offer the requested service then η is defined as the goodness of that server acting as a relay node.

In summary, the steps that compose this model are the following, as it is shown in Figure 1:

1. Client *C* executes TACS in order to find the "optimum"" server *S* all over the network offering the desired service *s*
2. TACS launches the ACS algorithm and ants modify the pheromone traces of the network
3. TACS finishes, having selected the "optimum" path to server *S′*
4. TACS informs the client *C* that the "optimum" server found is *S′*
5. Client *C* requests the desired service *s* to the server *S′*

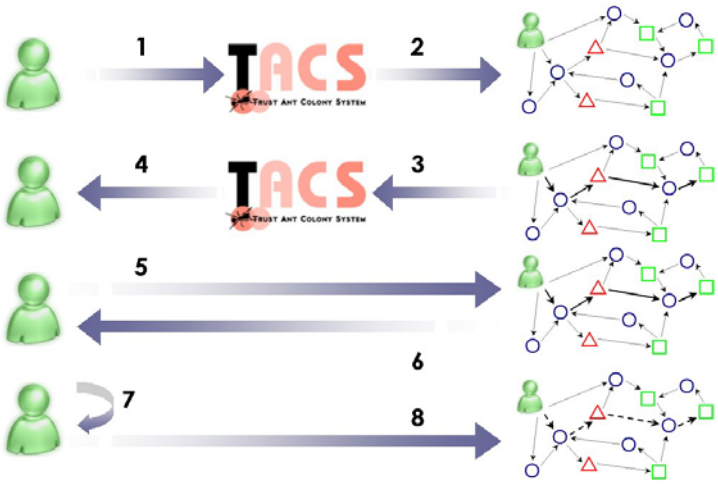


Figure 1. TACS Model Steps.

6. Server S' provides service s' to the client C
7. Client C evaluates his satisfaction with the received service s'
8. If client C is not satisfied with the received service s', he punishes the server S' evaporating the pheromone of the path that leads from C to S'. Otherwise no punishment is carried out

At the beginning of the process all the pheromone traces are initiated according to this formula:

$$\tau(0) = IniPh + (2 \cdot r - 1.0) \cdot IniPh \cdot (1.0 - IniPh) \tag{1}$$

where $IniPh \in [0,1]$ is a parameter indicating the initial pheromone value desired and $r \sim [0,1]$ is a random number within the interval $[0,1]$.

While an ant builds the path leading to the most trustworthy server, it modifies the pheromone traces along it. This pheromone local updating is carried out through the following expression:

$$\tau_{cs}(t) = (1-\varphi) \cdot \tau_{cs}(t-1) + \varphi \cdot z_1 \cdot \tau_{cs}(t-1) \tag{2}$$

where $\tau_{cs}(t)$ is the pheromone value of the edge e_{cs} (linking node c and s) at time t, $\varphi \in [0,1]$ is a constant called *phi* and z_1 is defined as follows:

$$z_1 = 1 + (1-\varphi) \cdot (1 - \tau_{cs}(t-1) \cdot \eta_{cs}(t-1))$$

What implies an increase of pheromone above the previous value, but never higher than a 100%. Moreover, with the term $(1 - \tau_{cs}(t-1) \cdot \eta_{cs}(t-1))$ we achieve that edges with lower values of pheromone can recover faster (increasing more its traces) and those who have higher values increase themselves slower.

Moreover, when all the ants (which number depends on the size of the network) have built a path, and the best of these has been chosen, an additional pheromone global updating is done along that selected route as follows:

$$\tau_{cs}(t) = (1-\rho) \cdot \tau_{cs}(t-1) + \rho \cdot z_2 \cdot \tau_{cs}(t-1) \tag{3}$$

where $\rho \in [0,1]$ is a constant called *rho*, and z_2 is defined as:

$$z_2 = 1 + \tau_{cs}(t-1) \cdot \eta_{cs}(t-1) \cdot Q(S_{Better\_Global})$$

where $Q(S_{Better\_global})$ is the quality of the best path that all the ants have found in one single iteration of the algorithm. Therefore now those edges with a higher value of τ and η are more rewarded than those with lower values. Thus, both expressions (2) and (3) have been designed in order to establish a good balance between exploration and convergence, when searching the most trustworthy server.

The quality of the path built by ant k, $Q(S_k) \in [0,1]$, is computed as follows:

$$Q(S_k) = \frac{A_k}{Length(S_k)^{PLF}} \cdot \overline{\tau}_k \tag{4}$$

where $A_k \in [0,1]$ is the ratio of ants that have selected the same path as the ant k, $Length(S_k)$ is the length of the solution S_k, $\bar{\tau}_k$ is the average pheromone of that path and $PLF \in [0,1]$ is a constant called *Path Length Factor*.

We think this is a reasonable good way of measuring the quality of a path (and the experiments have demonstrated it) since it takes into account three main factors: the ratio of ants who have chosen the same path, the average pheromone of that path and the length of the route. Having that expression we give a higher quality value to those paths which have been chosen by a larger number of ants, which have a greater average pheromone value and which are shorter in length.

When an ant k discovers a server offering the required service, it has to decide whether to stop and return the current path leading to that server, or keep on trying to find a better (more trustworthy) one. This decision could be expressed as the transition rule: if $\bar{\tau}_k > TraTh$ and $\bar{\tau}_k > r \sim [0,1]$ then ant k stops and returns current solution, where $\bar{\tau}_k$ is the average pheromone of the current path, $TraTh \in [0,1]$ is a constant called Transition Threshold and $r \sim [0,1]$ is a random number within the interval $[0,1]$.

And when an ant is currently in a server who does not offer the desired service, it has to move one step forward, choosing among the current node's neighbors. Let ant k be at node r, then the probability of choosing neighbor s as the next node in the path is computed as:

$$p_k(r,s) = \frac{[\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}{\sum_{u \in J_k(r)} [\tau_{ru}]^{\alpha} \cdot [\eta_{ru}]^{\beta}} \tag{5}$$

where $J_k(r)$ is the set of reachable nodes from *r* not visited yet by ant k and α, β are two weights establishing a balance between pheromone traces and heuristic values, respectively.
But specifically, the ACS adds a proportional probabilistic transition rule as follows:

$$p'_k(r,s) = \begin{cases} \arg\max_{u \in J_k(r)} [\tau_{ru}]^{\alpha} \cdot [\eta_{ru}]^{\beta} & \text{if } q \le q_0 \\ p_k(r,s) & \text{otherwise} \end{cases} \tag{6}$$

where $q \sim [0,1]$ is a random number within the interval $[0,1]$ and q_0 is the probability of choosing deterministically the most promising next node *u*.

As we have seen, at step 7 of TACS client C evaluates his satisfaction, $Sat \in [0,1]$, with the received service s'. To do so, he assesses the similarity between that service and the one initially requested: s. This assessment may be different for each client, since it depends on some weights (meaning client's preferences) that the own client can define by himself. It is modeled as follows:

$$Sat = Sim(s,s') = f(w_{prize}, w_{quality}, w_{delivery}) \tag{7}$$

Finally, the last step of the model consists of punishing a server in case he has provided a worse service than the one he initially offered. If $Sat \ge PunTh$, where $PunTh \in [0,1]$ is a constant called *Punishment Threshold*, then the punishment carried out is:

$$\tau_{cs} = \tau_{cs} - \varphi \cdot (1 - Sat) \cdot 2df_{cs} \tag{8}$$

where $df_{cs} \in [0,1]$ is a distance factor for edge e_{cs} that implies a stronger punishment for those edges closer to the malicious server and which is computed as follows:

$$df_{cs} = \sqrt{\frac{d_{cs}}{L \cdot (L - d_{cs} + 1)}}, \qquad d_{cs} = 1,2,\ldots,L-1$$

being L the actual length of the whole path and d_{cs} the distance of edge e_{cs} from the client.

Otherwise, if $Sat < PunTh$, the stronger punishment for e_{cs} is:

$$\tau_{cs} = \left(\frac{\tau_{cs}}{df_{cs}} - \varphi \right) \cdot Sat \tag{9}$$

The number of ants and iterations depends on the number of nodes of the network according to the expressions $N_A = N_N^{N_{ants}}$ and $N_I = N_N^{N_{iter}}$, respectively, where $N_{ants}, N_{iter} \in [0,1]$.

3.1 Advanced Features

Some features that can be derived from the mathematical model of TACS are now presented. This model allows the anonymity of the entities participating in the network since it does not need to associate an entity with its actual identity. Every entity is just given a pseudo-identifier per session not associated with its real identity. However, newcomers do not have more opportunities than non malicious remaining entities in the network. Otherwise an entity could achieve enough reputation to interact with other ones, then keep cheating until its level of reputation did not allow him to interact again (at least for awhile), and then leave and re-enter the network as a newcomer and start again. This is achieved in TACS by evaporating pheromone traces only when an unsuccessful interaction has been carried out.

Nonetheless, benevolent newcomers indeed have the ability to participate although there was a very trustworthy entity in the network, because those newcomers will receive gradually more ants depositing pheromone traces until they reach a certain level that allows them (in terms of probability) to be selected. Likewise, redemption of past malicious entities that has become benevolent is accepted. And an exploitation of a good built up reputation is avoided since the punishment for a high deception in a transaction is even higher (see equation (9)).

When evaluating the satisfaction perceived by a certain transaction, a subjectivity assessment is allowed since each client may define his own weights in order to compute the similarity between the requested service and the one actually received (as shown in equation (7)).

3.2 TACS Performance

The overhead added in this kind of networks is most of the times a critical issue since the interacting devices in a P2P, Ad-hoc or even Wireless Sensor Network usually have great constraints about memory, processing and communication capacity.

Therefore, we made some tests in order to measure the performance or throughput of our model. Our library [25] size is close to 31 Kbytes. Moreover, Figure 2 depicts the average time in milliseconds needed to perform one transaction (without taking into account transmission delays).

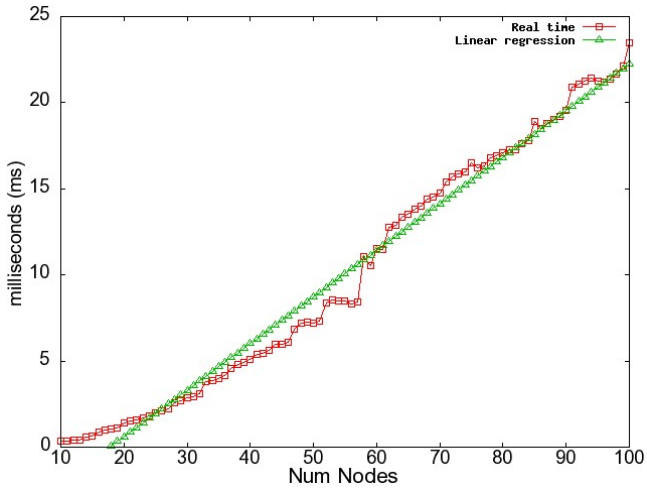


Figure 2. TACS Execution Time.

This graphic indicates the overhead introduced by TACS. It needs less than 25 milliseconds to be executed over a 100 nodes network and up to 0.33 ms when the network is only composed by 10 nodes. It can be checked that the performance nearly follows a linear function.

4. META-TACS

As we have seen, TACS includes several parameters involved in the model. And with the values shown in Table I we got good outcomes in comparison with other bio-inspired methods [24].

However, we thought that it would be interesting to have an optimization of these working parameters. Even more, it would be necessary to obtain knowledge about the behavior of TACS regarding the specific working parameters and trying to provide a more solid support for using a set of particular values, instead of only trying a few empirical values (as the ones shown in Table I, for instance).

That is the reason why we applied a genetic algorithm in order to optimize TACS parameters and give a formal support to those values. Our pursued intention was to check the behaviour of TACS model against a relatively wide range of values for each parameter (having in mind that every optimized parameter can take values within the interval $[0,1]$).

Genetic algorithms have been proved to be a good option when optimizing continuous variables [26], as it is our case. Specifically, the CHC algorithm [27], that stands for Cross generational elitist selection, Heterogeneous recombination, Cataclysmic mutation, was chosen. The most important reason as to choose CHC in particular in order to implement META-TACS lies in that it works with a small population size. For the present problem small population size is a convenient feature because evaluating an individual requires the execution of an instance of TACS. Although TACS is a remarkably fast algorithm such execution is comparatively costly compared with the other components of the CHC genetic engine. The other features found in CHC were designed to counterweight the weakness that small populations present. Elitist selection allows a monotonic improvement of the solution. A specially designed heterogeneous (and uniform) recombination, termed HUX, provides better sampling coverage when few individuals

Table I. TACS Parameters.

Parameter name	Value	Range	Meaning
φ	0.1	[0,1]	Pheromone local updating and punish and reward parameter
ρ	0.1	[0,1]	Pheromone global updating parameter
α	1.0	[0,1]	Learning weight in the transition rule
β	1.0	[0,1]	Heuristic weight in the transition rule
N_{ants}	0.35	[0,1]	Exponent to determine the number of ants
N_{iter}	0.35	[0,1]	Exponent to determine the number of iterations
q_0	0.98	[0,1]	Probability of choosing deterministically the most promising next node
IniPh	0.5	[0,1]	Initial pheromone trace
TraTh	0.5	[0,1]	Transition threshold, to determine if an ant must stop when it finds a node offering the requested service or not
PunTh	0.5	[0,1]	Punishment threshold, to determine if an edge must be punished or not
PLF	0.5	[0,1]	Path length factor, to determine the importance of the length of a path when measuring its quality

are available, aided by the incest prevention policy followed in CHC. With such aggressive crossover operator the typical mutation operator is not required. The cataclysmic mutation is performed when the expected premature convergence of small populations finally arrives. In all CHC is a well documented, good performance general purpose optimizer, with a small population size, and therefore seemed an adequate choice for META-TACS. Algorithm 1 shows its design.

In such algorithm L represents the size of an individual, M the number of individuals in a population and $r \in [0,1]$ a constant indicating the percentage of variation of the best individual when a re-initialization is carried out.

The hamming distance between two individuals ι^1 and ι^2 is computed as follows:

$$Hamming\_distance(\iota^1, \iota^2) = \sum_{i=1}^{L} hamming(\iota_i^1, \iota_i^2)$$

$$hamming(\iota_i^1, \iota_i^2) = \begin{cases} 0 & \text{if } |\iota_i^1 - \iota_i^2| < \delta \\ 1 & \text{otherwise} \end{cases}$$

where δ is a similarity threshold between the components of two individuals. Thus, if two individuals are very similar, they are not crossed, preventing this way the incest.

Moreover, instead of using the HUX cross, we implemented the BLX-α cross. That is, given two individuals $\iota^1 = (\iota_1^1, \iota_2^1, \ldots, \iota_L^1)$ and $\iota^2 = (\iota_1^2, \iota_2^2, \ldots, \iota_L^2)$, BLX-$\alpha$ cross generates another two individuals $\iota^k = (\iota_1^k, \iota_2^k, \ldots, \iota_L^k)$, where ι_i^k is randomly generated within the interval

$$\left[\iota_{\min} - I\alpha, \iota_{\max} + I\alpha\right]$$

being $\iota_{\min} = \min\{\iota_i^1, \iota_i^2\}$, $\iota_{\max} = \max\{\iota_i^1, \iota_i^2\}$, $\alpha \in [0,1]$ and $I = \iota_{\max} - \iota_{\min}$. The bigger α is the greater is the diversity when searching the best individual.

```
d = L / 4
initialize population P(t)
evaluate individuals in P(t)
while stop condition not satisfied do
  t = t + 1
  Copy all members of P(t-1) to C(t) at random
  /* HUX cross and incest prevention */
  for each of the M / 2 pairs of members in C(t)
    if (hamming_distance(i^i, i^j) / 2) > d
      swap half the differing bits at random
    else
      delete the pair of individuals from C(t)
  evaluate individuals in C'(t)
  /* Elitist selection */
  form P(t) from P(t-1) by replacing the worst members of P(t-1)
with the best members of C'(t)
  if P(t) equals P(t-1)
    d = d - 1
  /* Re-initialization */
  if d < 0
    replace P(t) with M copies of the best member of P(t-1)
    for all but one member of P(t)
      flip r × L bits at random
      evaluate individual
    d = r × (1 - r) × L
```

Algorithm 1. CHC algorithm.

In the re-initialization step, we did not use the method explained in algorithm 1 (i.e. flip $r \times L$ bits at random), since our individuals representation is not a bit string, but real parameters, that is $\iota_i \in \mathbf{R}$ (in fact $\iota_i \in [0,1]$). We used, therefore, a normal distribution with mean 0 and standard deviation σ, $\mathcal{N}(0, \sigma)$, in order to diverge and reinitialize the individuals of a population. Specifically, the divergence is carried out as follows:

$$\iota_i = \iota_i + \mathcal{N}(0, \sigma)$$

The bigger $\sigma \in [0,1]$ is, the greater the divergence of a member when re-initialization is done. We set σ to 0.1, which means that almost all possible divergence is inside the ±0.3 range.

Since we developed our customized genetic algorithm in order to optimize the parameters shown in Table I (except for alpha and beta), each individual is represented as a tuple:

$$\iota = (\varphi, \rho, q_0, IniPh, N_{ants}, N_{iter}, TraTh, PunTh, PLF)$$

Thus, evaluating a member means executing TACS with the parameters values specified by that certain member. And its fitness is the average satisfaction of the client with the received service minus half the standard deviation ($\mu_{Sat} - \sigma_{Sat} / 2$).

Finally, the stop condition for our CHC algorithm was:

$$fitness_{Best\_individual} > 1.0 - 10^{-6}$$

That value was considered that provided an acceptable while close-to-perfect result.

5. META-TACS EXPERIMENTS AND RESULTS

This section presents the whole set of experiments with their corresponding outcomes carried out in order to optimize TACS parameters through our customized CHC genetic algorithm.

As we have seen before, CHC algorithm has its own setting parameters. The values used for those parameters in all the experiments done can be observed in Table II.

Table II. META-TACS Parameters.

Parameter name	Value	Range	Meaning
M	40	[0,∞]	Number of individuals in a population
L	9	[0,∞]	Size of an individual
r	0.35	[0,1]	Percentage of variation of the best individual when a re-initialization is carried out
δ	0.01	[0,1]	Similarity threshold between the components of two individuals
α	0.3	[0,1]	Determines the diversity when crossing two individuals
σ	0.1	[0,1]	Divergence of individuals when a re-initialization is carried out

Here we defined the three same scenarios that were designed to test TACS alone [24, 25], that is, static networks, dynamic networks and oscillating ones. In fact, each individual fitness assessment requires an execution of TACS over 5000 random networks with 100 interactions for each of them (except for the oscillating scenario, where 1000 transactions are done, as we will explain later) using the parameters values contained in that specific individual.

5.1 Static Networks

Under these conditions, we launched one CHC genetic algorithm for each case of the first scenario. In other words, we launched a CHC algorithm for static networks with 10 to 20 nodes, one CHC algorithm for static networks with 20 to 30 nodes and so on (30 to 40 and 40 to 50).

Table III and Figure 3 show the mean, μ, the standard deviation, σ, the minimum and the maximum values for each one of the optimized parameters obtained within the first scenario.

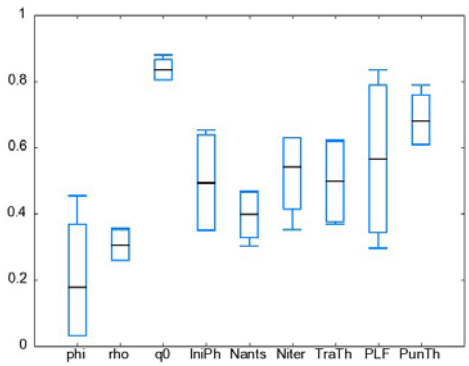
The first appreciation that can be done is that in a static environment φ can take a wide range of values, while ρ 's values fall into a smaller interval. φ is involved in the pheromone local updating (formulae (2)) and in the punishment method (formulae (8) and (9)), so the bigger φ is, the stronger the punishment is (and vice versa). However, the closer to 0.5 φ is, the greater is the pheromone local updating, and the closer to the extremes 0 or 1 φ is, the weaker is the pheromone local updating carried out.

ρ is only involved in the pheromone global updating, and a greater value of ρ implies a bigger updating (and vice versa). As it can be seen ρ is never greater than 0.5, its range of values is relatively small and on average, it is greater than φ.

About q_0, if it was equal to 0, every ant would use the transition rule described in equation (5) (direct exploration), otherwise, if $q_0 = 1$, the transition rule always used would be the one shown in equation (6) (probabilistic transition). In a static scenario, q_0 takes higher values with a low standard deviation.

Table III and Figure 3. META-TACS Outcomes for Static Networks.

	μ	σ	Max	Min
phi	0.1784	0.1903	0.4545	0.0319
rho	0.3045	0.0482	0.3576	0.2602
q0	0.8332	0.0321	0.8788	0.8057
IniPh	0.4929	0.1459	0.6514	0.3511
N_{ants}	0.3973	0.0689	0.4691	0.3032
N_{iter}	0.5404	0.1279	0.6301	0.3533
TraTh	0.4972	0.1214	0.6218	0.3689
PLF	0.5652	0.2226	0.8331	0.2961
PunTh	0.6806	0.0781	0.7896	0.6088
Fitness	0.9998	$9.7 \cdot 10^{-5}$	0.9999	0.9997



The most interesting thing to say about the $IniPh$ parameter is that its mean is close to 0.5 with a considerable standard deviation. This means that the best initial conditions for a client is neither to be very trustful, nor to be very untrustful, but a mean term with certain divergence.

In order to help understanding the meaning of the values of N_{ants} and N_{iter} (and even PLF), Figure 4 depicts the set of functions x^a where $a \in \{0.1, 0.2, \ldots, 0.9, 1.0\}$. As it can be observed, when the size of the network is less than 50 nodes, the difference between $N_N^{0.1}$ and $N_N^{0.5}$ is minimal. The difference begins to be relevant when N_{ants} or N_{iter} is greater than 0.5 (and even greater when the number of nodes increases).

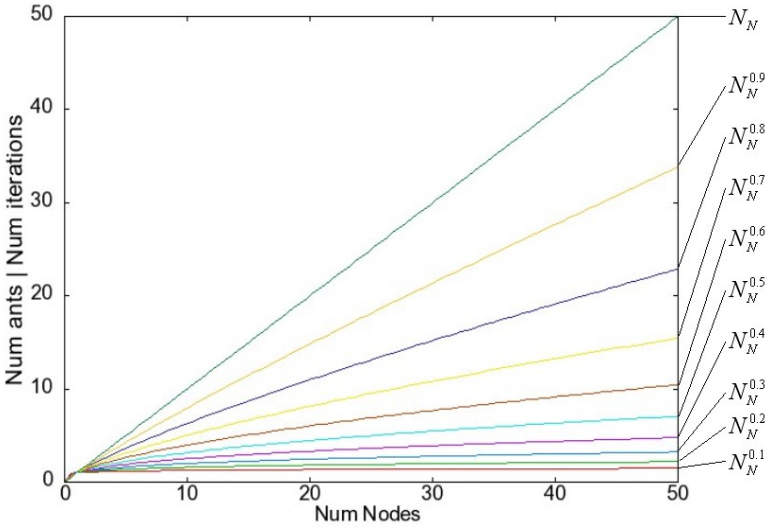


Figure 4. N_{ants} **and** N_{iter} **vs** N_N

Therefore, if N_{ants} and N_{iter} remain close to or less than 0.5, the difference between one value or another is not significant (maybe one or two ants/iterations of difference, which has no real strong effect in the results).

If $TraTh = 0$ then the probability that an ant *k* stops when it discovers a server offering the requested service is determined only by the average pheromone of the current path built by ant *k*, $\overline{\tau}_k$. That is, if $\overline{\tau}_k = 0.9$, for instance, ant *k* has a 90% of probability of stopping when discovering such a server. If $TraTh \to 1$, then $\overline{\tau}_k$ has to be close to 1 (a very trustworthy path) in order to let the ant to stop; otherwise it will keep on trying whenever it finds a server offering the requested service, until it reaches one of these servers without a neighbor to move forward.

PLF is used when assessing the quality of a path (see equation (4)). The bigger PLF is the more influence the path length has on its quality. The average length of the solutions depends on the number of nodes, but it is not usually very high [29]. Thus, there is no much difference between the values of *PLF* accepted in a static scenario. And since its mean value is near to 0.5, equation (4) may be viewed as:

$$Q(S_k) \approx \frac{A_k}{\sqrt{Length(S_k)}} \cdot \overline{\tau}_k \tag{10}$$

Finally, if $PunTh = 0$, every edge of the path leading to the most trustworthy server would be punished according to formula (8); otherwise, if $PunTh = 1$, the punishment applied to all those edges would be the one shown in formula (9). In this first scenario $PunTh$ takes higher values, thus meaning that a stronger punishment method is more suitable.

5.2 Dynamic Networks

The second tested scenario was similar to the first, but including dynamism. Note that P2P networks are strongly characterized by their high dynamism, where every node can enter or leave

the network at any moment, therefore the need of this type of scenario. In this dynamic scenario the topology of the network may change along the time. Table IV and Figure 5 show the outcomes of META-TACS for this scenario.

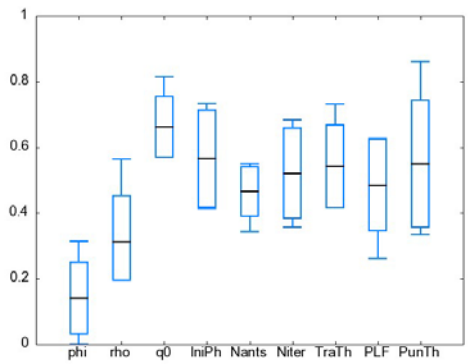
In this scenario φ parameter mean takes a very similar value than in the static scenario, but with a lower deviation, so it has a weaker pheromone local updating and punishment. ρ, however, also has a similar mean but this time the deviation is greater than in the first scenario. This means that greater and lower pheromone global updates are allowed in the dynamic scenario. It is deduced that this exchange of deviation sizes implies that in a dynamic P2P network a better balance is achieved by contributing with less pheromone in the local updating but with a greater trace in the global one, while the punishment is also low. As it can be observed, φ and ρ remain under 0.5 and ρ mean is still greater than φ's one.

As q_0 takes a lower value than in the first scenario, it seems that in a dynamic one is more suitable for ants to choose the next node to move forward using more often the expression (5). And this has sense since with equation (6) an ant always chooses the most promising node to move forward, and this might be good for a static environment. But in a dynamic scenario is better to sometimes use the probability defined in formula (5) to randomly select the next node of the path. Otherwise, following the most promising path could lead us to a dead end, since nodes can leave the network whenever they want.

IniPh has nearly the same standard deviation than in the first scenario, but the mean is higher. This can be interpreted as that in a dynamic environment is better to initially be a bit more confident.

Table IV and Figure 5. META-TACS Outcomes for Dynamic Networks.

	μ	σ	Max	Min
phi	0.1411	0.1096	0.3138	0.0000
rho	0.3116	0.1397	0.5642	0.1950
q0	0.6619	0.0924	0.8156	0.5707
IniPh	0.5655	0.1471	0.7344	0.4132
N_{ants}	0.4661	0.0748	0.5498	0.3435
N_{iter}	0.5208	0.1377	0.6839	0.3567
TraTh	0.5425	0.1264	0.7329	0.4165
PLF	0.4845	0.1395	0.6272	0.2608
PunTh	0.5501	0.1933	0.8613	0.3351
Fitness	0.9996	$3.5 \cdot 10^{-4}$	0.9999	0.9991



N_{iter} remains greater than N_{ants} in this second scenario, and the values of both parameters, as we explained before with Figure 4, are quite similar to those obtained for the first scenario.

The standard deviation of *TraTh* is also very similar in this environment than in the first one but, once again, here the mean is a bit greater than in the static scenario. This situation implies that ants do not stop when they find a server offering the requested service as often as they would do in a static network. In a dynamic network nobody guarantees that a server will stay in the network for a long time, so it is better that ants explore the network as much as they can.

PLF has here a range of values with lower bounds than in the first scenario, but its mean is close to 0.5 too, so the rewriting of formula (4) can be also applied here.

And the behavior of the $PunTh$ parameter is interesting. Here it has a lower mean, closer to 0.5, and a greater interval of values, which means that in a dynamic network the punishment scheme can vary from a quite hard and strict one until a soft and relaxed one. Or it can also be seen as that a good balance between strong and weak punishment is more suitable for dynamic networks.

5.3 Oscillating Networks

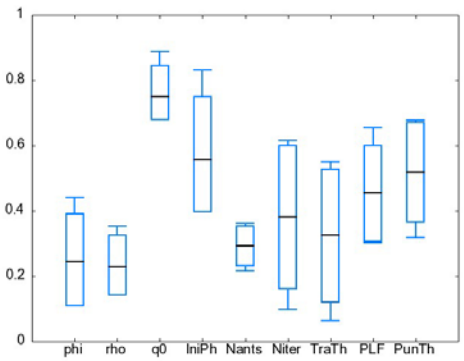
Finally, the third and last scenario consisted of a set of static networks where after every a certain number of transactions (50 in our case) the goodness of the currently selected most trustworthy server swapped, so it turned into a malicious one. In a P2P network any peer can suddenly change its goodness in order to cheat and try to get a greater self-profit. Thus, this kind of scenario aims to study this other type of dynamism, but this time focused on the behaviour of the nodes instead of the variability of the topology of the network In this oscillating scenario, in order to evaluate the fitness of each individual we executed TACS model over 5000 random networks with 1000 transactions per network with sizes of 10 to 20 nodes, 20 to 30, and so on. Outcomes of the genetic algorithm are shown in Table V and Figure 6.

A very significant issue derived from the experiments is that ρ parameter is now very close to φ. In fact it is a bit lower. But it can be because φ takes here the highest value and ρ its smallest one among the three tested scenarios. Remember that this meant that the pheromone local updating and the punishment are both greater, and the pheromone global updating is weaker.

Parameter q_0 has here a greater value than in the dynamic scenario, but lesser than in the static one, so in an oscillating network it is also better for an ant to choose the best path to follow by using equation (6) (probabilistic transition rule) rather than equation (5) (direct exploration).

Table V and Figure 6. META-TACS Outcomes for Oscillating Networks.

	μ	σ	Max	Min
phi	0.2460	0.1451	0.4417	0.1092
rho	0.2294	0.0947	0.3531	0.1429
q0	0.7496	0.0964	0.8888	0.6793
IniPh	0.5558	0.1939	0.8307	0.3987
N_{ants}	0.2926	0.0605	0.3617	0.3435
N_{iter}	0.3812	0.2198	0.6148	0.0985
TraTh	0.3243	0.2031	0.5497	0.0637
PLF	0.4543	0.1468	0.6546	0.3023
PunTh	0.5181	0.1528	0.6771	0.3175
Fitness	0.9995	$7.3 \cdot 10^{-4}$	0.9999	0.9983



The highest standard deviation for $IniPh$ is reached here, but its range of values is quite similar to those obtained in previous scenarios, that is, the most profitable initial conditions consist of not being too confident, neither too unconfident.

N_{ants} takes here its lowest values for both the mean and the standard deviation, while N_{iter} takes the lowest mean, but the highest standard deviation. This implies that, on the one hand a balance between these two parameters is necessary, and on the other hand, N_{iter} is in average term higher than N_{ants}.

$TraTh$ also has its lowest mean and highest standard deviation in the oscillating scenario, which means that the probability of an ant of stopping when it discovers a server offering the desired service is mainly determined by the average pheromone of the path currently built.

The lowest mean of PLF parameter is obtained in this scenario, but its range of values is very similar to the previous ones, and the rewriting of formula (4) done in formula (10) is also valid here.

Again, another lowest mean is reached here. And it is the $PunTh$ parameter. But its values interval is quite similar than the one achieved in the dynamic scenario, so it has the same impact it had there.

In summary, this scenario has the most extreme values among the three tested environments for many parameters. And the reason for this is that an oscillating scenario is the most changing and aggressive for our model (in fact the "worst" fitness values are also achieved here).

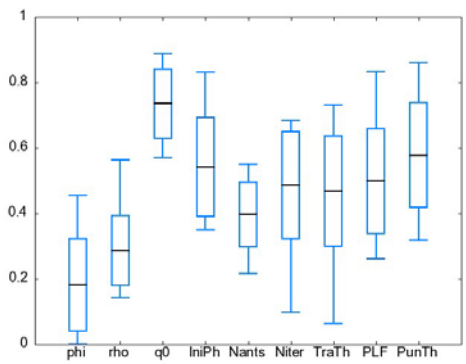
5.4 Global Outcomes

Finally, Table VI and Figure 7 summarize the global outcomes (average of the three tested scenarios) for META-TACS model. These outcomes can give us an interesting perspective of the performance of our model when there is no information about the behaviour of the network nodes or the stability of the topography of the network where it is going to be deployed.

As it can be seen, the standard deviation of every parameter goes from 0.1 to 0.16, approximately, which are high values since all the optimized parameters take values within the interval $[0,1]$. That is, all of them have a standard deviation between the 10% and the 16%, approximately.

Table VI and Figure 7. Global META-TACS Outcomes.

	μ	σ	Max	Min
phi	0.1817	0.1409	0.4545	0.0000
rho	0.2861	0.1073	0.3531	0.1429
q0	0.7359	0.1057	0.8888	0.5707
IniPh	0.5420	0.1516	0.8307	0.3510
N_{ants}	0.3969	0.0982	0.5498	0.2162
N_{iter}	0.4865	0.1645	0.6840	0.0985
TraTh	0.4672	0.1680	0.7329	0.0637
PLF	0.4989	0.1611	0.8330	0.2608
PunTh	0.5782	0.1608	0.8613	0.3175
Fitness	0.9996	$4.4 \cdot 10^{-4}$	0.9999	0.9983



We did not expect with these experiments to obtain a very accurate value for each one of the TACS parameters, but to demonstrate the robustness of the model against a certain range of input values for those parameters. As we can see, standard deviations take most of the times high values, which means a wider interval where to take values from for a certain parameter, obtaining however good outcomes (the worst one is a 99.83%, as it can be seen in Table V).

Some general relations between TACS parameters can be derived from the experiments carried out with the CHC genetic algorithm. These relations are described in Table VII.

These relations are just rough general approximations. In future work some rules relating several parameters could be extracted through a data mining process, for instance.

Table VII. TACS Parameters Relations.

$\rho \geq \varphi$
$N_{iter} > N_{ants}$
$q_0 > 0.5$
$IniPh \rightarrow 0.5$
$PLF \rightarrow 0.5$

6. CONCLUSIONS AND FUTURE WORK

This paper has studied the robustness of TACS (Trust Ant Colony System), a novel trust model for P2P networks using the bio-inspired algorithm of Ant Colony System. It has described its main characteristics and features and shown its performance.

A genetic algorithm based on the CHC algorithm has been applied in order to optimize the working parameters of TACS model and obtaining better knowledge about the behaviour of TACS with regards to them. The performance optimizations of this genetic algorithm have demonstrated the robustness of TACS model against a wide range of working parameter values. That is, TACS seems to work reasonably well regardless the values given for its parameters (within a certain interval).

Besides, this work opens some future ways of research. For instance, some rules relating TACS parameters could be obtained through a data mining process as to better adapt to the particular scenario found when applied in real networks. Although TACS has been shown as a robust system within wide parameter ranges it is always a recommendable safeguard to set the parameters in the best parameter setting areas depending on the type of network at hand. Therefore a self-adapting system based on meta-knowledge rules or metaheuristics may prove a useful addition to META-TACS in the future.

We have focused our model on P2P networks; nevertheless, other environments such as wireless sensor networks, ad-hoc, or even V2V (Vehicular-to-Vehicular) networks are also firm candidates for applying our model on them.

We are also planning to implement and test our model over a simulation environment such as OMNeT++ [28] and even participate in workbenches competitions such as ART Testbed [29].

Finally we are thinking of introducing ontologies, as well as fuzzy sets, in our model in order to provide it with some semantics. For instance, concepts as the similarity between two services could be better modelled in this way, in our opinion.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish MEC as part of the project TIN2008-06441-C02-02 SEISCIENTOS and by a Seneca Foundation grant within the Human Resources Research Training Program 2007 (code 15779/PD/10). Thanks also to the Funding Program for Research Groups of Excellence granted as well by the Seneca Foundation with code 04552/GERM/06.

REFERENCES

1. M. Dorigo and T. Stüztle, “Ant Colony Optimization,” *Bradford Book*. 2004.

2. O. Cordón, F. Herrera, and T. Stüztle, "A review on the ant colony optimization metaheuristic: Basis, models and new trends," *Mathware and Soft Computing* 9 (2-3), pp 141-175, 2002.
3. G. Zacharia and P. Maes, "Trust management through reputation mechanisms," *Applied Artificial Intelligence* (14), pp 881-907, 2000.
4. J. Sabater and C. Sierra, "Regret: reputation in gregarious societies," *Fifth International Conference on Autonomous Agents*, Montreal, Canada, pp 194-195, 2001.
5. J. Sabater and C. Sierra, "Social ReGreT, a reputation model based on social relations," *SIGecom Exch* 3 (1), pp 44-56, 2002.
6. J. Carbó, J. Molina, and J. Dávila, "Trust management through fuzzy reputation," *International Journal of Cooperative Information Systems* 12, pp 135-155, 2003.
7. S. Songsiri, "Mtrust: A reputation-based trust model for a mobile agent system," *Autonomic and Trusted Computing*, Third International Conference, ATC 2006, Wuhan, China, pp. 374-385.
8. L. Mui, M. Mohtashemi, and A. Halberstadt, "A Computational Model of Trust and Reputation," *35th Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, Washington DC, USA, 2002.
9. Y. Shoham, "Agent-Oriented Programming," *Artificial Intelligence* 60 (1), pp 51-92, 1993.
10. F. Almenárez, A. Marín, C. Campo, and C. García, "PTM: A pervasive trust management model for dynamic open environments," *Privacy and Trust*, First Workshop on Pervasive Security and Trust, Boston, USA, 2004.
11. Y. Wang, V. Cahill, E. Gray, C. Harris, and L. Liao, "Bayesian network based trust management," *Autonomic and Trusted Computing*, no. 4158 in LNCS, Third International Conference, ATC 2006, Springer, Wuhan, China, pp 246-257, 2006.
12. A. Tajeddine, A. Kayssi, A. Chehab, and H. Artail, "PATROL-F – a comprehensive reputation-based trust model with fuzzy subsystems" , *Autonomic and Trusted Computing*, no. 4158 in LNCS, Third International Conference, ATC 2006, Springer, Wuhan, China, pp 205-217, 2006.
13. S. Kamvar, M. Schlosser, and H. García-Molina, "The EigenTrust Algorithm for Reputation Management," *P2P Networks*, Budapest, Hungary, 2003.
14. C. Huang, H. Hu, and Z. Wang, "A dynamic trust model based on feedback control mechanism for P2P applications," *Autonomic and Trusted Computing*, no. 4158 in LNCS, Third International Conference, ATC 2006, Springer, Wuhan, China, pp 312-321, 2006.
15. K. Aberer and Z. Despotovic, "Managing Trust in a Peer-2-Peer Information System," *10th International Conference on Information and Knowledge Management*, 2001.
16. L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities," *IEEE Transactions on Knowledge and Data Engineering* 16 (7), pp 843-857, 2004.
17. S. Buchegger and J. Y. Le Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks," *Second Workshop on the Economics of Peer-to-Peer Systems*, Cambridge MA, USA, 2004.
18. A. Abul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," *33rd Hawaii International Conference on System Sciences*, Hawaii, USA, 2000.
19. A. Boukerche, L. Xu, and K. El-Khatib, "Trust-based security for wireless ad hoc and sensor networks," *Computer Communications* 30 (11-12), pp 2413-2427, 2007.
20. W. Wang, G. Zeng, and L. Yuan, "Ant-based reputation evidence distribution in P2P networks," *Fifth International Conference on Grid and Cooperative Computing*, IEEE Computer Society, Changsha, Hunan, China, pp 129-132, 2006.

21. T. Zhuo, L. Zhengding, and L. Kai, "Time-based dynamic trust model using ant colony algorithm," *Wuhan University Journal of Natural Sciences* 11 (6), pp 1462-1466, 2006.
22. M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, "Ant Colony Optimization and Swarm Intelligence," *5th International Workshop*, ANTS 2006, Vol. 4150 of LNCS, Springer, Brussels, Belgium, 2006.
23. T. Stützle and H.H. Hoos, "MAX-MIN Ant System," *Future Generation Computer Systems* 16 (8), pp 889-914, 2000.
24. F. Gómez Mármol, G. Martínez Pérez, and A.F. Gómez Skarmeta, "TACS, a Trust Model for P2P Networks," *Wireless Personal Communications*, Special Issue on "Information Security and Data Protection in Future Generation Communication and Networking", vol. 51, no. 1, pp 153-164, 2009. Camera-ready version available on-line at: http://ants.dif.um.es/staff/felixgm/pub/GomezMarmol-TACS.pdf
25. TACS – Trust Ant Colony System, http://ants.dif.um.es/staff/felixgm/research/tacs
26. D.E. Goldberg, "Genetic Algorithms in Search," *Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston MA, USA, 1989.
27. L. Eshelman, "The CHC Adaptive Search Algorithm: How to Have a Safe Search When Engaging in Non-traditional Genetic Recombination," *Foundations of Genetic Algorithms*, Morgan Kaufman, San Mateo CA, USA, pp 265-283, 1991.
28. OMNeT++ Site, "OMNeT++ Discrete Event Simulation System," http://www.omnetpp.org
29. A. Vázquez, R. Pastor-Satorras, and A. Vespignani, "Internet topology at the router and autonomous system level," 2002, e-print cond-mat /0206084.
30. S. Marti and H. García-Molina, "Taxonomy of trust: Categorizing P2P reputation systems," *Computer Networks* 50, pp 472-484, 2006.
31. X. Chen, K. Zhao, and X. Chu, "SepRep:A Novel Reputation Evaluation Model in Peer-to-Peer Networks," *Autonomic and Trusted Computing*, no. 5060 in LNCS, 5th International Conference, ATC 2008, Springer, Oslo, Norway, pp 86-99, 2008.
32. Z. Yan, V. Niemi, and G. Yu, "A User Behavior Based Trust Model for Mobile Applications," *Autonomic and Trusted Computing*, no. 5060 in LNCS, 5th International Conference, ATC 2008, Springer, Oslo, Norway, pp 455-469, 2008.

ABOUT THE AUTHORS

F. Gomez Marmol is a researcher at the Department of Information and Communications Engineering of the University of Murcia. His research interests include authorization, authentication and trust management in distributed and heterogeneous systems, security management in mobile devices and design and implementation of security solutions for mobile and heterogeneous environments. He received an MSc and PhD in computer engineering from the University of Murcia. Contact him at felixgm@um.es

G. Martinez Perez is an associate professor in the Department of Information and Communications Engineering of the University of Murcia. His research interests include security and management of IPv4/IPv6

communication networks. He received an MSc and PhD in computer engineering from the University of Murcia.

J. G. Marín-Blázquez received his B.Sc. (Honors) and M. Sc. degrees in Computer Science from the University of Murcia, Spain, in 1992 and 1994. He was a Research Assistant at the University of Murcia from 1994 to 1997. He received a M. Sc. and a Ph. D. in Artificial Intelligence from The University of Edinburgh, UK. in 1998 and 2002 respectively. He was a Junior lecturer at The University of Edinburgh and did his postdoc at Napier University, UK until 2004.

He is currently a "Ramon y Cajal" program researcher in the Department of Information and Communications Engineering in the Faculty of Computer Science, University of Murcia, Spain. His research interest includes several branches of Computational Intelligence, including Evolutionary Computation, Linguistic Fuzzy Logic, Artificial Intelligence for Videogames, and Meta and Hyperheuristics.