Intelligent system for exploring dynamic crisis environments

Bogdan Tatomir
Delft University of Technology, NL
DecisLab, NL
B.Tatomir@ewi.tudelft.nl

Leon Rothkrantz
Delft University of Technology, NL
DecisLab, NL
L.J.M.Rothkrantz@ewi.tudelft.nl

Mirela Popa
Delft University of Technology, NL
mirela@mmi.tudelft.nl

ABSTRACT

The routing in complex buildings is provided by information systems. But during a crisis situation, these systems may collapse due to certain incidents like an explosion, a fire or sabotage. The task of guiding people in this situation has to be handled in some way. In this paper we present a possible solution to this problem. We use a multi-agent system in a mobile ad-hoc network, without the need of any infrastructure. The main idea of the paper is that just by exploring the damaged building, the data of the changing environment becomes available and the challenge is how to fuse this data from different observers. We focused on the way of building, sharing and merging topological maps, using observations from individuals present in this infrastructure-less network. Besides a more efficient exploration of the building, the system presented in this paper can provide the rescue teams with additional services like finding the nearest exit. Some results of the tests we run with our system are also presented.

Keywords

MANET, emergency, crisis, infrastructure-less, topological map, graph matching, map merging.

INTRODUCTION

The routing in complex buildings and places is based on static signs. In case of a crisis routes (doors, staircases and corridors) can be blocked and the static routing system breaks down. There is a need for a dynamic, personalized, adaptive routing system. An answer to this problem can be an infrastructure-less mobile ad-hoc network that enables the system to share information concerning the state of the environment and other valuable information about the people within. The advantage of using these ad-hoc networking technologies is that the exchange of information can be done anywhere, anytime without any prior knowledge of the network infrastructure. Using handheld devices that operate in a wireless environment, communication is still possible when major infrastructural communication links have been damaged, destroyed or overloaded.

The work presented in this paper is part of the Combined project running at DECIS LAB (Burghardt, 2004). Roughly the aim of the project is to develop an environment wherein rescue services can communicate using handheld devices dynamically forming MANETs (Tatomir and Rothkrantz, 2005). Users of such networks should be able to exchange observations through agent technology and intuitive GUI's located on a handheld set. Agents aid the user in finding, storing and retrieving information from the network. Our specific interest in this paper is the distribution of world knowledge in these ad-hoc communication networking environments.

In the traditional view of rescue workers there is a centralized decision making process. We adapt to a decentralized a decision making process. Rescue workers are exploring a dynamic changing environment. Based on the results of exploration, a new structure/map of the building emerges. A rescue workers base their decisions on the emerging map information. A centralized approach is less appropriate and implies a lot of communication. In wireless environments we have to take care of communication overload.

This paper is organized as follows. In the next subsections we describe the assumptions we made about the environment where our system is going to run and present related work. Next we introduce a greedy algorithm we use for merging the topological maps and discuss some strategies we applied in order to avoid errors and better recover from inconsistencies. We describe the developed simulation environment, tests and the results we obtained. The last section contains conclusions, future work and further applicability of our approach for crisis response.

Assumptions

As an anticipated setting for the system, consider a building on fire, covered with smoke where people are trying to leave the building and firemen are performing tasks such as trying to rescue these people and exterminating the fire. We make the reasonable assumption that both the firemen and the people inside have no map of the building and therefore are hindered in their goals by their lack of world knowledge (Figure 1). In case a map is valid we can't expect it is valid anymore. During a crisis doors, staircases and corridors can be blocked.

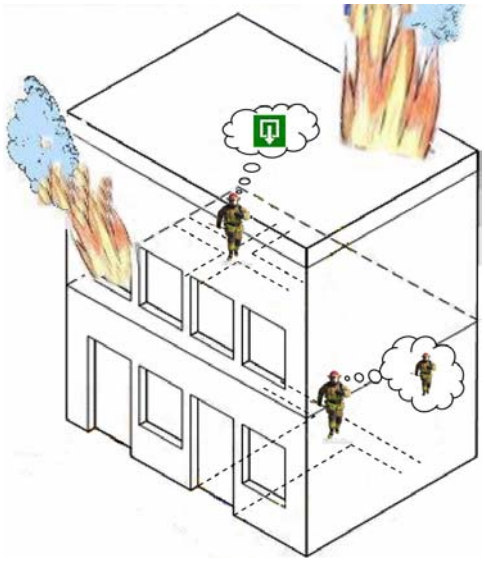

Figure 1. Lack of world knowledge hinders users

For people (including firemen) to be able to effectively reason about their environment without any pre-knowledge, a model of the world needs to be constructed. Next, all the local models should be fused to a share world model. To build a local world model in a world with limited view (smoke, fire), implies to keep record of the crossings and directions and make distance estimations. But usually the main focus of interest for a fireman is to rescue people. We can imagine though that this process could also be left to a computer (PDA) and most of this work can be done in the background. As a first step to a fully automated input supply, we assume that all a person would have to do is to indicate that at a certain time he encounters an intersection and the number of paths with their directions. For reporting about the situation, we have developed an application based on icon-communication. This natural interaction style is based on a graphical user interface (Figure 2). It is language independent, which makes it suitable for crisis situations (Fitrianie and Rothkrantz 2005).

The icons are grouped into categories, according to their meaning (signs, building elements, directions, numbers, crisis elements and people). Each one of the categories has a representative icon that shows the main characteristics, being like a hint for the 'background' list. In our example case, the exit sign icon is chosen as representative for the set of icons used to construct the map. In this category we also have 6 icons representing the type of crossings and one for a road block. The category with numbers is used to mention the number of the starting floor. For directions, four icons (arrows) are used to mention when a turn is made at a crossing or in case the floor is changed. A category with icons for stair cases, windows, doors can be used to add additional details on the map. Also is possible to indicate crisis elements like fire or smoke, and to give details about the presence of victims or other members of the emergency team. For the current floor, a graph created on observations and actions from user reports can be displayed on the screen of the user's PDA.

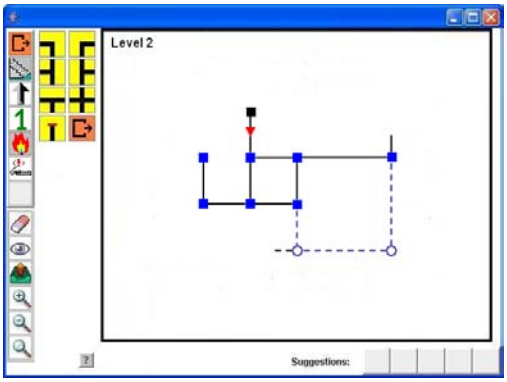

Figure 2. Iconic interface for PDA

It is also important to be able to estimate (relative) distances traveled. These distances can be calculated from the time between two observations, combined with the type of movement (running, walking, etc) and can be refined by using pedometer input. There are more possibilities but we prefer to use as little specialized hardware as possible. In our system we will make the assumption the user provides relatively accurate indications of distances traveled (+/- 10%).

During the exploration activity, each individual builds a map of his own world (Figure 3). The challenge is to share all this individual world models (Figure 4). Assuming they are all carrying a portable computing device capable of wireless communication, it is possible for them to communicate about their world models with each other and make attempts to build a more or less shared world model. Because a single human user is only able to explore so much of a particular world in a certain period, it looks promising to make attempts to combining the knowledge collected by different users, to be able to form more complete maps of the world. This can be a difficult problem when the agents do not have a common reference frame. Finding such a common reference frame is greatly simplified when topological maps are used, as they provide a concise description of the world. By topological maps we mean graph-like representations of a world with ordinal distances between graph nodes (Remolina and Kuipers 2004). Another constraint is that our world model will only use angles of exactly 90°, i.e. we will only create and merge rectilinear topological maps.

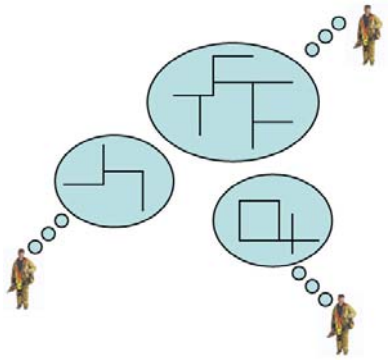

Figure 3. Users have distinct world models

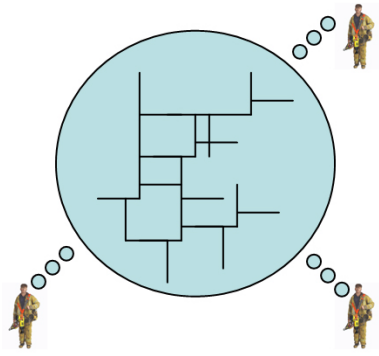

Figure 4. Combined world model

As a proof of concept, we developed a simulation system for MANET, where topological maps of an environment can be created and distributed (Rothkrantz, Van Velden, and Datcu 2005). Our main goal involves storing and distributing of location information based on user observations. The processes involved are primarily based on individual routes and any context information they might gather, such as where exits are, and determining the shortest route to one of them. The goal for each agent node is to construct this internal map of the world by using its own observations and to share it with other nodes. In this way, the knowledge of one node is distributed to the other, nearby nodes. In practice this means that nodes are within communication range. In our case the data shared relate to information about halls and crossings and can be extended to suit the full usage needs.

Related work

In recent years, there has been increased interest on exploring environments and building maps using sensing robots, which in fact is similar to our situation. Many different approaches have been proposed for such problems. Some systems use occupancy maps (Stewart, Fox, Konolige, and Limketkai 2003), some topological mapping (Dedeoglu and Sukhatme 2000; Huang and Beevers 2004), some use robots with many sensory capabilities and some with almost none. Our approach though is that the robot is 'replaced' by a human user. This means there will be different forms of environmental input than when using a robot. Many of the robot-mapping systems depend on the sensors to be very accurate. A human can never be as accurate in measuring distances, angles etc, as most combinations of sensors used in these systems. What a user is able to be accurate enough in (better than the average robot) is mentioning when he encounters an intersection, meaning indicating there is a corridor to the left and one to the right etc. A user is also able to give a rough estimation of distances, which means he indicates one corridor is for example twice as long as the other.

Even though people have different sensing capabilities than robots, it is still possible to use many ideas of robot mapping in our system. As soon we have a collection of consistent partial maps, that have enough data in common, we can match these maps with each other. This is done separate for each floor of the building. If a match is found they can be merged, resulting in more complete maps of the world. Such a matching and merging process can in essence be seen as a form of the maximum common sub graph problem (Bunke and Kandel 2000). It is important to note that to be able to write an algorithm for merging topological maps, the maps are assumed to be consistent (Huang and Beevers 2005). Meaning no two vertices may represent the same place. Not having this constraint would make merging maps virtually impossible and would require different techniques for solving the merging problem.

Human observations about the world can be considered subjective and inexact. The users can have different ages, rapidity, a certain grade of attention and a lot of others characteristics which could influence the way they measure a corridor. The input received by the agents is thus fuzzy, local and probabilistic. Fuzziness in this system means although a corridor has been measured as 100 steps a time t_1 it can be measured 90 steps at t_2. So each element in the graph representation of the environment, vertexes and edges, will have different attributes (or labels (Champin and Solnon 2003)) which will be compared later during the matching process.

TOPOLOGICAL MAP MERGING

In this section we describe the algorithm we used for the map merging for each single floor. We consider the 2D representation of a map as a graph. To get the complete 3D model of the building, the process has to be repeated separately for each different floor.

Constructing the hypotheses graph

The topological maps are represented in our case as graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where V_1 and V_2 are the vertices and E_1 and E_2 are the connecting edges. In order to be able to merge two maps, we first need to match the maps together, building a hypothesis and choosing the correct one (i.e. the best match). A hypothesis is a possibly rotated sub graph that the two maps have in common.

The first step taken into account for map matching is building a list of all vertices that match each other in the two maps. Two vertexes only match if they have the same edge directions. A vertex that needs to be rotated in order to match is also taken into consideration (Figure 5). We distinguish 4 cases: no rotation of G_2, rotate G_2 with 90°, 180° and 270°.

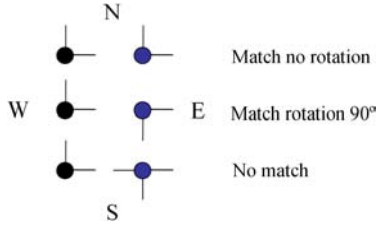

Figure 5. Matching vertexes

So the vertices of the two maps (the agent's map and the one received from another agent) are compared in every of the four possible directions (north, east, west and south) and if they have the same edge directions, they are added in

a list of matches for the chosen rotation. We expect exactly known attribute vertices, such as the type of the vertices to match perfectly. This means, that even if the user can explore only one corridor at the time, being at a crossing, he will indicate the presence of the other unexplored corridors. However, attributes that are subject to measurement error, like the length of edges, can be compared with a similarity test. In the case of our simulation we don't have any fuzzy variables of a vertex, but in a real environment or an extension of our simulation, such variables might occur.

As soon as the list of matching vertices is available we use it to build the hypothesis graph. All the matched pairs will represent a vertex in the set V_H of the graph G_H. The next step is to build the set of edges E_H of the graph. For each pair of vertexes $(u_1, u_2) \in V_H$ (where $u_1 \in V_1, u_2 \in V_2$), we analyze the matches by testing the corresponding pairs of edges (e_1, e_2) (with $e_1 \in E_1, e_2 \in E_2$) leaving the paired vertices u_1 and u_2. If the edges are compatible and the vertices at the ends $v_1 \in V_1$ and $v_2 \in V_2$ are also compatible ($(v_1, v_2) \in V_H$), then the pair of edges (e_1, e_2) is added to the set E_H of the hypotheses graph. Edges may also have both exactly and inexactly known attributes. In our system, they have their path length compared with a similarity test. Next we present the pseudocode for the graph constructing algorithm.

```
for all (u1, u2) ∈ VH do
  for all d in {north, east, south, west} do \\ test the four possible directions
    e1 ← getEdge(E1, u1, d)
    e2 ← getEdge(E2, u2, d)
    if e1 ≠ null and e2 ≠ null then
      v1 ← getOtherVertex(V1, u1, e1)
      v2 ← getOtherVertex(V2, u2, e2)
      if v1 ≠ null and v2 ≠ null then
        if (v1, v2) ∈ VH then
          if compareEdges(e1, e2) then
            add (EH, [(u1, u2), (v1,v2)]) \\ add the edge to the hypotheses graph
          endif
        endif
      endif
    endif
  end for
end for
```

Once the hypotheses graph $G_H = (V_H, E_H)$ is constructed, we determine the resulted hypotheses by computing all the connecting components of the obtained graph G_H. Each connecting component represents a possible hypothesis. From all the four rotations, the best one is chosen. The selection criterion is the number of its components.

The complexity of the algorithm is $O(4|V_1||V_2|)$ for the construction of V_H, $O(4|V_H|)$ for the construction of E_H and again $O(4|V_H|)$ for the hypothesis selection. If we consider $|V_1|=|V_2|=n$, then we have $|V_H| < n$ and the total complexity of the algorithm is $O(n^2)$.

An example

For a better explanation we give a small example. Let consider the two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with the vertexes labeled as in the Figures 6 and 7.

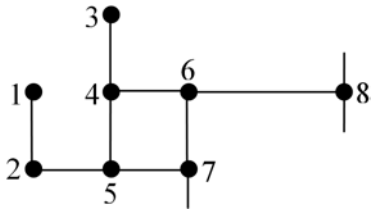

Figure 6. Example graph G_1

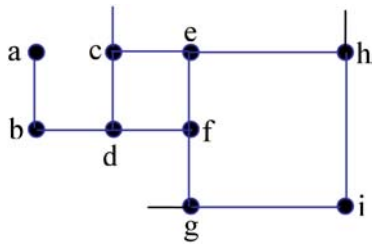

Figure 7. Example graph G_2

The elements of the hypotheses graph $G_H = (V_H, E_H)$ are in case of the rotations with 0° and 270° of the graph G_2:

- no rotation (Figure 8):

$V_H = \{(1,a); (2,b); (3,a); (4,c); (5,d); (5,g); (6,e); (7,f); (7,h); (8,f); (8,h)\}$

$E_H = \{[(1,a),(2,b)]; [(2,b),(5,d)]; [(4,c),(5,d)]; [(4,c),(6,e)]; [(5,d),(7,f)]; [(6,e),(7,f)]; [(6,e),(8,h)]\}$

The five connecting components are: $\{[(1,a); (2,b); (4,c); (5,d); (6,e); (7,f); (8,h)], [(3,a)], [(5,g)], [(7,h)], [(8,f)]\}$.

- rotation 270° (Figure 8):

$V_H = \{(4,e); (5,c); (6,f); (6,h); (7,d); (7,g); (8,d); (8,g)\}$

$E_H = \{[(4,e),(5,c)]; [(4,e),(6,f)]; [(5,c),(7,d)]; [(6,f),(7,d)]\}$

The four connecting components are: $\{[(4,e); (5,c); (6,f); (6,h); (7,d)], [(7,g)], [(8,d)], [(8,g)]\}$.

A sub graph in one map can be matched to multiple sub graphs in the other map under separate hypotheses, but a pair of matched vertices with a given edge correspondence can appear in only one hypothesis. In Figure 9 we see the two cases of possible failure in the hypothesis construction: $(3,h)$ is not a matching pair and the edges $(6,8)$ and (f,g) are not compatible because of the big difference between their lengths.

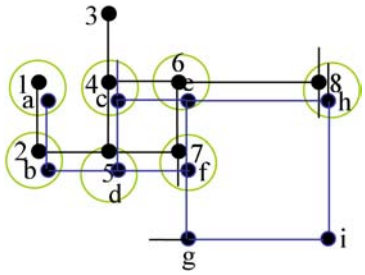

Figure 8. Hypotheses graph construction - no rotation

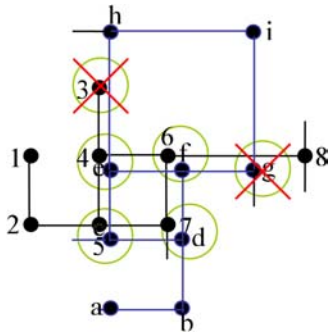

Figure 9. Hypotheses graph construction – G_2 rotated with 270°

So the selected hypothesis in our case is: $[(1,a); (2,b); (4,c); (5,d); (6,e); (7,f); (8,h)]$.

Merging the maps

There is a chance the processes described above do not supply a (large enough – less than 3 elements) hypothesis. If this is the case and not enough vertices can be matched to make a good hypothesis yet, the map received is stored internally. In that case we can try the matching process later on, when a more complete world model is available.

If the process is successful, the next step is to merge (or flatten) the two maps into one single map. Estimates of path lengths can be updated by combining the measurements from the two maps for corresponding edges. The edge orientations at the corresponding vertices can be similarly merged. Parts of one map not present in the other should be added. The merging process is globally performed in four steps (Figure 10):

- Rotate the received map so its orientation matches the local node's map
- Shift the rotated map so its coordinates match the local agent's map
- Add any new vertexes from the rotated and shifted map to the local node's map
- Connect everything together (update edge lengths, check for inconsistency's etc.)

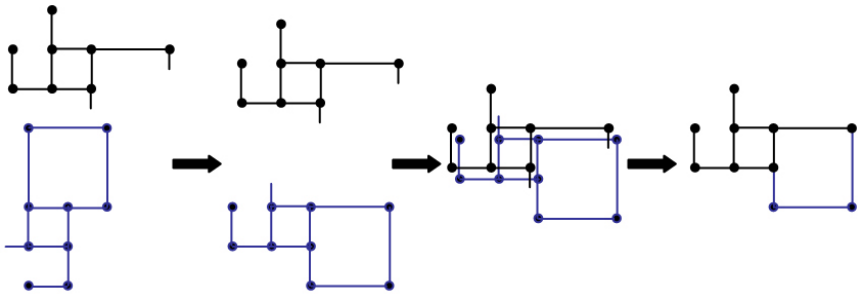

Figure 10. Map merging process

PREVENTING AND RESTORING FROM ERRORS

Closing the loop

If a user would travel long enough in a building, he will eventually return to a location that was visited before from a different direction. The process described, will result in a loop in the graph, which should be detected and closed (Savelli and Kuipers 2004). As an example of closing loops we take an even simpler square shaped world (Figure 11). Applied to the square world the process described before might result in the following 'map', where the two upper-left intersections, should be recognized as one and the same, but at the moment we still have 5 vertices in the graph where there are only 4 intersections in the world.

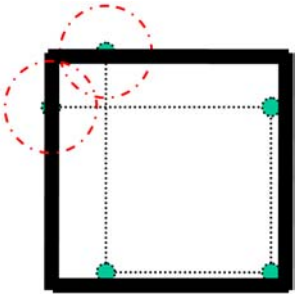

Figure 11. Open loop

A correctly detected closed loop is very valuable information, as it is required to make a map consistent. But before being able to close a loop we should be able to detect and build hypotheses concerning possible loops. For this we roughly follow a procedure that checks for the existence of vertexes next to a given one that might close a certain loop. If we can find two matching vertexes, then a loop hypothesis is started. For our algorithm it is also important to note that a loop always has a minimum of 4 vertices and an even number of turns. To make the graph consistent when two matching vertexes are found, small adjustments can be made to the endpoints of edges. Whenever a loop hypothesis has been formed we can start testing it by comparing edge lengths of the supposed loop with new measurements. These measurements will result in accepting or rejecting the loop.

Recovering from inconsistencies

Please note that the choices made, may later turn out to be incorrect. For example, early in the process of exploring a self-similar environment, a user might seem to be exploring the same area when in fact they are exploring similar but distinct areas. As it is crucial a local map is consistent, this is checked each time a change is made: when a new observation is made, when a loop hypothesis was accepted and closed, and each time two maps have been merged. Merging two maps can lead to inconsistencies, as the selected hypothesis might not have been correct or the map received wasn't entirely correct. By inconsistency we mean that according to the map there should be an edge leaving a vertex in a certain direction, but when the agent arrives at that point, he just notices that there isn't any edge. The conclusion is that the matching was made in a wrong way. Encountering such an inconstancy will in most

cases only lead to merely discarding the changes made for this hypothesis which have not been verified by local observations or other agents thus far. A better way to solve such situations is to record with every node received from other agent, the *id* of that agent and a timestamp when this thing happened. This makes it possible for discovered inconsistencies to be removed or corrected later on without discarding the whole map. The reason why we include also the timestamps is to know when the matching with the map of another agent was computed. If only the *id* of an agent would be stored, finding an inconsistency for a vertex received from that agent would lead to the removal of all the other vertexes received from him, at different moments of time. This is not correct. What it has to be done is to delete only the vertexes received from the same agent at the same time with the wrong vertex. Of course that these vertexes could be related to others, maybe added later to the agent's map. When removing such a vertex from the map, also the corresponding edges have to be updated. After this process, the map might get split in more pieces. Only the one containing the part explored by the user himself will be stored.

As an example have a look back to Figure 2. There we display the graph on user's PDA. The circled vertexes and the doted lines represent elements of the map which have been received by another agent but haven't been yet explored by the current user. So it might be the chance that there was a mistake in the merging process. They will be completely validated only when the user himself will pass that crossings and corridors.

SYSTEM TESTS

A building like environment, where individuals are exploring an unknown map is simulated. Each individual in the field is equipped with a Personal Digital Assistant (PDA) and can communicate with other PDA's which are in his vicinity. All the PDA's form dynamically ad-hoc networks. To make our simulation as realistic as possible, we implemented the IEEE 802.11B MAC Layer (IEEE 802.11 working group 1999) features concerning the series of agreements used for sending and receiving of data.

The simulation was performed on three different maps: one with 18 nodes and 830 m of corridors (Figure 12), one with 30 nodes and 4800 m of corridors (Figure 13) and respectively a map with 100 nodes and 9180 m of corridors (Figure 14). The PDA's transmission range was limited to 160 m. In order to get the fuzzy information the users got different step length, between 0.9m and 1m. Automatic navigation was implemented to simulate the user exploration decisions, looking for the nearest unexplored area. Each test was run for 10 times and the average time it took to an agent to find the complete map was measured.

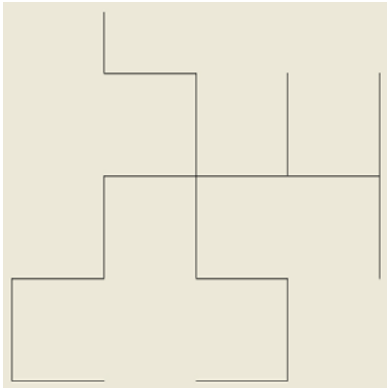

Figure 12. 18 intersections world

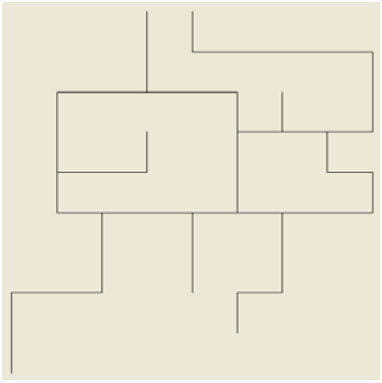

Figure 13. 30 intersections world

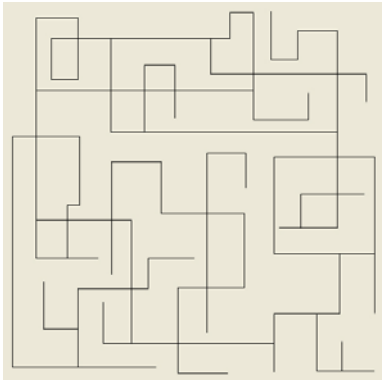

Figure 14. 100 intersections world

The results presented in the Table 1, clearly shows the gain of the process of sharing and merging maps: the larger the map, the larger the gain and also the more agents, the better. Agents that start in a map that was already explored by others, logically, have the most gain; after they have explored a small part of the world they can simply merge the large map parts with their own.

In another test, the 30 intersections world was pre-explored completely by 5 agents and after that a fresh agent was added. The new agent received the complete map from multiple agents and was able to find the complete correct map within 536 steps traveled. Considering it takes 8000 steps on average for an agent to explore this world on its own, this is a considerable gain.

Map	Nr of agents	Min. steps	Max. steps	Avg. steps
18.map	1	1211	1261	1230
18.map	2	1161	1198	1179
18.map	3	895	1244	1065
18.map	4	849	1081	965
30.map	1	7603	8401	8002
30.map	2	5995	7401	6698
30.map	3	5779	6013	5896
30.map	4	5026	5703	5364
30.map	5	4395	5601	4998
100.map	6	10314	12616	11465
100.map	7	9932	11314	10623
100.map	8	9277	10878	10077

Table 1. Test results in finding the complete map

CONCLUSIONS

Our experiments conducted in a simulated world show that it is very well possible to distribute, and merge world knowledge in a mobile ad-hoc multi-agent environment. Even in such an environment with limited communication possibilities our test results showed there is a significant gain found when solving a mapping problem with multiple distributed agents. As expected, the larger the map the better results on how useful distributing and merging partial maps is. We are also now implementing a client server version of our system in order to test our approach in real time and with real data.

It might be necessary to retain in the map of the building, some details, like the position of a door, a window or other significant elements which could be easily observed by an agent. The purpose of using these details is to help the agents avoid the confusion between similar places of the building. If the system has only the representation of the building as a topological map, and the building is a symmetric one, it is very probable for the agents to believe that they have explored the same parts of the building, in the merging process, when instead they have explored different ones, but with the same representation. By using some particular elements, it could be easier for them to make the distinction between similar areas of the building and in this way to improve the matching algorithm.

The knowledge gathered by an agent running on a user's PDA, can be used to provide different services such as the guidance of the user to a certain location or find the nearest exit. In the context of a crisis, such a system can collect information about crisis indicators (fire, smoke, etc), and reason about the state of the building (blocked corridors or locked doors). Having this type of knowledge available in an agent network can be used to coordinate the rescue actions of individuals and groups.

ACKNOWLEDGMENTS

The reported research was performed within the Combined Systems project that runs under the flag of the Delft Cooperation on Intelligent Systems (DECIS) and was partly funded by the Dutch Ministry of Economic Affairs.

REFERENCES

1. Bunke, H., and Kandel, A., (2000) Mean and maximum common subgraph of two graphs, *Pattern Recognition Letters*, 21, 2, 163 – 168.
2. Burghardt, P., (2004) The COMBINED SYSTEMS point of view on Crisis Management, *Proceedings of the First International Workshop on Information Systems for Crisis Response and Management ISCRAM 2004*, Brussels, Belgium.

3. Champin, P., and Solnon, C., (2003) Measuring the similarity of labeled graphs, *Proceedings of the 5th Int. Conf. On Case-Based Reasoning ICCBR 2003*, Trondheim, Norway.
4. Dedeoglu, G., and Sukhatme, G., (2000) Landmark-based matching algorithm for cooperative mapping by autonomous robots, *Proceedings of the 2000 International Symposium on Distributed Autonomous Robotic Systems DARS 2000*, Knoxville, USA.
5. Fitrianie, S., and Rothkrantz, L., (2005) Communication in Crisis Situations using Icon Language, *IEEE International Conference on Multimedia and Expo ICME '05*, Amsterdam, NL.
6. Fitrianie, S., and Rothkrantz, L., (2005) Language-Independent Communication using Icons on a PDA, *Proceedings of 8th International conference on Text, Speech and Dialogue TSD 2005,* Carlsbad, Czech Republic.
7. Huang, W., and Beevers, K., (2004) Topological Mapping with Sensing-Limited Robots, Proceedings of the Sixth International Workshop on the Algorithmic Foundations of Robotics WAFR 2004, Utrecht, NL.
8. Huang, W., and Beevers, K., (2005) Topological Map Merging, *The International Journal of Robotics Research*, 24, 8, 601-613.
9. Ko, J., Stewart, B., Fox, D., Konolige, K., and Limketkai, B., (2003) A practical, decision-theoretic approach to multirobot mapping and exploration. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, LasVegas, USA.
10. IEEE 802.11 working group, (1999) ANSI/IEEE std. 802.11, 1999 edition: Wireless lan medium access control (mac) and physical layer (phy) specifications. *Technical report*, ANSI/IEEE.
11. Remolina, E., and Kuipers, B., (2004) Towards a General Theory of Topological Maps, *Artificial Intelligence*, 152, 1, 47 - 104.
12. Rothkrantz, L., Van Velden, M., and Datcu, D., (2005) A Location based Approach for Distributed World-Knowledge in Mobile Ad-Hoc Networks, *Proceedings of the International Conference on Computer Systems and Technologies ComSysTech'05*, Varna, Bulgaria.
13. Savelli, F., and Kuipers, B., (2004) Loop-closing and planarity in topological map-building, *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan.
14. Tatomir, B., and Rothkrantz, L., (2005) Crisis Management using mobile ad-hoc wireless networks, *Proceedings of the Second International Workshop on Information Systems for Crisis Response and Management ISCRAM 2005*, Brussels, Belgium.