Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 76 (2015) 539 – 546

Procedia Computer Science

2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)

Predictive Planning Method for Rescue Robots in Buildings

Szymon Szomiński, Zbigniew Kaleta, Wojciech Turek*, Krzysztof Cetnarowicz

AGH University of Science and Technology, Krakow, Poland

Abstract

Rescue robotics is a domain with great potential for important, real-life applications. Recently its development mostly focused on creating machines able of moving in collapsed buildings remains, leaving group operations in undamaged buildings slightly neglected. In this paper we are focusing on methods for managing multi-robot rescue actions in buildings, where some damages should be expected, but the general structure of a building remains unchanged. We propose a predictive planning approach, which allows using time demanding optimization algorithms for the task, where environment can change rapidly. The proposed solution predicts possible exceptional situations and pre-calculates alternative plans. A prototype system for parallel calculating of plans using high performance computing hardware is presented. A comparison of the optimized plans with a greedy approach is shown for different buildings and different sizes of robot teams. Finally the evaluation of the approach on real mobile robots is described.

Peer-review under responsibility of organizing committee of the 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)

Keywords: mobile robots, rescue robotics, multi-robot planning

1. INTRODUCTION

The concept of supporting rescue actions with mobile robots have been studied since early eighties, developing from ideas to real-life applications [12]. Use of robots supporting humans in zones of various disasters can reduce the number of endangered staff and the time required for finding victims. Moreover, it allows accessing areas which cannot be examined by humans.

The research in this area focuses mostly on serious catastrophes like building collapses after explosion or earthquake. Such events occur very rarely, which in general is a positive fact. However, this causes little chance to verify developed solutions in reality. In her recently published book [12] professor Robin Murphy describes all 34 deployments of robots in real rescue actions over the last decades. Variety of situations and problems which can arise in such situations makes the development of useful platforms a very complex task. Significant effort is put into research on basic abilities of a single robot, which are useful in the considered task. These include victims detection [4], maneuvering in complex terrain [10], remote-control methods [8] and human-robot interaction [11].

* Corresponding author. Tel.: +48-12-328-33-31.
E-mail address: wojciech.turek@agh.edu.pl

Peer-review under responsibility of organizing committee of the 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)
doi:10.1016/j.procs.2015.12.340

It seems that fully autonomous robots, operating in heavily damaged places, will require many years of research and experiments, before becoming reliable and valuable members of rescue teams. On the other hand, there are many different emergency situations, requiring less specialized features and abilities, where rescue operations could also benefit from robots support. Building fires, gas leaks, chemical contamination or bomb alerts happen more often and require locating and evacuating people from complex, but undamaged indoor environments. This class of problems seem to receive far less attention, which limits possibilities for real-life successes of rescue robots.

The work presented in this paper focuses on rescue actions in buildings, which have not been significantly damaged. The task is to locate people in a known or partially known building, where robots can move similarly to normal indoor environments. Finding people requires visiting all places where people may remain, which is a task similar to exploration. We focus on the problems of effective planning of actions for groups of robots with incomplete knowledge about the structure of the environment.

The considered planning problem of finding time-optimal paths for several robots, which cover all rooms in a building, is computationally complex. The incompleteness of initial knowledge and possibility of detecting unexpected changes in the environment during system's operation requires the ability to quickly adopt to the situation. These two features of the undertaken problem make a simple replanning-based approach unsuitable.

The approach presented in this paper explicitly assumes that plan's execution may be disturbed by detected changes or robots malfunctions. The planning algorithm is therefore multi-variant – it tries to predict possible modifications of the situation and prepare solutions in advance. The details of the algorithm are provided in the third section, after a brief review of the existing planning methods used in similar problems. Following sections present planning quality evaluation and results of the experiments with real robots.

2. RESCUE ACTIONS PLANNING

The need for rescue robots actions planning was identified more than 25 years ago by the group of professor Satoshi Tadokoro, who proposed the frames of the RoboCup Rescue competition[5]. The problems faced by the domain included multi-agent planning, understood as the need for planning actions thousands of physical agents performing search operations. The scale of the rescue action was driven by the type of disaster the authors considered – their research was motivated by the earthquake in Kobe, Japan in 1995, where more than 100,000 buildings collapsed. The authors predicted the need for real-time and anytime planning methods, which were to provide rescue actions plans on demand, in reaction for detected situation.

The research conducted within the RoboCup Rescue league went into slightly different direction. The competition aims at testing real robots in specially crafted environments mimicking urban disaster zones. Therefore most of the work focuses on single robots abilities. The work considering teamwork focuses on collaboration of teams of robots and human operators. In[13] experiments on efficiency of such teamwork are presented. The evaluation leads finding to optimal human-operator-to-robot ratio for particular types of robots.

Summary of efforts focusing around the the RoboCup Rescue league and more general problem of Robotic Urban Search and Rescue is presented in[7]. The authors note that despite 25 years of research in the area, majority of solutions is based on remote operation or at lease remotely supported decision making. Several mentioned solutions, which make use of groups of robots, also do not offer automated actions planning.

Cooperation between robots in a team has been considered in a few projects focusing on ad-hoc network building. The work presented in[14] undertakes the problem of limited range and quality-of-service in a communication network build between several rescue robots.

The problem of planning a search action for several robots in a building is similar to different problems considered in the domain of mobile robotics. In the before-mentioned survey[7] the authors also analyze some solutions designed for the task of Multi-Robot Exploration (MRE) and Multi-Robot Task Allocation (MRTA).

The MRE problem considers exploration of an unknown building. A robot or a group of robots is supposed to cover every part of a building with sensor measurements, which is similar to the task of rescue robots searching for victims. Solutions to this problem are mostly based on frontier calculation and following. The frontier is defined as the border of visited and uncovered space. The character of the problem – lack of knowledge about task size – makes advanced planning excessive. Satisfactory solutions can be achieved using autonomous or local decisions

making. Both algorithms compared in[2] follow the assumption, that "local dispersion is a natural way to achieve global coverage", giving good results in simple environments.

The assumptions of the MRTA problem are most similar to the robot rescue actions planning considered in this paper. In both problems some information about environment structure and size is assumed and in both problems a group of robots must be assigned to a set of tasks. Currently the MRTA problem focuses on complex tasks definitions, including hierarchical tasks decomposition and complex dependencies[6], which wont be applicable in the considered case. However, the basic case of the MRTA, with constant set of tasks and one task kind only, represents very similar assumptions. If a space in a building is divided into well-defined fragments, then the operation of searching one fragment can be considered a task. Each robot in a group needs a schedule of fragments to visit – the aim is to minimize the overall time of visiting all the fragments.

In general the solutions to the MRTA problem can be divided into centralized and distributed. Distributed approaches (greedy methods, agent-based approaches[3]) are considered more robust, because no single-point-of-failure exists[16]. However typically they cannot provide optimal solutions. Centralized solutions adopt complex and time-consuming planning and optimization algorithms for calculating best possible solutions.

In[9] the authors propose a method for applying MRTA approach to the search operation in an open-space. The space is divided into regions based on the probability of finding a person. Then two planning algorithms are executed in order to prepare schedules for robots: simple load-balancing method and a variant of particle swarm optimization. The results show that both methods are very demanding – solutions for four-robots scenario have been calculated after more than 150 seconds.

The problem of MRTA can be also addressed by solutions to other well-known computer science problems, which are not specific for robotics. Existing solutions to the problems of combinatorial optimization, like Multiple Traveling Salesmen Problem (mTSP) and Vehicle Routing Problem (VRP) can be adopted to the specifics of the considered multi-robot rescue action. Currently the solutions to the mentioned problems are used in industrial applications[1], which guarantees algorithms maturity and high quality of available tools.

3. PREDICTIVE PLANNING ALGORITHM

The predictive planning approach operates on an abstract model of a building, which represents the structure of spaces and passages as a undirected graph. Each space is represented by a central-node connected to a set of passage-nodes, which are connected to passage-nodes of adjacent, accessible spaces. An example of a graph model for a simple environment is presented in figure 1.

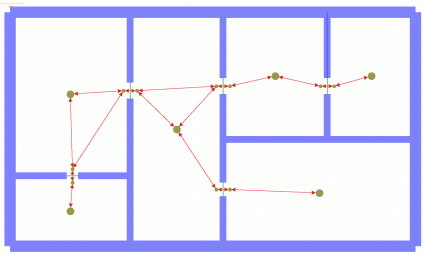

Fig. 1: Example of a spaces layout and the created graph model

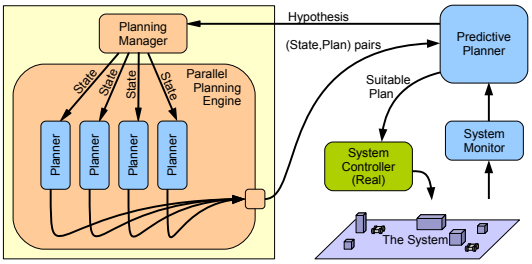

Fig. 2: Architecture of the predictive planning system

Each space is characterized with type and size. The size will be later used for estimating the time required to search the space.

The planner is supposed to calculate paths for all the robots. It should provide a list of *sub-plans*, one for each of the robots in the team. A *sub-plan* is a list of nodes to be visited, together with the information which spaces should be searched and which are only passed. The robots behave according to plan and report when a step of the plan is finished.

```
1  initialPlan = generatePlan(initialSystemState);
2  systemStateHipothesis = generateHipothesis(initialPlan);
3  while(!empty(systemStateHipothesis))
4  {
5    systemStateHipothesis = sortByExceptionTime(systemStateHipothesis);
6    systemStateHipothesis += generateHipothesis(generatePlan(systemStateHipothesis[0]));
7    systemStateHipothesis = removeFalseHipothesis(systemStateHipothesis, robotsReports);
8  }
```

Listing 1: The algorithm of the predictive planner.

It is assumed that the model of the environment can change while a plan is being executed. In the considered problem unpredictable situations can occur at any point in time: a wall can collapse, a robot may fail or delay its actions, a door can be locked or simply the model can be inaccurate.

If a robot detects inability of reaching a particular node, it reports the situation to the planner. The planner marks the edge as blocked and must provide a new plan for the new situation. The obvious solution is to re-plan all actions of all robots using the new environment model. This approach can generate significant downtime, especially when the robots operate in a large building. Therefore we propose a different, predictive planning method.

The predictive planner starts its operation by creating the first, initial plan using the building model provided – it assumes that all passages are open. The first plan is sent for execution by the robots. While the robots are moving, the planner starts to predict possible exceptional situations. Current version of the predictive planner assumes that only passages can be blocked – this is the most often situation caused by closed doors. The outline of the algorithm is presented in listing 1.

The *generateHipothesis* function creates a set of n possible system states, where n is the number of robots. Each of the generated states assumes that one of the robots will report inability of achieving next node in its plan, which results in the environment model with one edge removed. Other robots' state is calculated using the provided plan and the time of the exceptional situation detection.

The plans generated in line 6 of listing 1 are stored as the potential solutions for the future situations. The planner removes the states hypothesis and the plans if the state observed by working robots is different than the hypothesis.

In the created implementation the planning process executed in line 6 of listing 1 is performed in parallel for several state hypothesis. The parallelization is very simple because there are no dependencies between planning for different system states. This allows utilization of advanced computational hardware, including multi-core architectures, clusters or supercomputers for preparing plans for huge number of possible future situations. The architecture of the created planning system is presented in figure 2.

The Planning Manager and the Parallel Planning Engine are implemented in Scala, as a domain-independent, distributed application. The Planner component must be provided in order to configure the system for particular problems domain. Provided interfaces allow defining a list of tasks (Hipothesis) to be processed by the Planning Engine, which consist of a state definition and a deadline. The Planning Manager schedules tasks to available hardware, ensuring that the computations are finished before the deadline.

The predictive planning method uses the computational power in advance, calculating plans before it is needed. Therefore the planning algorithm implemented in the *generatePlan* function and executed by the Planner component can require some time to provide the results. The predictive planner can estimate the available time by calculating the time required by a robot to reach the hypothetically blocked edge. This value is passed as the deadline the Planning Manager and is used for limiting the time available for the planning algorithm. This mode of operation encourages use of optimization methods for finding plans for particular hypothesis. Optimization algorithms provide a feasible solution almost immediately after starting and continuously try to improve it.

In our experiments we decided to use one of available libraries for solving optimization problems, namely the OptaPlanner (http://www.optaplanner.org/) library. The library is written in Java and distributed as an open-source. It implements various optimization algorithms and provides large numbers of examples solving well-defined problems, like TSP or VRP. We modified the cost function of the VRP solution in order to adopt it to the multi-robot rescue

problem. Our implementation calculates the time needed by each of the robots to fulfill the assigned tasks as:

$$cost(subplan) = \sum_{task \in subplan} p * pathToTask(task) + s * spaceArea(task) \quad (1)$$

where *pathToTask* returns the shortest path lenght to the *task* in meters and *spaceArea* returns the area of the space to be searched in square meters. p and s are constant factors.

The cost of the solution is calculated as the maximum of all costs of the robots in the group. The whole task ends when the last robot finishes its plan by searching the last accessible room in the building.

4. PLANNING PERFORMANCE EVALUATION

In order to estimate possible outcomes of using the presented plan optimization algorithm with the predictive planing approach, a series of experiments have been carried out. The results have been compared with a simple, greedy solution. The greedy algorithm selected the nearest, not-yet–searched space each time a robot finished its task.

Both solutions do not require re-planning when an exceptional situation occurs in the system. The greedy algorithm uses the simple rule, while the predictive planning method has the alternate plans ready.

The tests involved two different buildings. The first is a one-storey building with 40 spaces, which contains a cycle of corridors; it is a part of authors' department building, generated using design plans. The second is a much bigger, four-storey layout with a single elevator in the middle and 112 spaces to search. For simplicity it is modeled in two dimensions with a central corridor connecting four blocks. The layout of spaces and the graph used in the planning process are shown in figures 3 and 4.

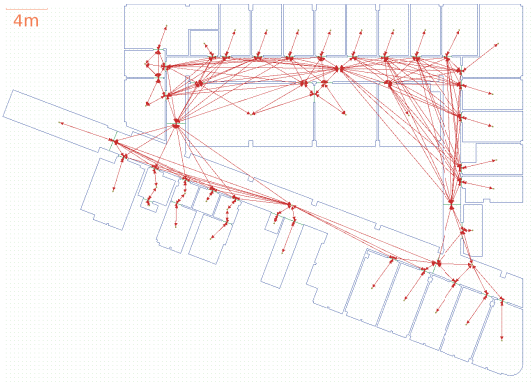

Fig. 3: The layout of spaces in the one-storey building

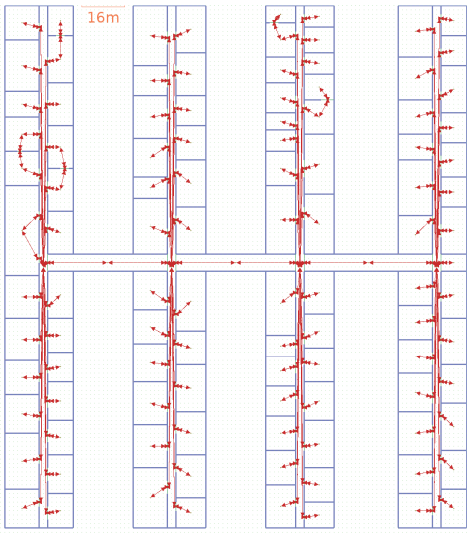

Fig. 4: The layout of spaces in the four-storey building

Each experiment involved calculating the total cost of searching each of the buildings with a different number of robots – groups of 1, 2, 4, 8, 16, 24 and 32 robots were considered. In addition different layouts of robots in the building have been tested – the robots were scattered in random locations or grouped in one, randomly selected room. Each variant have been repeated 10 times.

The optimization algorithm was executed on a modern PC with 3.5 GHz CPU. Only one core was used for creating a single plan in this synthetic benchmark. The planning time was fixed to 60 seconds. The cost function defined in equation 1 has been calculated with $p = 2$ and $s = 1$, which can be interpreted as the ability of searching the spaces with the same speed as in case of moving, covering two-meter-wide strips.

The comparison of the results is presented in figures 5 and 6. Average values and standard deviation is shown.

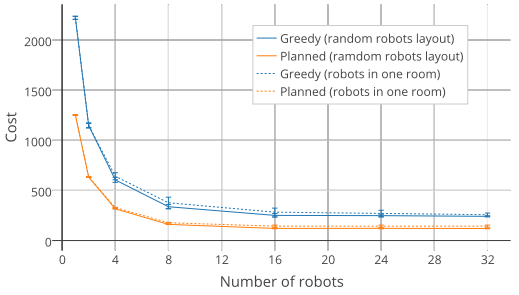

Fig. 5: Comparison of planning and greedy approach in the one-storey building

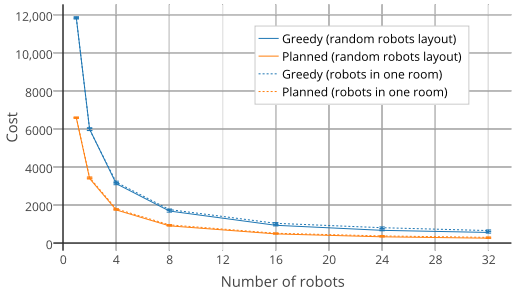

Fig. 6: Comparison of planning and greedy approach in the four-storey building

The characteristics of both algorithms is correct – the cost decreases with increasing number of robots. The results show, that the planning method gives results between 30% to 50% better than the greedy approach in both types of the buildings. Shortening a rescue action by nearly half is a very significant gain, which proves that use of the presented method is justified. Surprisingly, the initial layout of robots in the buildings has very little impact on the final cost of the plan. When robots are scattered, the results are better, but the difference is minor.

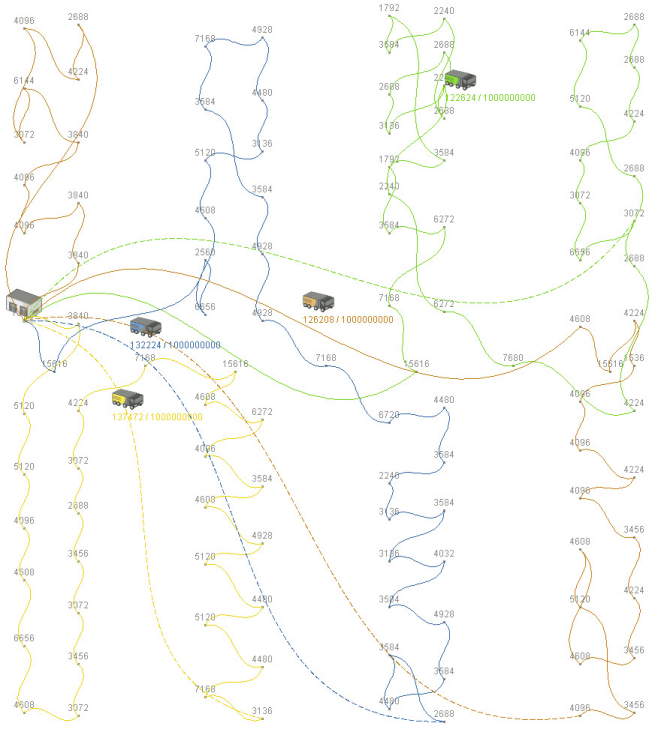

Fig. 7: Example of a plan for four robots in the four-storey building

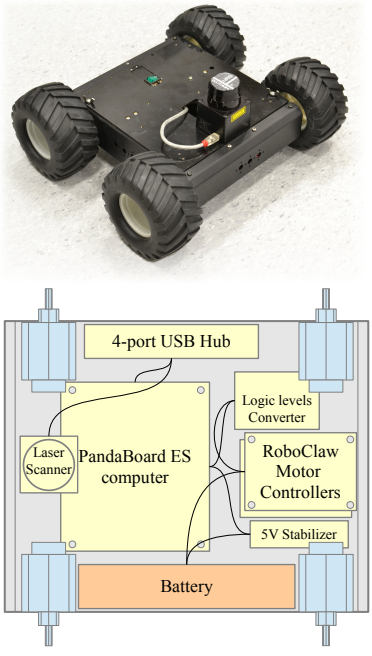

Fig. 8: Capo robot and its internal architecture

An example of a plan generated by the optimization algorithm for four robots in the four-storey building is presented in figure 7. The plan clearly divides the building between the four robots – each robot covers two out of eights wings. The cost of the plan was determined by the yellow robot, because its schedule was the longest. The paths of other robots are clearly not optimal, which is caused by the formulation of the cost function – total execution time is optimized, therefore robots which finish their part earlier are not necessarily using optimal paths.

5. EXPERIMENTS WITH ROBOTS

The created system has been also tested using real robots, which cooperatively searched a small environment. The tests aimed at verifying the correctness of the predictive planning approach in real-life situations, where a robot detects an obstacle blocking its passage.

The experiments were conducted using the CAPO robots [15]. The CAPO robot have been designed and build in the authors' department as a general-purpose mobile platform (figure 8).

The CAPO robot is controlled by a PandaBoard ES (http://pandaboard.org/) single-board microcomputer, which is equipped with 2-core Cortex-A9 CPU running at 1.2 GHz and 1GB of RAM. The computer runs under a Linux OS and allows execution of programs written in various high-level languages. It provides wireless communication and variety of peripheral interfaces for connecting sensors and effectors. The robot is driven by four 12V DC motors controlled by two RoboClaw (http://www.orionrobotics.com/) units, which use quadrature encoders for precise velocity control. Internal state is monitored by a 9DOF IMU sensor. Additionally each robot can be equipped with a laser scanner, cameras or other advanced sensors. The whole system is powered by high discharge 5Ah LiPo batteries.

Software platform is based on the Erlang technology, which has been designed for creating concurrent system for high availability. Its built-in mechanisms for recovering from exceptional situations and for handling concurrent I/O operations are very well suited for creating this type of systems. The drivers for particular devices are written mostly in C++. The client libraries are available in C++, Java, Pytohn, C# and Erlang. The experiments involved two CAPO robots searching the environment presented in figure 9.

Fig. 9: Initial situation and the created plan

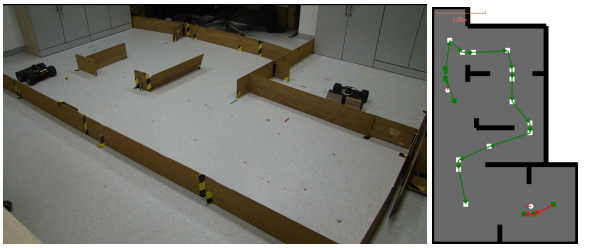

Fig. 10: Detected blockade and the alternative plan

The environment consisted of eight spaces and the robots were located in different starting positions. After reaching the second space the robot in the bottom-right detected the blockade (figure 10) and the system switched to the alternative plan, which had been pre-calculated in the meantime. The other robot fulfilled the new tasks successfully.

The experiments with the integrated robotics system show that it is possible to immediately adopt to the changes detected in the environment using the presented predictive planning approach.

6. CONCLUSIONS AND FURTHER WORK

The proposed solution for the problem of finding optimal plans for a group of robots searching a building seems to be a promising approach, which might be applicable in different mobile robotics problems. The domain often struggles with time consuming computations and with dynamically changing situation. The idea of predicting most probable future situations and utilizing available computational power in advance allows using time consuming algorithms in solving problems which require short response times.

The implemented prototype of the parallel planning system will be further improved and evaluated on high performance computing platforms. Hopefully, this domain-independent tool can be adopted for different problems with similar, real-time requirements.

Conducted experiments show, that the centralized planning for multi-robot rescue actions in buildings can be significantly improved by using optimization algorithms and the predictive planning approach. We believe that robot rescue systems, which operate in buildings, could soon find real-life applications. Therefore we are planning further research on management and control algorithms in this area.

Acknowledgement

The research leading to this results has received funding from the Polish National Science Centre under the grant no. 2012/05/B/ST6/03094 and from the AGH-UST statutory fund no. 11.11.230.124.

References

1. S. P. Anbuudayasankar, K. Ganesh, and Sanjay Mohapatra. *Models for Practical Routing Problems in Logistics*. Springer International Publishing, 2014.
2. Maxim A. Batalin and Gaurav S. Sukhatme. Spreading out: A local approach to multi-robot coverage. In Hajime Asama, Tamio Arai, Toshio Fukuda, and Tsutomu Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 373–382. Springer Japan, 2002.
3. Krzysztof Cetnarowicz. From algorithm to agent. In Gabrielle Allen, JarosÅaw Nabrzyski, Edward Seidel, GeertDick van Albada, Jack Dongarra, and PeterM.A. Sloot, editors, *Computational Science ICCS 2009*, volume 5545 of *Lecture Notes in Computer Science*, pages 825–834. Springer Berlin Heidelberg, 2009.
4. Asha Gupta, Nidhee Panchal, Dhruti Desai, and Divya Dangi. Live human detection robot. *International Journal for Innovative Research in Science & Technology*, 1(6):293–297, 2014.
5. H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 6, pages 739–743 vol.6, 1999.
6. G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.
7. Yugang Liu and Goldie Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, 2013.
8. H. Mano, K. Kon, N. Sato, M. Ito, H. Mizumoto, K. Goto, R. Chatterjee, and F. Matsuno. Treaded control system for rescue robots in indoor environment. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1836–1843, Feb 2009.
9. R. Meuth, E.W. Saad, D.C. Wunsch, and J. Vian. Adaptive task allocation for search area coverage. In *Technologies for Practical Robot Applications, 2009. TePRA 2009. IEEE International Conference on*, pages 67–74, Nov 2009.
10. Mark J. Micire. Evolution and field performance of a rescue robot. *Journal of Field Robotics*, 25(1-2):17–30, 2008.
11. R.R. Murphy. Human-robot interaction in rescue robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):138–153, May 2004.
12. R.R. Murphy. *Disaster Robotics*. The MIT Press, 2014.
13. Naoji Shiroma, Yu-Huan Chiu, Noritaka Sato, and Fumitoshi Matsuno. Cooperative task excecution of a search and rescue mission by a multi-robot team. *Advanced Robotics*, 19(3):311–329, 2005.
14. H. Sugiyama, T. Tsujioka, and M. Murata. Integrated operations of multi-robot rescue system with ad hoc networking. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 535–539, May 2009.
15. S. Szominski, K. Gadek, M. Konarski, B. Blaszczyk, P. Anielski, and W. Turek. Development of a cyber-physical system for mobile robot control using erlang. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 1441–1448, Sept 2013.
16. Malgorzata Zabinska, Tomasz Sosnicki, Wojciech Turek, and Krzysztof Cetnarowicz. Robot task allocation using signal propagation model. *Procedia Computer Science*, 18:1505–1514, 2013.