MONITORING SEARCH AND RESCUE OPERATIONS IN LARGE-SCALE DISASTERS

A Multi-Agent Approach to Information Fusion

Alessandro Farinelli, Luca Iocchi and Daniele Nardi
Dipartimento di Informatica e Sistemisica, Università di Roma "La Sapienza," Via Salaria 113, 00198 Roma, Italy

Abstract: The present contribution is concerned with designing tools to monitor a situation after a large-scale disaster, with a particular focus on the task of high-level Information Fusion within a multi-agent approach. The Multi-Agent System is based on the RoboCup-Rescue simulator: a simulation environment used for the RoboCup-Rescue competition, allowing for the design of both agents operating in the scenario and simulators for modeling various aspects of the situation, including a graphical interface to monitor the disaster site. The design of a Multi-Agent System with planning, information fusion, and coordination capabilities is described according to the agent model underlying the Cognitive Agent Development Toolkit, which is one of the outcomes of our recent research. Finally, we discuss the issues related to the evaluation of the performances achieved by Multi-Agent Systems in search and rescue operations according to the proposed approach and discuss some results obtained in a case study relative to the Umbria and Marche earthquake of 1997.

Key words: multi-agent systems, information fusion, search and rescue

1. INTRODUCTION

The analysis of the problems encountered in rescue operations concerning catastrophic events, specifically earthquakes, has driven the attention of the international scientific community towards the research on tools and techniques that can improve the effectiveness of rescue aids. Both in Japan and in the USA the problem has given rise to specific research programs (see for example [1, 2]).

One major goal of these researches is the realization of tools supporting search and rescue operations, to be used in provisional analyses, training of personnel and field operations. The achievement of this research goal involves the methods and techniques for cooperation in a scenario with multiple heterogeneous autonomous agents. In this context, the RoboCup Rescue initiative [3] supported the realization of the RoboCup Rescue simulator [4], a basic framework for the simulation of the post-earthquake scenario, in which user designed agents interact with the simulated environment to search and rescue the victims of the disaster. The RoboCup Rescue simulator, which was initially designed for the Kobe earthquake scenario, integrates three major components: (1) modeling of the events, activated directly, or indirectly, by the main catastrophe, (2) acquisition and integration of data coming from heterogeneous sources, (3) operation resource monitoring/modeling/planning. The simulator thus provides a framework based on a multi-agent approach (see for example [5, 6]), where Information Fusion plays a significant role. In fact, according, for example, to the reference model for Information Fusion proposed in [7], the process of Information Fusion is concerned with different types of data processing that are required to obtain different kinds of information about the environment: data assessment, object assessment, situation assessment, impact assessment and process refinement.

The goal of the present paper is to provide an introduction to the multi-agent approach to Information Fusion, by adopting the RoboCup Rescue simulator as an experimental framework. The use of the RoboCup Rescue simulator requires the design of a Multi-Agent System based on an autonomous agent model (see for example [8]). Specifically, we propose an agent model which combines planning, information fusion and coordination capabilities. Such a model is the basis of a tool specifically developed at our University to design cognitive agents for many applications and, in particular, search and rescue operations.

A research issue relevant to the multi-agent approach to Information Fusion is the need for suitable frameworks and methodologies to evaluate the performance of the system. In fact, the inherent complexity of the tasks that a system is expected to perform, makes it very difficult to apply standard techniques for testing, validation and measure of performance. The RoboCup Rescue simulator provides a synthetic simulation scenario, which is suitable for the development of evaluation techniques of multi-agent approaches [9]. In the paper, we address the evaluation of performance of MAS in search and rescue [10].

Finally, the RoboCup Rescue simulator has been adopted in research projects carried out at our University [11, 12] to address a specific case study relative to the Umbria and Marche earthquake (1997).

The paper is organized as follows. In the next section we provide some background on Multi-Agent Systems and Information Fusion; in Section 3 we sketch the main features of the RoboCup-Rescue simulator; in Section 4 we describe the design of the agents and their implementation; in the last section we address the experimentation carried out on the mentioned case study and discuss some outcomes of this experience and possible future developments.

2. BACKGROUND AND APPROACH

2.1 Multi-agent systems

In the last ten years, agents have become a very important research topic. Many authors often consider agents, with respect to the software design process, as the descendants of objects: they both encapsulate a state and rely on the concept of message exchange. Developing software by making use of agent techniques is also referred to as Agent Oriented Programming [6]. While it is hard to provide a precise and generally accepted definition of Agent, it is generally agreed that there are some important features an agent must exhibit, such as autonomy in its actions and behaviors, the ability to cooperate with other agents and/or users, the capability to learn from other agents and/or users and the ability to “Understand” the environment, acquiring and maintaining knowledge about it (see for example [8, 5, 6]).

The main focus for Multi-Agent Systems (MAS) is the study of interactions that arise among active entities which are co-located in the same working environment. The interest in MAS is due both to theoretical and practical issues: from a theoretical viewpoint many researchers argue that the same concept of intelligence as we normally intend is deeply related to interacting systems, and even more, that interaction is a necessary property for an intelligent system, mainly because intelligence is not an individual characteristic [5]. From a more practical perspective, the problems that we have to face in real world applications are often physically widely distributed (air traffic management, space applications); moreover, the complexity of such applications often requires combining local solutions to smaller sub-problems. Finally, from a software engineering perspective, designing solutions based on the interaction of specific autonomous modules entails several nice properties for the developed system, such as modularity, flexibility and re-usability.

MAS are widely applied in several areas addressing different practical problems. One significant application of MAS is the simulation of synthetic scenarios, which require the software agents to act in the simulated environment and interact with other entities in a way that resembles, as closely as possible, real life. Human support in hard real time, dangerous situations, such as disaster response or military applications, is becoming more and more interesting from the viewpoint of MAS applications. In such domains the main focus is to coordinate several agents involved in the operations facing the hard-time constraints imposed by the application.

In the following sections of this paper we will primarily focus on cooperative environments, where agents have the same goal and are not self interested. Moreover, we will consider dynamic and unpredictable environments, in which agents are embedded entities and therefore do not have a global vision of the environment. In such domains, Information Fusion becomes a fundamental issue in order to provide efficient planning/execution of activities and effective coordination among the cooperating agents.

2.2 Information fusion

Information Fusion is also a research area that has been growing significantly in the past years (see for example [7, 13, 14]). In particular, Information Fusion is broadening the scope of Data Fusion, by addressing complex dynamic scenarios, where the knowledge that enables decision making is gathered by a number and variety of different sources. Information Fusion is broadly used in various application fields such as defense, geosciences, robotics, health, industry and many techniques have been developed for the fusion process. In the following, we briefly address two aspects that are strictly related to MAS: the levels of the fusion process and the architecture of the fusion system.

2.2.1 Fusion levels

The structure and the nature of the fusion process are thoroughly investigated by the literature concerning Information Fusion. The main notion in this respect is the *level* of fusion. Basically, data can be fused at three levels (see [14]): *signal*, *feature* and *symbol* level.

The choice of fusing data at a certain level is strictly related to the purpose and characteristics of data fusion and strongly influences the techniques adopted for the fusion. When the fusion is made at symbol level, the objects to be combined are a logical representation of data; therefore to fuse data at the symbol level in most cases logical operators are used. When the fusion is made at signal level, data are directly extracted from sensors and no intermediate representation is given; the techniques used to fuse this kind of data are taken from signal processing. Finally, feature level fusion involves some kind of preprocessing of raw data; aggregating raw data in some kind of higher-level representation allows to avoid noisy data coming from the sensors measurement and to reduce the data communication payload. Several efforts have been made to approach the problem of Information Fusion focusing on the level of fusion, and to integrate different levels of Information Fusion in coherent architectures [15]. In a MAS approach, Information Fusion can be performed at different levels depending on the chosen representation.

In the synthetic scenarios addressed in this paper, feature and symbolic levels are normally considered, while in other applications, such as sensor networks [16, 17], the signal level may be relevant.

2.2.2 Fusion architecture

The design and development of system architectures is a central issue in the design of a fusion system. The architectures proposed in the literature may be grouped into three categories: Centralized, Hierarchical and Distributed.

The centralized approach to the development of fusion architectures is the most popular in the literature due to the fact that centralized fusion can be characterized as a well-defined problem. Data collected from all the sensors are processed in a single central unit that performs the fusion task. This approach is optimal when there are no communication problems (bandwidth, noise) and the central unit has enough computational resources to perform fusion of data. Most fusion algorithms have been developed for the centralized fusion architecture, and many applications have been realized using this approach.

However, in recent years, distributed and hierarchical approaches are becoming more popular, due to the wider availability of communication technology. Hierarchical fusion architectures are based on different layers of nodes: at the lowest layer, fusion nodes collect data from sensors to perform a first fusion process on these data, sending their results to a higher layer of fusion nodes. Each of the higher layer nodes collects the results of fusion from lower layers and performs a different fusion process among them. The overall architecture can be seen like a tree where each node is a fusion node and the leaves of the tree are the sensors (see for example [18]).

Finally, distributed architectures differ from hierarchical ones in the topology of the fusion nodes. Also in this case, each node performs locally a fusion process and sends the results to other fusion nodes. However, in distributed architectures there are no hierarchical layers, but each fusion node can communicate with every other fusion node. The connections are thus arbitrary and the overall architecture can be represented as a graph of fusion nodes. A distributed approach to Information Fusion is very frequent in agent-based systems, see for example [16, 17]. As compared with the centralized ones, distributed and hierarchical architectures have the following advantages:

– lighter processing load at each fusion node, due to the distribution over multiple nodes;
– lower communication load, due to the reduced amount of data to be communicated;
– faster user access to fusion results, due to reduced communication delay.

On the other hand, distributed or hierarchical architectures require the development of fusion algorithms that are specialized for those architectures.

3. A SIMULATION ENVIRONMENT

A simulation environment for emergency scenarios is an extremely useful tool for evaluating emergency plans and for decision support. In a recent research project [12] we have analyzed the tools available to the Italian Fire Department: while they are up-to-date with the current technologies, they suffer from some limitations. First, the tools used are often realized as closed applications: they are not integrated with similar applications, or other kinds of systems, which bring relevant information to plan rescue actions. Moreover, the tools that the current technology offers to plan interventions are mainly static and they are not designed to continuously acquire information about changes in the situation and, thus, they are not able to supply the desired support where the dynamics of the events need a continuous update both of the strategy and of the intervention plan. Finally, another technological issue is the difficulty of integrating different provisional models connected to different phenomena, such as fire, house and building damages, disruption of roads, electricity, water supply, gas and other infrastructures, movement of refugees, status of victims, hospital operations, etc.

For some of the above-mentioned situations simulators have already been developed, but most of them are not dynamic and, most of all, they are not integrated. For example, existing fire simulators model this process as stochastic heat propagating and fire catching with threshold functions over static terrain, but damages of buildings and effort of fire-fighting are not considered; the few simulators which can predict road blockage are not coupled with other simulators; the simulators that, using Artificial Life approaches, model the behavior of refugees, use a static knowledge of the terrain, while a dynamic one should be considered (the disaster continuously changes the ground surface). For all these reasons, only a comprehensive simulator for large-scale disaster rescue makes it possible to compare numbers of different approaches, so that strategies and tactics, which best improve the quality of the intervention, can be chosen.

A simulator with the above-mentioned features is a fundamental tool to develop real-time systems, which allows for monitoring and planning rescue operation. The availability of the RoboCup Rescue simulator [19] represents a great chance to develop and apply the methods and techniques for action planning in a coordinated MAS, referring to a real context much larger in size and complexity with respect to those considered until now.

The RoboCup-Rescue simulator has a distributed architecture, formed by several modules, each of them being a separate process running in a workstation on a network. For a detailed description of the simulator the interested reader can refer to [4].

Below, we summarize the main components of the simulator:

– *Geographic Information System* - The GIS module holds the state of the simulated world. Before simulation begins, it is initialized by the user in order to reflect the state of the simulated area at a given time, then it is automatically updated at each simulation cycle by the kernel module;

- *kernel* - This module is connected to any other module. At each step it collects the action requests of the agents and the output of the simulators, merging them in a consistent way. Then the kernel updates the static objects in the GIS and sends the world update to all the connected modules;
- *simulators* - Fire-simulator, Collapse-simulator, Traffic-simulator, etc. are modules connected to the Kernel, each one simulating a particular disaster feature (fire, collapses, traffic, etc.). At the beginning of every simulation cycle, they receive from the kernel the state of the world; then, they send back to the kernel the pool of GIS objects modified by the simulated feature (for example, a pool of burned or collapsed buildings, obstructed roads, etc.);
- *agents* - agent modules are connected to the kernel and represent "intelligent" entities in the real world, such as civilians, police agents, fire agents, etc. They can do some basic actions, such as extinguishing a fire, freeing obstructions from roads, talking with other agents, etc. Agents can also represent non-human entities: for example they can simulate a police station, a fire station, an ambulance-center, etc;
- *viewers* - their task is to get the state of the world, communicating with the Kernel module, and graphically displaying it, allowing the user to easily follow the simulation progress.

In order to use the RoboCup-Rescue simulator two aspects must be considered: 1) the model of the area on which we want to run disaster simulations; 2) the model of the agents involved in the intervention. These two issues are described in the next paragraph, while the next section describes in detail a framework for developing MAS in this environment.

3.1 World model

The world model adopted in the RoboCup-Rescue simulator is somewhat minimal, but it could be easily extended to fit real scenarios more closely. It deals with three main entities: *buildings*, *roads* and *nodes*. The road network is described by a graph having one or more edges for each road and one node for each crossroad and for each junction between adjacent edges constituting a road. Also, a node can represent a linkage point (*access-point*) between a building and a road. Each object class (*building*, *road*, *node*) is characterized by a number of attributes describing a specific instance of the class.

Building objects represent every kind of building on the map: houses, police offices, hospitals, fire stations, ambulance centers, refugees, etc. As a result of an earthquake shock, a building can collapse and obstruct a road; moreover, a building is more or less likely to catch fire according to its constituent material; for example, a concrete building is less flammable than a wooden one. Further, buildings can have one or more floors and one or more linkage points with the surrounding roads. The main attributes of buildings are: Plant, Kind, Material, Fieriness, Brokenness, Floors, and Entrances.

Road objects are the edges of the road network graph; they represent every street, lane, tunnel, bridge, etc. in the map. A road can be partially or totally obstructed by rubble, as a consequence of the collapsing of an adjacent building. Furthermore, a road has one or more traffic lanes on each side and can have a sidewalk or not. The most relevant road attributes: Kind, Length, Width, Block, Repair-Cost, Lines-to-head/Lines-to-tail, Sidewalk-width.

Nodes represent crossroads or linkage points between buildings and roads. Moreover, a whole road can be split into two or more adjacent edges, connected to the others by nodes. The following are the most important attributes of this class: Roads, Signal, and Signal-timing.

In order to adapt the simulator to a generic intervention area a GIS Editor [12] allows the user to define an initial rescue scenario by specifying both the static structure of the environment (e.g. buildings, roads, etc.) and dynamic information about the initial status of the world (e.g. burning buildings, blocked roads, etc.). While editing a map it is possible to generate inconsistent situations, therefore the GIS editor can also perform checks in the edited map and warn the map designer about possible inconsistencies.

3.2 Agents

Agents modeled within the RoboCup-Rescue simulator are of four kinds: *firefighting* agents in charge of extinguishing fires; *medical* agents, that bring injured civilians to hospitals; *police* agents that are in charge of removing road blockages or solving traffic jam; *civilians*, that can be injured by fire or collapsing building and may need assistance.

In addition, firefighting, police and medical agents can also be modeled with a central station that acquires information from the agents and distributes directives on how to perform in the environment.

The definition of a MAS, that represents the behavior of agents during the intervention, is an important process that the user has to accomplish in order to evaluate intervention plans in the modeled area. This process can be supported by a design and implementation framework, which is described in Section 4.

Moreover, in order to evaluate the behavior of the MAS during search and rescue operations, evaluation methodologies that measure the *goodness* of the agent's activities are used. These measures are important for evaluating performances of intervention plans as well as for comparison of different approaches to a specific problem, e.g. information fusion, planning strategies, etc.

For example, the evaluation of the general behavior of the agents is given by a formula that computes the number of civilians saved, the number of fires extinguished and the number of roads cleaned from debris. This formula allows for evaluating the effectiveness of a MAS operating in a scenario and is currently used during the RoboCup-Rescue competitions. A more detailed description of evaluation methods for such systems is given in Section 5.

Another important issue in a rescue scenario is the reconstruction of the status of the environment from the information acquired by the agents. In order to evaluate such a process, we make use of "Knowledge Viewers," graphical tools that show the knowledge that a set of agents has about the environment, which is usually different from the real status of the world. This visualization is extremely useful for monitoring the evolution of the knowledge acquisition process during the mission and for evaluating the techniques of Information Fusion that have been implemented.

4. DEVELOPMENT OF MAS FOR SEARCH AND RESCUE

The development of MAS for search and rescue operations can be effectively performed by taking advantage of a design and implementation framework for MAS development. The Agent Development toolkit (ADK) [20] is a tool for implementing agents for the RoboCup-Rescue simulator. Such a tool hides the agent-simulator communication details from the user.

The framework presented in this section represents an extension of the ADK, by providing planning, cooperation and information fusion capabilities. Specifically, the Cognitive Agent Development Kit (CADK) [21], briefly described here, allows the users to design and implement agents acting in a dynamic environment for accomplishing complex tasks. The agents developed with his tool have the following characteristics: (i) they can act autonomously in the environment by selecting the actions to be performed according to the information acquired in the environment; (ii) they can communicate with each other and cooperate to achieve a common goal; (iii) they can exchange information about the environment in order to reconstruct a global situation by using appropriate information fusion techniques.

All the agents are realized with three fundamental components:

1. *Plan Executor*, that is responsible for executing a plan (i.e. appropriately execute elementary actions that the agent can perform) for accomplishing a given task; plans are stored in a plan library and can be generated by an automatic planner, as well as by using a graphical tool; the actual plan to be executed depends on the information coming from the *Coordination Manager* which is described below;
2. *Information Integrator*, that is in charge of applying a technique for fusing the information about the world coming from its own sensors and from communication by other agents;

3. *Coordination Manager*, that is responsible for analyzing the current world state and the other agents' coordination information, and choosing the agent specific goal (and thus the corresponding plan) in order to achieve a global goal for the team; the coordination protocol is distributed and thus it is robust to network failures and allows the agents to remain autonomous.

In the following, we describe these components and their relations in more details.

The agent architecture uses a *world* structure that represents the whole agent knowledge about the world. In order to grant consistency it can only be modified by the *Information Integrator*, which updates the world in a sound way, with respect to the updating sources. For instance, in the rescue domain the world is composed by the rescue world objects such as roads, agents, buildings, etc., plus some information about the agent state. At each instant the world contains the result of the integration of the information received since the starting time, and, hence, it contains the knowledge for decision making.

In order to acquire information from the environment an agent can be equipped with a large variety of sensors and/or sensing capabilities, each one providing a different type of input. The task of the information integrator is to calculate the new state of the world starting from the previous states and the incoming information. As previously mentioned, there are different levels at which the integration can be performed, depending on the properties of the data being integrated. In our approach, Information Fusion is performed at symbol and feature level, in terms of properties of the world objects, since the RoboCup-Rescue domain, in which we tested the system, is well suited for this kind of data.

According to the situation at hand an agent must decide which actions must be performed. To this end we make use of the notion of *plan*. A plan may be seen as a graph that specifies the actions an agent has to perform in order to reach a goal. Each node is a state, while edges specify the state transitions caused by action execution. A state represents a particular world configuration in terms of properties that are true in the current situation. In order to handle both action failures and dynamic environments (where some world property can change in a way that does not depend on the agent), the states have to be considered as *epistemic*. An epistemic state represents the knowledge of the agent, not the real world status. An agent can act properly, when its epistemic state matches the world configuration; this can be achieved by the introduction of particular kinds of actions that increase knowledge, and allow for conditional plans as described in [22]. The plan library is a collection of plans; each of those plans is indexed by its goal. These plans may be generated either by an automatic planner or directly specified by means of a graphical tool.

The *Plan Executor* performs two tasks:

1. it receives from the Coordination Manager the goal to be reached and peeks into the plan library the plan to achieve it;

2. it executes a plan by starting or stopping primitive actions at each state, finding which edge to follow in case of a conditional branch, according to the world status. A plan switch can occur either after the recognition of a plan failure or a when the goal is changed by the coordination module.

Finally, coordination among the agents is a fundamental issue and there are many schemas that can be adopted in coordinating a MAS. It is possible to perform a fully distributed approach as well as a centralized one, or to combine both leading on a hybrid coordination schema.

In our architecture we allow both a hybrid and distributed coordination approach. Hierarchical coordination is obtained by the use of messages (commands) sent to a low level agent from a higher-level one. The commands instruct an agent on its particular goal. The high level agent performs the task allocation on the basis of the owned information and the knowledge about the usage of resources. Distributed coordination is achieved by a distributed coordination protocol that allows for a dynamic assignment of roles to the members of the team.

5. EXPERIMENTS AND EVALUATION

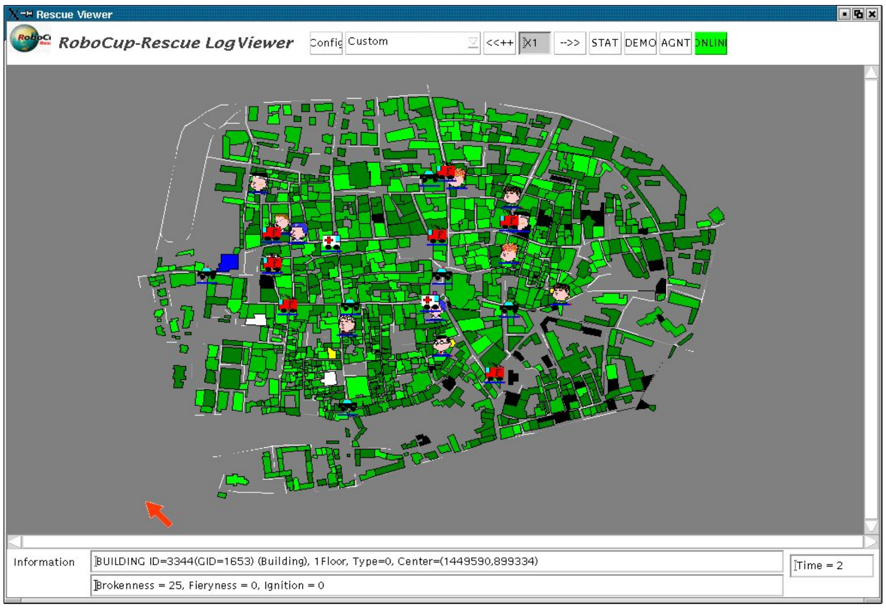
This section provides a description of a typical use of the RoboCup-Rescue environment for performing experiments in a rescue scenario. Specifically, we describe a set of experiments performed in the rescue scenario of the city of Foligno, that has been derived from actual data from an earthquake which occurred in that region in 1997 (see also [12]).

The initial situation contains information about the magnitude of collapses, burning buildings, blocked roads and the position of the agents (civilians and rescue agents). In Figure 1 we show a 2D visualization of one initial situation as provided by the standard viewer.

This initial situation is not completely known to the central stations, they only know the map of the city and the initial position of all its agents. Moreover, the central stations communicate with each other and perform information integration and cooperation in order to collect all available information from the environment, reconstruct a global view of the environment, remove debris from the streets, extinguish fires, and rescue the civilians.

In order to evaluate the performance of a rescue system, we have defined a methodology [10], which considers not only their efficiency under normal conditions, but also their robustness under nonstandard operative circumstances, which often occur in emergency situations.

To acquire a measure of the efficiency and the robustness of a MAS, a series of simulations have been executed by varying operative conditions. These tests give a measure of the system adaptability to unexpected situations. The parameters that we have considered for variation are: (i) perception radius; (ii) number of agents; (iii) errors in the communication system. Each parameter characterizes a particular series of simulations, referred to as the *visibility test*, the *disabled agents test* and the *noisy communications test*, respectively.



Information
BUILDING ID=3344(GID=1653) (Building), 1Floor, Type=0, Center=(1449590,899334)
Brokenness = 25, Fieryness = 0, Ignition = 0
Time = 2

Figure 1. The map of Foligno city in the RoboCup Rescue simulator

The performance of a rescue MAS is measured in terms of efficiency and reliability. The efficiency is directly evaluated by the formula used in RoboCup-Rescue tournaments (2003), that takes into account the number of living agents, the remaining hit points (health level) of all agents, and the area of houses that are not burnt. The reliability describes how much system efficiency is affected by the variation of operative conditions and is evaluated with a linear regression slope formula.

Measures of efficiency and reliability of a single MAS are of little significance if not compared with the results obtained from simulations of other rescue systems. Performance comparison allows one to establish the effectiveness of a new technique over the previous ones, or over the state-of-the-art. Rarely, in these tests the same rescue system is best for both measures, since usually, sophisticated techniques that improve efficiency turn out to be less robust to nonstandard operative conditions.

Therefore, the optimality of the system is hard to cast in absolute terms. Depending on the application, the system, which offers the best score with respect to efficiency, reliability, or to a (weighted) combination of the two, may be selected. Indeed, the choice of a measure to select the best approach is a non-trivial task.

We conclude the paper with a few comments on the development of tools to support search and rescue operations in large-scale disasters. The approach described in this paper shows a significant use of multi-agent technology to support the acquisition of information as well as the planning of activities, when there is the need to act immediately with partial information about the situation, as in a typical emergency scenario. The RoboCup-Rescue simulator has been the basis for a prototype implementation, which is currently designed for demonstration and is not intended for actual operation. However, it has the merit to show the potential benefits of an integrated approach to the simulation and monitoring of a real search and rescue scenario. While it is premature to consider the effectiveness of the tool in the management of operation, both the analysis of past scenarios as well as the training of personnel seem to be already suitable for application.

ACKNOWLEDGEMENTS

This research was supported under contract number F61775-01-WE-030 of the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, United States Air Force Research Laboratory and under the programme Agenzia 2000 of the Italian National Research Council.

We also gratefully acknowledge the technical support of the RoboCup Rescue Project.

REFERENCES

1. J. Llinas. “Information fusion for natural and man-made disasters.” *Proc. of IEEE Int. Conf. on Information Fusion*, AnnaPolis, July 2002.
2. H. Kitano, et al. “RoboCup Rescue: Search and Rescue in Large Scale Disasters as a Domain for Autonomous Agents Research.” *Proc. of IEEE Int. Conf. on System, Man, and cybernetics (SMC99)*, Tokyo, 1999.
3. S. Tadokoro, et al. “The RoboCup Rescue Project: a multi-agent approach to the disaster mitigation problem.” *Proc. of IEEE International Conference on Robotics and Automation (ICRA00)*, San Francisco, 2000.
4. Robocup Rescue Web Site. http://robomec.cs.kobe-u.ac.jp/robocup-rescue.
5. J. Ferber. “Multi-Agent Systems.” *Addison-Wesley*, 1999.
6. M. Wooldridge. “An Introduction to Multi-Agent Systems.” *John Wiley and Sons,* 2002.
7. J. Salerno. Information fusion: a high-level architecture overview. *Proc. of IEEE Int. Conf. on Information Fusion*, AnnaPolis, July 2002.
8. S. Russell and P. Norvig. “Artificial Intelligence: a Modern Approach,” *Prentice Hall*, 2003.

9. G. A. Kaminka, L. Frank, K. Arai and K. Tanaka-Ishii. “Performance Competitions as Research Infrastructure: Large Scale Comparative Studies on Multi-Agent Systems.” *Journal of Autonomous and Multi-Agent Systems*, 2002.
10. A. Farinelli, G. Grisetti, L. Iocchi, S. Lo Cascio, D. Nardi. “Design and Evaluation of Multi-Agent Systems for Rescue Operations.” *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS'03)*, 2003.
11. F. D’Agostino, A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi. “Monitoring and Information Fusion for Search and Rescue Operations in Large-scale Disasters.” *Proc. of IEEE Int. Conf. on Information Fusion*, AnnaPolis, USA, July 2002.
12. D. Nardi, A. Biagetti, G. Colombo, L. Iocchi, and R. Zaccaria. “Real-time planning and monitoring for search and rescue operations in large-scale disasters.” *Technical Report DIS-15-2002*, University "La Sapienza" Rome, 2002. Available at http://www.dis.uniroma1.it/~rescue/.
13. D.L. Hall and J. Llinas (Eds.). “Handbook of Multisensor Data Fusion.” *CRC Press*, 2001.
14. L. Valet, G. Mauris and P. Bolon. “A statistical overview of recent literature in information fusion.” *IEEE Aerospace and Electronics Systems Magazine*, 16(3): pages 7 -- 14, 2001.
15. B.L. Ferrell, J.L. Cruickshank, B.J. Gilmartin, C. Massam, S.J. Fisher and F.D. Gass. “Case study methodology for information fusion interface definition.” *Aerospace Conference, 2001, IEEE Proceedings.* Volume: 6, pages 3003 -- 3015, 2001.
16. A. Knoll and J. Meinkoehn. “Data fusion using large multi-agent networks: an analysis of network structure and performance.” *Proc. of Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI '94)*, pages 113 -- 120, 1994.
17. V. Regis and W. Thomas. “The soft real-time agent control architecture.” *Technical Report 02-14*, University of Massachussets, 2002.
18. Du Qingdong, Xu Lingyu and Zhao Hai. “D-s evidence theory applied to fault diagnosis generator based on embedded sensors.” *Proc of the IEEE Int. Conf. on Information Fusion*, 2000.
19. A. Farinelli, G. Grisetti, L. Iocchi, S. Lo Cascio, D. Nardi. Using the RoboCup-Rescue Simulator in an Italian Earthquake Scenario. *Proc. of the 1st Int. Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*, Padova, Italy, July 2003.
20. Micheal Bowling. “Robocup rescue: Agent Developement Kit.” Available at RoboCup-Rescue Web Site, 2000.
21. A. Farinelli, G. Grisetti, L. Iocchi, D. Nardi. Design and Implementation of a Rescue-Agent Development Tool. *Proc. of Int. Workshop on Planning and real-time monitoring of rescue operations*, Foligno, Italy, October 2002.
22. Luca Iocchi, Daniele Nardi, and Riccardo Rosati. Planning with sensing, concurrency, and exogenous events: logical framework and implementation. In *Proc of the Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*, Boston, USA, 2000.