Prog Artif Intell (2012) 1:183–191
DOI 10.1007/s13748-012-0016-8

REGULAR PAPER

Double multiagent architecture for dynamic triage of victims in emergency scenarios

Estanislao Mercadal · Sergi Robles · Ramon Martí · Cormac J. Sreenan · Joan Borrell

Received: 25 July 2011 / Accepted: 20 January 2012 / Published online: 16 May 2012
© Springer-Verlag 2012

Abstract This paper introduces a double multiagent architecture allowing the triage of victims in emergency scenarios and the automatic update of their medical condition. Gathering updated information about the medical condition of victims is critical for designing an optimal evacuation strategy that minimizes the number of casualties in the aftermath of an emergency. The proposed scheme, currently under development, combines Wireless Sensor Networks (WSN), an Electronic Triage Tag and a double multiagent system (Agilla-JADE) to achieve a low cost, no infrastructure-based, efficient system. Initial results of the WSN roaming done by Agilla agents are presented.

Keywords Mobile agents · Wsn · Emergency scenarios · Multiagent architecture

E. Mercadal (✉) · S. Robles · R. Martí · J. Borrell
Departament of Information and Communications Engineering, Universitat Autònoma de Barcelona, 08193 Cerdanyola del Vallès, Spain
e-mail: emercadal@deic.uab.cat

S. Robles
e-mail: srobles@deic.uab.cat

R. Martí
e-mail: rmarti@deic.uab.cat

J. Borrell
e-mail: jborrell@deic.uab.cat

C. J. Sreenan
Departament of Computer Science, University College Cork, Cork, Ireland
e-mail: cjs@cs.ucc.ie

1 Introduction

The majority of victims whose life is saved after a mass casualty incident (MCI) are treated in the first moments of the incident. At that point only a reduced number of resources are available, and is essential to coordinate efforts to evacuate and urgently treat the most harmed yet curable victims. Therefore, the accurate triaging of victims according to their medical status, done by trained personnel (doctors, nurses, paramedical), is of capital importance. Up to now this triage has been done using a cardboard tag (Fig. 1), which identifies victims' injuries severity using a color code, as well as including other basic medical information. The information written in the tag is decided following a standard triage method such as MTS [11] or START [20].

The efficiency of these scenarios can be improved using Information and Communication Technologies. However, there are some limitations that have to be considered. Emergency personnel, for instance, need to act quickly and will not agree to use a complex system that could slow down its job. Another limitation is the need to rely on some communications system, existing local infrastructure cannot be considered due to damages or for being out of order.

Mobile agents [3] are an interesting technology which can considerably help in this type of scenarios. Mobile Agent based Electronic Triage Tag (MAETT) [12] is an attractive application using mobile agents centered in adding technology to triaging while keeping the budget low. Using a touch screen handheld device, equipped with a GPS receiver, a WiFi interface and a JADE [2] mobile agent platform, a mobile agent is created carrying the START color code, GPS position and victim's information when the triaging is performed. The traditional cardboard triage tag is not substituted by this method, but is also placed on the victim.

The mobile agent will then try to reach the emergency coordination center leaping from handheld to handheld, being stored in a device carried by the emergency personnel for a while, until an appropriate handheld is at reach. The network created to communicate handhelds goes beyond ad hoc or MANET possibilities, for no permanent communication is required from the source to the destination. The routing protocol uses the estimated time to return (TTR) to the coordination center of the handheld device bearer.

The MAETT approach works well and the information about victims is available to the emergency coordination center (ECC) at a low cost. Then this information is used to plan the evacuation of the victims. Though, changes in victims' medical conditions are never informed to the emergency coordination center, and they can be of great importance for the accurate planning of victims' evacuation. A good solution for this is not trivial; the chances for field personnel of finding a victim are much higher than the other way around, but the new information is generated by the victims and not by the field personnel carrying the handheld devices.

Our intended goal is to create a new dynamic electronic triage tag system where any change in any victim's medical state will be easily communicated to the ECC. To achieve this dynamism, we place wireless nodes equipped with medical sensors to every triaged victim, maintaining the triage tag of the MAETT scenario. This wireless nodes run Agilla [6] a platform for mobile agents with communication possibilities with other nodes pertaining to the same Wireless Sensor Network (WSN) [4]. The sensor nodes monitor victims' vital signs and, if in range, communicate any significant change to nearby emergency personnel. The nodes in the WSN are very resource limited, thus different communication strategies than those of the handhelds carried by the emergency personnel, and an interface to connect both devices are required.

To increase the probability of the information being correctly sent from a node in the WSN to a handheld device, a mechanism based on Agilla mobile agents is currently under development, thus exploring all the nodes of the WSN and fusing all the information about changes on victims' status.

As a result, any node in the WSN will be able to communicate these changes as a whole to any JADE platform (triage or rescue member) that moves into the wireless coverage area of any node in the WSN.

To minimize the cost of exploring the WSN a genetic algorithm (GA)-based approach [13] was proposed in our architecture [14]. This paper extends [14] in two ways. First, we show the analysis of using GAs [13], just to conclude that it is too costly to be used in our scenario. Second, we show the results of using an optimized version of Depth-First Search (DFS) to compute the path used by Agilla agents to perform the desired data fusion. DFS is the classic algorithm to compute the spanning tree of a graph when the number of vertices is small. It is fully described in [8].

JADE mobile agents will act as data mules [19], which will carry the initial classification of the victims and every important change in the medical status, gathered by the Agilla mobile agents in every independent WSN, to the emergency coordination center.

In Sect. 2, we describe MAETT and Agilla agents as the starting point of our architecture and scenario, then in Sect. 3 we present the very architecture and how it works, to then show the experimental evaluation of the algorithms proposed to roam the WSN in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Background

2.1 MAETT

Mobile Agent Electronic Triage Tag (MAETT) [12] is a system focused on the triage of victims of Mass Casualty Incidents (MCI), to provide early resource allocation during emergencies when no network infrastructure is available. In [12] MAETT is compared to other triaging systems, agent based or not. MAETT is neither intended to be a comprehensive management system for emergency situations, such as [17], nor a decision support system such as [5,18]. MAETT is closer to the search and rescue subsystem of [21], although based on software agents instead of robotics. For additional information about multi-agent based disaster management systems see e.g. [1].

The foundation of MAETT is mobile agent technology [3], which allows information to be directly transported from terminal to neighbor terminal regardless of the status of the rest of the network at that particular time. Handheld devices run an execution environment for agents, the platform, where mobile agents can be created, executed and forwarded to other terminals, the agents themselves are who decide the route to follow depending upon the available information on the neighbors.

The main actors of the system are the victims, the triage personnel, and the rescue teams (see scenario in Fig. 1). Victims are usually scattered over an arbitrarily large area of emergency and the triage personnel scour all this area looking for victims and triage them according to standard methods. The result of this triage is written in a physical tag and placed visibly on the victim. Finally, the rescue teams collect all the victims, prioritizing depending on triage results. The Emergency Coordination Center (ECC) coordinates all actions. Triage personnel, rescue teams and the ECC have wireless devices with a JADE mobile agent platform and a GPS receiver.

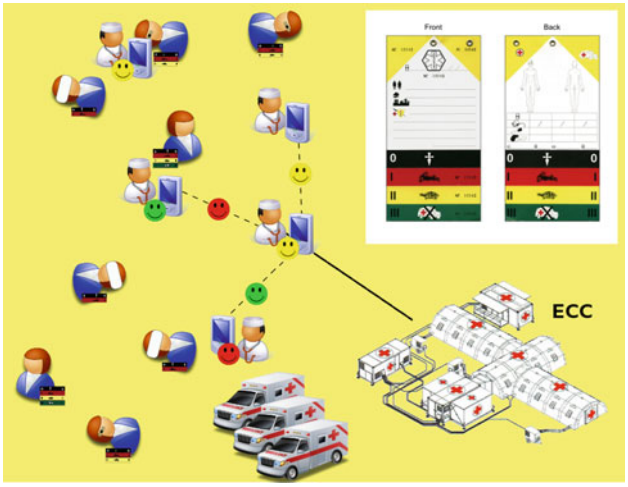


Fig. 1 MAETT triaging scenario showing the classical cardboard triage tags

Triage personnel leave the ECC, and have an estimation on when they will get back, the time to return (TTR). When a victim is found, they use the standard START method [20] and place a cardboard triage tag (see Fig. 1) on the neck of the victim with their evaluation written on it. The tag has an integrated RFID. At the same time, an agent is created containing the information in the tag plus the GPS position of the victim and the RFID of the tag. All this information will be used later in the ECC to optimize the route of the rescue teams. This agent is transmitted to neighbor devices only if the bearer has a lesser TTR. This is to make sure that moving the information is never going to make it arrive later. Consequently, all handheld devices carried by triage personnel are used to create agents with information about found victims, and also to forward agents corresponding to other victims. When agents arrive to the ECC, the ECC send the rescue teams with a detailed schedule of the route based on the GPS position of victims as well as their medical condition.

2.2 Wireless sensor networks and Agilla mobile agent middleware

A Wireless Sensor Network (WSN) [4] is a specific type of *ad hoc* networks, built using wireless radio communication. It consists of sensor nodes collecting particular measures, i.e. temperature or blood pressure, and processing elements, which collect these measures for further processing. Usually sensor networks are strongly resource-restricted in terms of communication, processing and storage capabilities, and in terms of available energy. For the most part, all sensor nodes deliver their data to a base station or *sink* of data. This *sink* can be part of the network, or be external, accessed trough a gateway to other networks.

Application examples of WSNs [4,6] are increasing, and include, among others, emergency operations, habitat monitoring, precision agriculture, home automation and health care or logistics. Initial applications within WSNs were static, i.e. all the nodes run statically installed software loaded prior to their deployment. Nowadays, different systems have been developed to allow more adaptable WSN applications (see [6] for an excellent survey of such systems), from mechanisms that allow a total or partial reprogramming of the nodes, to middlewares that allow the execution of several mobile agents inside each node, allowing application self-adaptability. Among the different mobile agent middlewares proposed for WSNs, Agilla [6] is the first one deployed inside real WSNs.

Agilla provides a programming model in which applications consist of evolving communities of agents that share a WSN. Agents can dynamically enter and exit the WSN, can autonomously clone and migrate themselves in response to environmental changes, and can maintain a global coordination through a tuple space, a type of shared memory accessed via pattern-matching that enables a decoupled style of communication. The size of the tuple space is up to 48 bytes in each node. Agilla was implemented on top of TinyOS WSN operating system [7], and experimentally evaluated on several real WSNs, for instance those consisting of TelosB [16] nodes. A basic Agilla installation in a TelosB node takes up 3866 bytes out of 10kB of RAM and 45308 default bytes out of 48kB of ROM.

3 Double multiagent architecture to provide dynamism to MAETT

Despite being two different agent technologies, JADE and Agilla can coexist and share information to build a more complex agent system. Albeit agents themselves cannot migrate transparently between the two platforms, we show that Agilla-JADE cooperation is feasible.

In our solution, we use Agilla to continuously monitor Mass Casualty Incident (MCI) victims inside WSNs and JADE to carry the monitored data to the Emergency Coordination Center (ECC), introducing dynamism to MAETT. We take advantage of the communication between both technologies to share victim information and route details, thus improving the efficiency of the triaging system.

We extend MAETT scheme (Fig. 1) by adding a wireless human body monitoring device to each victim, for example that of Fig. 2a, a TelosB compatible node manufactured by Maxfor (http://maxfor.co.kr), thus creating a WSN among neighboring victims, which we can use to dynamically update the medical status of every victim.

To communicate the WSN and the handheld devices of triage or rescue members we also need these members carry

Fig. 2 The different devices of the scenario

(a) Watch type body monitoring device

(b) A sensor node connected to a handheld

a WSN node (Fig. 2b) attached to their handheld device, also a TelosB compatible node from Maxfor.

The new double multiagent architecture can be seen in Fig. 3, where black smileys correspond to Agilla victim monitoring agents and colored ones to JADE agents found in MAETT.

3.1 WSN set up and operation

In the first run of the triage personnel, every victim receives the classical triage tag, and is also equipped with a wireless body monitoring sensor node. Inside this node resides an Agilla agent (*Victim* agent) with the specific data of the patient (Victim ID and Medical Condition). Each of this *Victim* agents supervise the associated victim health constants maintaining an up-to-date log with his/her medical condition. In turn, the wireless body monitoring sensor node is identified by a two-dimensional coordinate (WId, AId), the first one (WId) indicates the WSN it pertains, and the second one (AId) identifies the node into the network.

As every nearby victim is both paper tagged and electronically tagged, neighboring wireless nodes belonging to the same triage member wirelessly connect creating a growing WSN of victims.

When every victim in the vicinity is tagged and every body sensor node monitors its own victim, the triage team member ends the creation of his WSN. First of all, the handheld device starts the calculation of the spanning tree of the graph using the DFS algorithm [8] (see Sect. 3.2) to obtain an itinerary to roam the newly created WSN and fetch the updates of every victim's medical condition. A roaming agent (*Traveler* agent) with this computed itinerary is then injected inside the last node of the WSN. *Traveler* agent gets and sets the differences in the state on every node.

When a new rescue or triage team member approaches the WSN, and their handheld device contacts a node in the WSN, the sensor node attached to the handheld automatically identifies itself as the *sink* node. Then an Agilla mobile agent (*Extractor* agent) containing the information of every victim is sent to the node on the handheld to flush all the gathered data to a JADE agent (*Courier* agent). As every node of the WSN has the most up-to-date state as possible, any member of its members in direct contact with the *sink* node can indistinctly output the information. The handheld device routes *Courier* to the ECC, as was done in MAETT.

To use any node of the WSN to flush the status of every victim to the handheld device, every node must maintain a record of every other node status. This status is identified for each victim by his concrete medical condition, the ID of the victim, and a modification flag. The *Traveler* agent, in communication with each *Victim* agent continuously updates this information maintaining an up-to-date record of every victim.

3.2 Roaming the whole WSN

The problem of visiting every node avoiding repetition and optimizing distance is deeply studied in graph theory, also known as the Traveling Salesman Problem (TSP), and is known to be NP Complete [15] even in the Euclidean plane.

The case of covering all the nodes of a WSN can be seen as a particular case of the TSP, with the particularity of pretending to also optimize the energy consumption, and is also proven to be NP Complete [13,23].

Genetic algorithms are being used to solve NP-Complete problems since the early 90s [9] and have been proved useful to efficiently solve the concrete particular of the TSP [10].

In our handheld device, with its restrictions, both of time and computation power, is not feasible to calculate the optimal route for a problem of this kind. Thus, instead of using a more complex algorithm, e.g. those of [22], we tried a highly studied genetic algorithm approximation [13], which is proved to offer good enough solutions.

The chosen genetic algorithm starts creating an initial population of random paths that cover the network. Then, from that set of paths only those whose number of hops to cover the network is smaller or equal to the time remaining to meet the time deadline (maximum number of nodes per path) are selected. After that, a new generation is generated by selecting x individuals to be crossed. A subset of the paths of two

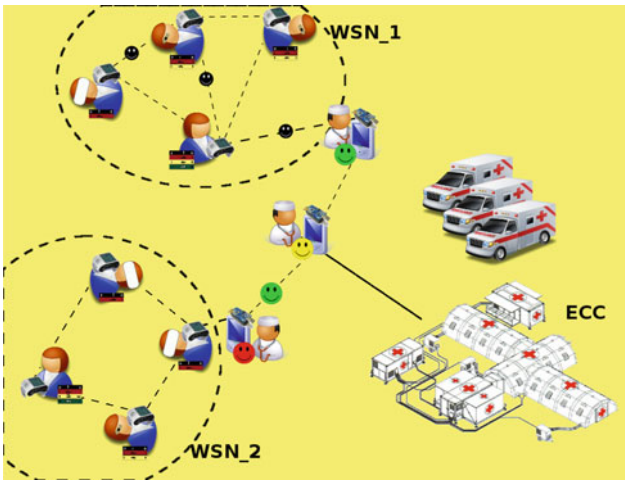


Fig. 3 Dynamic MAETT triaging scenario

individuals are crossed to create a new individual. Finally, all solutions are evaluated and sorted from the minimum to the maximum. We adapted this genetic algorithm not to calculate the minimum route to a node, but to determine an efficient way to cover the whole WSN.

Massaguer [13] worked well (see Sect. 4) in scenarios where all the nodes (vertices) of the network (graph) were connected. When this premise is not fulfilled, the cost of generating valid individuals is too high to be used with our handheld device. Indeed, in scenarios where the network is partially connected, it is difficult to find good individuals after every iteration, due to the high cost of finding a valid route for the graph in the crossing and mutation phases of the genetic algorithm.

In such scenarios, another method to compute the path for our mobile agent was needed. Our chosen solution is the Depth-First Search (DFS) algorithm [8], which computes the spanning tree of a given graph. For us, the returned tree is the path the Agilla agent will use to visit all the nodes of the network.

Note that the computation of the DFS algorithm in the handheld device is possible due to the following facts:

1. The triage personnel saves the topology of the WSN while tagging victims.
2. The number of nodes is limited. Both to speed up the route calculation process and to lighten the medical personnel bags. Notice that every health monitoring sensor weights about 70 g including the required two 1.5V batteries. Assuming that the triaging personnel carries the handheld device and the paper tags, they may agree to carry the extra weight of 20–25 sensor nodes, that is, the weight of a small laptop (i.e. ∼1.5 kg).

Moreover, to restrict the itinerary to our concrete needs we established three limitations to the DFS algorithm to return a valid solution for our network. The first of them is the selection of the initial node, which is the last victim a triaging personnel member tags. This is done for the triage member to not have to move to the best possible starting node, but inject the node where he is.

The second limitation is that the itinerary does not need to be cyclic, that is, does not have to connect the last node to the first node. We added this restriction firstly to ease the computation of a solution, and secondly because what we want is not to reach a specific node, but to have the sensor readings of every one of them. Thus, we can just follow the same path backwards and return to the first node with it.

The third and final restriction is about the amount of time the triaging member will wait for a good itinerary to show up. In our handheld devices Nokia N810, a valid DFS solution appears in a short period of time, less than a second, but with little more time (∼20s) we can get a better result in some cases, see Sect. 4. We established a maximum waiting time of 30s, which gives enough time to compute a good solution, and is short enough for the triaging member to wait for.

Likewise, we defined which are the conditions to determine if an itinerary is better than another. In our case what we want to minimize the amount of energy consumed by the sensor nodes, thus increasing its lifetime. As all the nodes are up and running sensing victims' medical conditions, the only thing we can improve to minimize the energy consumption is the time spent in the transmissions. To do so, a good approach can be to minimize the distance an agent has to move to reach the following node. In doing this, we also reduce the errors made during the transmission, thus minimizing even more the amount of time needed for the agent to migrate.

As a final step, the handheld device injects the *Traveler* Agilla mobile agent containing the route into the WSN. This agent then covers the whole network collecting (in cooperation with *Victim* agents) and informing changes in victim's state, thus writing the information of every victim in every node. The agent keeps doing the same route keeping the victim's medical condition up-to-date.

Additionally, if a problem in the communication between WSN nodes occur, e.g. due to a malfunctioning or broken node, the *Traveler* agent has the ability to reroute itself and find an alternative path to cover the whole WSN. In our case, the mobile agent tries to move to the next available node in the WSN. If it is not reachable tries to move to a random available node.

3.3 WSN maintenance

Another addition that boosts the dynamism of our triaging method is the possibility to modify an already deployed WSN such that a victim can be added or removed, while being added to or removed from the itinerary of the roaming agent.

Using JADE agents and their routes to the ECC, we have an excellent infrastructure to attain this dynamism.

In every triage, the handheld device of the triage personnel is also loaded with the position of every node of the WSN, thus ending up with a full view of the topology of the network. After the computation of DFS, this topology is also loaded into a JADE agent and routed to the ECC.

On arrival to the ECC, the JADE agent flushes the WSN topology, which is shared to every other handheld device deployed in the emergency. Hence if some other triaging member has to tag or remove a victim in another's WSN, he just removes or adds this new victim to the corresponding topology in the handheld device, and calculates a new itinerary (using DFS) with the modification.

Of course, just as in the first calculation, the new topology is routed, along with the medical data of the victims in the WSN, with a JADE agent to the ECC.

3.4 The WSN—handheld device interface

To properly set the interface we had to modify Agilla to run side-by-side with JADE. Doing so Agilla can obtain the instance of the running platform and generate an agent there. Then, we modified Agilla's AgentInjector to switch on JADE when invoked, this way we are sure that from that moment on, every received agent will be able to generate a JADE agent. The agent which migrates or sends data to the handheld device, the one to be encapsulated into a JADE agent, needs a special portion of code permitting the communication with the Injector at the other side of the USB interface. This code moves or sends the data to a special address pertaining to the Injector in the handheld device.

Our test platform (Nokia N810 with the Maemo OS (Diablo 5.2008.43.7)) did not come with the needed drivers for the mote to work. Thus, we had to recompile the stock kernel adding the appropriate drivers, and load them into the device. Moreover, albeit the device supports *host mode USB*, not every USB cable is prepared to notify the host device of this requirement, thus, we had to explicitly tell our test platform to switch itself to *host mode*. Once our test platform was plugged and the drivers loaded, the system recognized a new USB device, but complained about not being able to access it. The problem was related to the power the device was receiving not being sufficient to maintain it operating. We had to add a rule to the *udev* daemon of the handheld device indicating that this particular device needs more power than the allocated by default.

4 Experimental evaluation

We tested the genetic algorithm of [13] and the DFS algorithm of [8], both with the optimizations described above, in our Nokia N810 platform. The goal of our evaluation was to check if these algorithms were appropriate to compute the route of our *Traveler* agent in scenarios where all nodes had connectivity, and in scenarios where some of the links were unavailable.

Table 1 Results for the N810 using the genetic algorithm with a fully connected graph of *j*Nodes

	10 individuals	50 individuals
5N Sc.	*cost*: 16.0755	*cost*: 15.5693
	time: 0.7540	*time*: 6.0744
10N Sc.	*cost*: 31.5538	*cost*: 31.3979
	time: 2.7115	*time*: 13.0550
25N Sc.	*cost1*: 68.3264	*cost1*: 69.5351
	time1: 13.2146	*time1*: 15.0000
	cost2: 55.7277	*cost2*: 59.3514
	time2: 30.0000	*time2*: 30.0000
		cost3: 55.8480
		time3: 52.9582
50N Sc.	*cost1*: 134.1062	*cost1*: 115.9869
	time1: 15.0000	*time1*: 15.0000
	cost2: 106.7367	*cost2*: 107.5450
	time2: 30.0000	*time2*: 30.0000
	cost3: 102.8928	*cost3*: 101.5540
	time3: 42.7622	*time3*: 60.0000

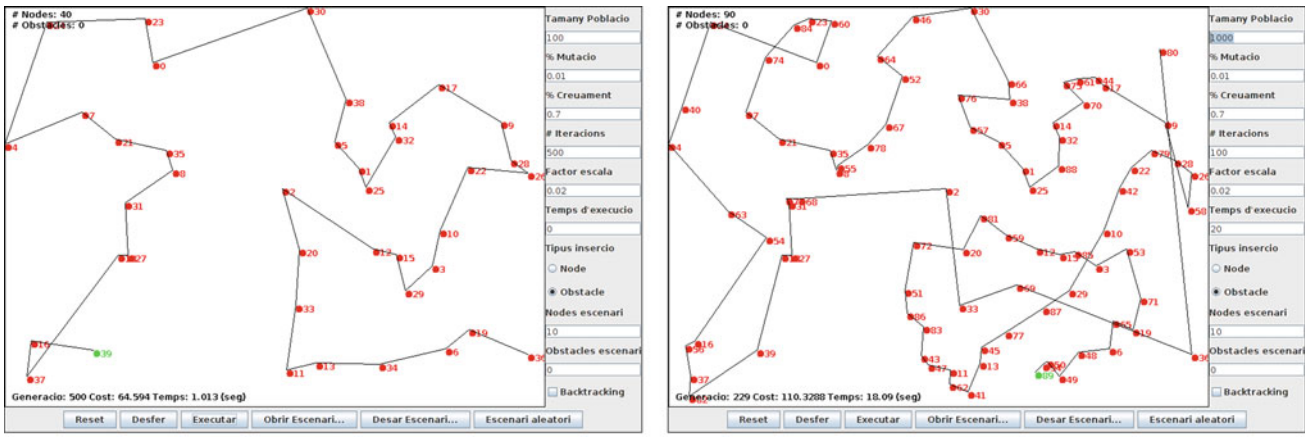
We first tested the genetic algorithm against fully connected, randomly generated graphs. The tests were conducted with initial populations of 10 and 50 randomly generated individuals. Generations had a mutation probability of 0.01 and a crossover probability of 0.7.

The first results (see Table 1) with the genetic algorithm and with fully connected graphs (Fig. 4a, b) showed good values both in computation time and in path cost (length). In that table each cost value is the mean of five executions of the algorithm in our Nokia N810 handheld.

The results show the best solution found for time limits of 15, 30 and 60 s. In some cases, the best solution is found before reaching the limit, and in other cases the algorithm is still computing better solutions when the limit is reached.

As we can see, a solution is found within the 15 s time limit in a 50 nodes scenario, but executing the algorithm during some more time until reaching the 30 s limit, an improvement of approximately 20 % is accomplished while not trespassing the hypothetical time limit established for the medical personnel to wait during a triaging (30 s).

We also tested the genetic algorithm against not fully connected randomly generated graphs. We used the same configuration as in the previous tests with 10 and 50 randomly generated individuals, with a mutation probability of 0.01 and a crossover probability of 0.7.



(a) 25 nodes **(b)** 50 nodes

Fig. 4 WSNs solved with the genetic algorithm

Table 2 Results for the N810 using DFS in a partially connected graph

5N1O Sc.	10N1O Sc.	25N3O Sc.	50N3O Sc.
cost: 21.995	*cost1*: 77.0664	*cost1*: 78.3977	*cost1*: 103.3174
time: 0.066	*time1*: 0.181	*time1*: 0.102	*time1*: 0.179
	cost2: 68.4044	*cost2*: 77.5867	*cost2*: 103.1823
	time2: 1.567	*time2*: 81.448	*time2*: 135.264
	cost3: 54.4441	*cost3*: 77.3539	
	time3: 5.351	*time3*: +300	
	cost4: 50.2357		
	time4: 203.782		
	cost5: 45.0835		
	time5: +300		

In these cases, where there is some obstacle blocking the communication between any two nodes, the genetic algorithm is not able to obtain a solution in a reasonable period of time. The cost of validating every individual and maintaining the population constant is so high that it is not feasible to find a solution to our problem within our established time limits. Only the validation of the individuals after every generation lasts longer than the established time limits.

Nonetheless, using DFS [8] we solved this situation and found good solutions in a small amount of time. Moreover, in some cases, waiting for the the time limit to end, the solution improves considerably (see Table 2, where jNkO stands for j Nodes and k Obstacles).

DFS proved to work well with random scenarios (Table 2), so we tested it with special scenarios, easily found in actual catastrophes, for example reflecting a building (Fig. 5a), a node distribution with one single solution (Fig. 5b), a zone separated by a wall (Fig. 5c), or a star shaped scenario (Fig. 5d). The algorithm also performed well in this kind of scenarios, finding good solutions in reasonable amounts of time as can be seen in Table 3.

Differently from the genetic algorithm, a new solution pops out nearly instantly using DFS, and usually is the best solution we can find, even waiting for the time limit to end. In the specific scenarios tested, the difference between the first solution and the next better solution is so small, even inappreciable, that waiting for the new solution is, in most cases, not worthy. First responders have a limited amount of time to triage the greatest number of victims in a MCI, so having them waiting for a better solution to appear is not advisable.

In the special cases of the Unique Path (Fig. 5b) and the Star (Fig. 5d) cases, there is only one possible solution, and the algorithm finds it almost instantly, as is reflected in Table 3.

5 Conclusions

The triage stage during the aftermath of an emergency is critical to minimize the number of casualties. There are many systems in the bibliography introducing Information and Communication Technologies to achieve this, most times at a high cost or requiring a non-practical interaction with the system that field personnel refuse to perform. This paper presents a multiagent architecture allowing the triage of victims in emergency scenarios and the automatic update of their medical condition at a low cost.

In our approach, neighbor victims are automatically grouped together by placing a wireless sensor node on each of them monitoring their status. This groups form a Wireless Sensor Network (WSN). Changes are shared within the WSN using Agilla mobile agents and a gathering algorithm designed using genetic algorithms. Now, changes can be communicated to any member of the

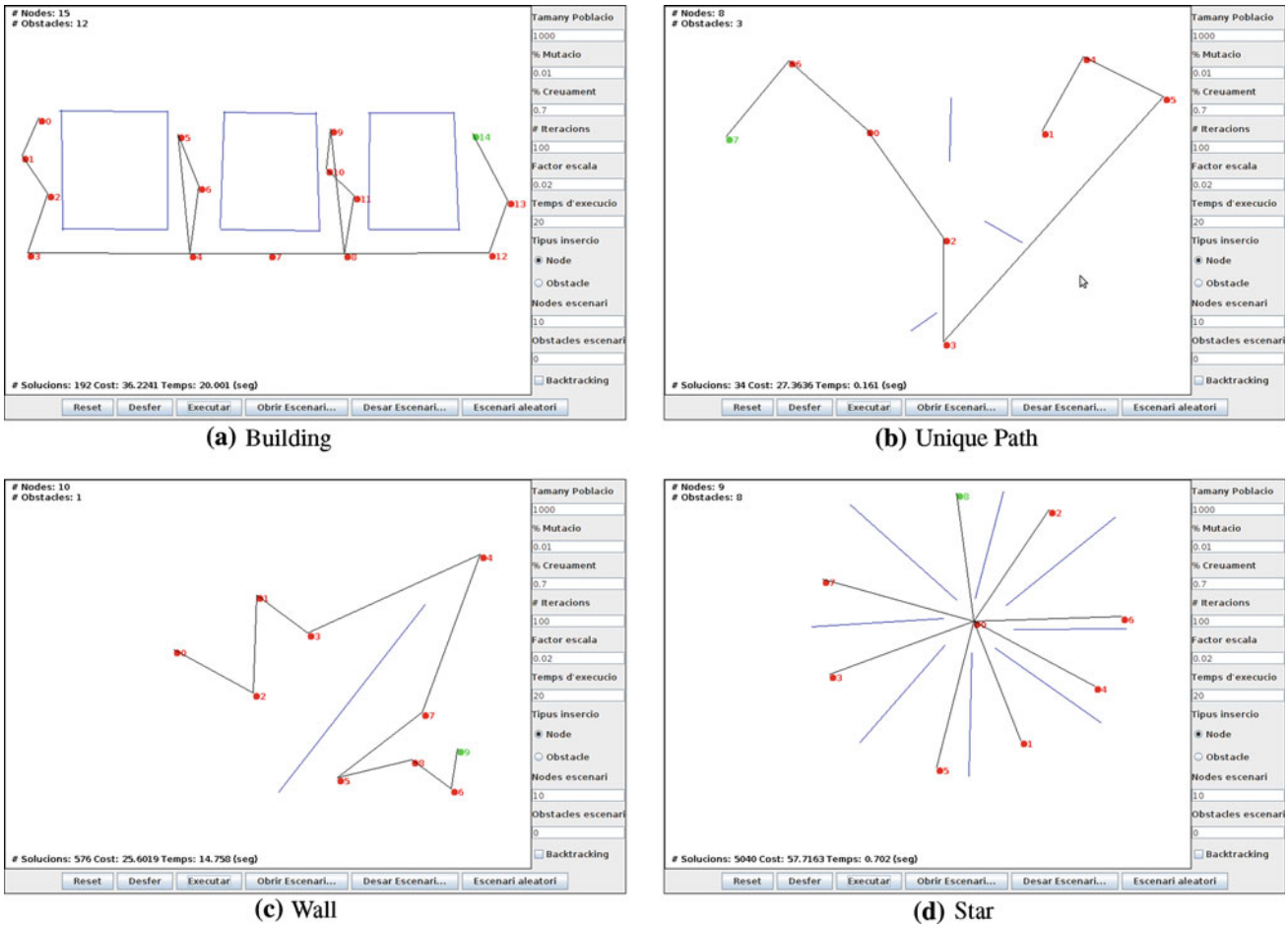


Fig. 5 Special interest scenarios

emergency personnel by any node of the group, increasing in this way the probabilities of sending this information. From there, a JADE mobile agent will carry the changes to the coordination center where they will update old information.

The choice of using genetic algorithms to find a solution for our problem has been dropped in favor of the DFS algorithm due to the difficulties of finding good individuals after each iteration in a reasonable amount of time. DFS offers good enough solutions in a smaller amount of time, thus making it appropriate for a mass casualty aftermath triaging, where time is of capital importance.

The results also show that waiting for a better solution is not always worth, as the benefit obtained when finding the next better solution is generally so small that the waiting ends up being misused time.

Although the simulations offered good results in the tested scenarios, we are now using our solution in real scenarios with actual wireless nodes, i.e. a real building or an open rough terrain, to validate the simulated results.

Table 3 Results for the N810 using DFS in special interest scenarios

Building	Unique path	Wall	Star
cost1: 36.5687	*cost*: 27.3636	*cost1*: 27.6036	*cost*: 57.7163
time1: 0.08	*time*: 0.001	*time1*: 0.058	*time*: 0.061
cost2: 36.0348		*cost2*: 25.6019	
time2: +120		*time2*: 12.434	

Simultaneously, we are running Agilla agents with the generated paths and we test them against node failures on realistic scenarios. We are considering two different redundancy strategies, the first one using just one single agent able to switch among k different paths, and the second one using k simultaneous mobile agents roaming the WSN, each with a different itinerary.

Acknowledgments This work has been funded by the Spanish Ministry of Science and Innovation through the project TIN2010-15764. We would like to thank Aitor López for his collaboration when imple-

menting both the genetic algorithm and the DFS, and Daniel Massaguer for his helpful comments on this implementation.

References

1. Basak, S., Modanwal, N., Mazumdar, B.D.: Multi-agent based disaster management system: a review. Int. J. Comput. Sci. Technol. **2**, 343–348 (2011)
2. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. Wiley, New York (2007)
3. Cucurull, J., Ametller, J., Martí, R.: Agent mobility. In: Developing multi-agent systems with JADE, Wiley Inc., pp 115–130 (2007)
4. Dressler, F.: Self-Organization in Sensor and Actor Networks. Wiley, New York (2007)
5. Filippoupolitis, A., Gelenbe, E.: A decision support system for disaster management in buildings. In: Proceedings of the Summer Computer Simulation Conference, pp. 141–147, Istanbul, Turkey (2009)
6. Fok, Ch-L., Roman, G-C., Lu, Ch.: Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. ACM Trans. Auton. Adapt. Syst. **4**(3), 1–26 (2009)
7. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 93–104. ACM (2000)
8. Iyengar, S.S., Parameshwaran, N., Phoha, V.V., Balakrishnan, N., Okoye, C.D.: Fundamentals of Sensor Network Programming: Applications and Technology. Wiley-IEEE Press (2010)
9. De Jong, K.A., Spears, W.M.: Using genetic algorithms to solve NP-complete problems (1989)
10. Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artif. Intell. Rev. **13**, 129–170 (1999). doi:10.1023/A:1006529012972
11. Mackway-Jones, K. (ed.) Emergency triage, 2nd edn. Wiley, New York (2006)
12. Martí, R., Robles, S., Martín-Campillo, A., Cucurull, J.: Providing early resource allocation during emergencies: the mobile triage tag. J. Netw. Comput. Appl. **32**, 1167–1182 (2009)
13. Massaguer, D.: Multi mobile agent deployment in wireless sensor networks. Master's thesis. University of California, Irvine (2005)
14. Mercadal, E., Robles, S., Martí, R., Sreenan, C., Borrell, J.: Heterogeneous multiagent architecture for dynamic triage of victims in emergency scenarios. In: In 9th International Conference on Practical Applications of Agents and Multiagent Systems (PAAMS), pp. 237–246 (2011)
15. Papadimitriou, C.H.: The Euclidean travelling salesman problem is NP-complete. Theor. Comput. Sci. **4**(3), 237–244 (1977)
16. Polastre, J., Szewczyk, R., Culler, D.: Telos: Enabling ultra-low power wireless research. In: IPSN '05 Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, pp. 364–369. ACM and IEEE (2005)
17. Quillinan, T. Brazier, F., Aldewereld, H., Dignum, V., Dignum, F., Penserini, L., Wijngaards, N.: Developing agent-based organizational models for crisis management. In: Proceedings of Eighth Joint Conference on Autonomous and Multi-Agent Systems (AAMAS 2009), pp. 45–52 (2009)
18. Schoenharl, T., Szabó, G., Madey, G., Barabási, A.-L.: Wiper: A multi-agent system for emergency response. In: Proceedings of ISCRAM (2006)
19. Shah, R.C., Roy, S., Jain, S., Brunette W.: Data mules: modeling a three-tier architecture for sparse sensor networks. In: Proceedings of Sensor Network Protocols and Applications (SNPA), pp. 30–41. IEEE (2003)
20. Super G.: START: a triage training module. Newport Beach, California: Hoag Memorial Hospital Presbyterian (1984)
21. Tadokoro, S., Kitano, H., Takahashi, T., Noda, I., Matsubara, H., Shinjoh, A., Koto, T., Takeuchi, I., Takahashi, H., Matsuno, F., Hatayama, M., Nobe, J., Shimada S.: The robocup-rescue project: a robotic approach to the disaster mitigation problem. In: Proceedings of ICRA, pp. 4089–4094 (2000)
22. Wang, X., Chen, M., Kwon, T., Chao, H.C.: Multiple mobile agents' itinerary planning in wireless sensor networks: survey and evaluation. Commun. IET **5**(12), 1769–1776 (2011)
23. Wu, Q., Rao, N.S.V., Barhen, J., Iyengar, S.S., Vaishnavi, V.K., Qi, H., Chakrabarty, K.: On computing mobile agent routes for data fusion in distributed sensor networks. IEEE Trans. Knowl. Data Eng. **16**(6), 740–753 (2004)