HELPING INSTEAD OF REPLACING
Towards A Shared Task Allocation Architecture

Foad Ghaderi and Majid Nili Ahmadabadi
Control and Intelligent Processing Center of Excellence, Mobile Robot Lab
Dept. of Elect. and Comp. Eng., Faculty of Eng., University of Tehran

Keywords: Cooperative robotics, shared task allocation, fault tolerance.

Abstract: Some failures cause the robots to loss parts of their capabilities, so that they cannot perform their assigned tasks. Considering requirements of typical robotic teams during different missions, a distributed behavior based control architecture is introduced in this paper. This architecture is based on an enhanced version of ALLIANCE, and provides the robots the ability of performing shared tasks based on help requests. The architecture contains a mechanism for adaptive action selection and a communication protocol for information and task sharing which are required for coordination of team members. The proposed architecture is used in a box pushing mission where heterogeneous robots push several boxes with different masses.

1 INTRODUCTION

Sensitivity of distributed robotic systems to changes in working environments and common failures in their mechanical and electrical components is a barrier to their wide physical implementation (Ahmadabadi, 2001). So a dynamic task allocation mechanism that supports fault tolerance behavior is a mandatory requirement for cooperative robotics solutions (Ghaderi, 2002).

Mataric et al. showed empirically that there is no optimal task allocation strategy for all domains, and even it is difficult to identify the optimal task allocation strategy for a particular task (Mataric, 2003). In their framework (Gerkey, 2003) only one robot could be assigned to a task, and no redundancies were allowed.

In (Ahmadabadi, 2004) two distributed and cooperative methods for load reallocation among some object lifting robots without requiring them to change their grasp positions are introduced.

Vig et al. provided RACHNA (Vig, 2005) which is a market-based architecture for allocating multi-robot tasks based on individual robot capabilities. In this architecture, if a robot can not support the team, it is more likely that another robot with similar capabilities replaces this robot.

Some advanced behavior based control architectures are introduced that allow adaptive task allocation. Parker's architecture (ALLIANCE) is based on Subsumption architecture (Brooks, 1986). It supports adaptive action selection for faulty robot replacement (Parker, 1998), (Parker, 1994).

Kasahara has considered reorganization in an organizational learning model (Kasahara, 1998). Whenever a fault occurs, agents try to compose new organizations using learning techniques. This process can lead the team to its goal.

In this paper, a new task allocation method is introduced which is based on ALLIANCE and supports help in a team of cooperative robots. In this method partially faulty robots do not leave the team, but the team tries to redistribute task among other members in order to use the whole capabilities of the robots. The suggested architecture supports adaptive action selection with help request processing and allows the group to perform its mission coordinated.

In the next section importance of help in a cooperative team of robots is discussed. The help supporting method is introduced in section 3. Box pushing problem is reported in section 4. Conclusions are presented in section 5.

2 HELP IMPORTANCE IN COOPERATIVE MISSIONS

Most of the real world mobile robots applications are performed in dynamic environments. Missions of rescue robots, cleaning robots, robots used for defense purposes and every other mission that robots are used instead of human in order to decrease dangers are examples of these applications. In these applications a lot of changes may occur in environment by time, and sometimes these changes cause a team to fail in some (or all) of its tasks. If the task selection mechanism is static and without flexibility, then there is no way to complete the tasks, except waiting for some robots to complete their tasks and replacing them with those robots that are not able to continue. Here we refer to an action selection mechanism as static when robots do not change their tasks even if there are some tasks in the team with higher priorities.

In this case if there is redundancy in the team, higher priority tasks are assigned to idle robots, otherwise these tasks would be assigned after a time interval that is unknown. Obviously such a task assignment may cause a disaster for the group or environment if the critical tasks are not assigned and performed in an acceptable time. We define that a task in a cooperative mission is unassigned, if no robots have selected it ever, or if the selecting robot(s) is (are) not able to perform the task.

Disability of a robot in performing a task may have two reasons. First, the robot is faulty and hence it can not complete the task, and second the robot is not faulty but its capabilities are not enough to complete the task.

Sometimes replacing the disabled robot with a new one provides a new chance to the team to achieve its goal. If there is redundancy in quantity of the robots, problem is solved easily, and otherwise one of the robots must ignore its task and perform the uncompleted task. But there are some situations that this method does not acquire mission's goals.

We divide these missions into two categories:

❒ **None of the team members can finish the uncompleted task alone.** In the other words, performing the task requires efforts of more than one of the existing robots and the task is not divisible to simpler subtasks. This kind of tasks is called "shared task" which requires closed coordination and real time action of a team of robots. Transferring of an injured person in a rescue mission is a good example of shared tasks. In this mission, none of the robots can complete the task alone and since some member's actions may disturb efforts of the others, a close coordination between robots is mandatory.

❒ **Performing the task depends on some robot's capabilities.** These capabilities may include some special mechanical mechanisms, processing power, knowledge and etc. Therefore, the replacement would be possible if there is another robot with at least the same capabilities.

It is clear that the traditional way of replacing robots will not have considerable effect in performance of the team. In this case, the best way to achieve the group's goal is helping the weak robots.

3 THE HELP SUPPORTING METHOD

There are two important issues to be considered in any architecture that supports help for faulty agents. Let's review a scenario first. Assume that some robots are cooperating to perform a mission. During their action, one of them senses that it is not possible to complete its task lonely and so broadcasts a help request. Receiving this message, other robots must decide about helping disabled robot. After making decision, the helping robots must cooperate to complete the disabled robot's task. During lots of cooperative help tasks a closed coordination between robots is required. So the help supporting architecture must include appropriate mechanisms to do action selection and closed coordination for a cooperative task.

3.1 Action Selection

The way in which robots process the help request and make decision has great effects on functionality and performance of the team. In addition, there are many factors to be considered when processing a help request. Some of them are listed below:

- Distance to the disabled robot (in general the cost to reach to the position in which the helper robot can help the disabled robot),
- Cost of performing the task,
- Having useful mechanical capabilities, knowledge and experience,
- Priority of robot's current task compared with the shared task,
- Criticality of the disabled robot's task,
- Progress of the task of the team members (including the faulty robot).

These are general parameters that can affect performance of a help process. Besides these, other parameters might be chosen according to specific nature of the mission.

We used ALLIANCE architecture (Parker, 1998) as a base for adaptive action selection. This behavior based control architecture has adaptive action selection capabilities that can be used in different missions, but this architecture does not support help. So we enhanced ALLIANCE to address the mentioned requirements.

Considering motivation function and it’s parameters in ALLIANCE architecture, it’s apparent that if a robot selects a task, (except in some special conditions), it will not ignore it till the task is finished. So while a robot is performing a task and receives a help request it is not possible to change its behavior and select a new task. In this case the help request is not answered if there is no idle robot in the team.

We supposed that each task is allowed to be shared just between two robots. In order to achieve adaptive action selection, some changes were applied to ALLIANCE. In fact, we designed a two dimensional motivation vector for each robot. Two axes of this vector represent team members and tasks of the team. So m_{ijk} in robot i is motivation function corresponding to behavior j, while robot k broadcasts a help request. The new motivation function for the shared task j in robot i is defined by the following equation:

$$
\begin{aligned}
m_{ijk}(0) &= 0 \\
m_{ijk}(t) &= [m_{ijk}(t-1) + \mathrm{impatience}_{ijk}(t)] \\
&\quad \times \mathrm{sensory\_feedback}_{ijk}(t) \\
&\quad \times \mathrm{activity\_suppression}_{ij}(t) \\
&\quad \times \mathrm{impatience\_reset}_{ijk}(t) \\
&\quad \times (1 + \mathrm{Help\_Request}_{ijk}(t)) \\
&\quad \times \mathrm{acquiescence}_{ij}(t)
\end{aligned}
\qquad (1)
$$

The parameters are defined below:

Impatience$_{ijk}$(t): This parameter shows robot's impatience for interfering other robot's tasks. If task j is not selected by any other robot, *Impatience* will have a large value; otherwise its quantity depends on the type of task, environmental conditions, and progress of the task. In shared task, some more factors such as the criticality of the task, the efficiency of the robot in performing the task and type of the help request are important.

Sensory_feedback$_{ijk}$(t): This parameter indicates whether behavior j is required at the time to reach the goals of the team or not. Often, physical robots sensors produce this information, but in practical robotics applications it's possible to use virtual sensors such as memory (as a flag), and communication lines to indicate this parameter. If robot k needs some help to complete task j and robot i can cooperate in this task, *sensory_feedback* is one, otherwise it is equal to zero.

Activity_Suppression$_{ij}$(t): When a behavior is active in a robot, this parameter suppresses behaviors with lower priorities to be activated. This suppression is not applied to higher priority tasks and so permits the robot to change its active behavior in order to increase team's efficiency or help other robots in critical tasks.

If priority of task j is higher than active behavior in robot i, this parameter is equal to 1, otherwise it's 0. In Parker's architecture (Parker, 1994), (Parker, 1998), the criticality of the tasks is not taken into account in evaluating this parameter. So if any behavior is active, others would be suppressed, regardless of the criticality of the others.

Impatience_Reset$_{ijk}$(t): The default value of this parameter is 1. When a robot selects behavior j to help robot k, it broadcasts its decision. As a result, *Impatience_Reset* in the other robots becomes zero for a defined period of time, and then increases to 1 again, and the robots motivation to participate in task j resets consequently.

Help_request$_{ijk}$(t): Default value of this parameter is zero and takes a positive value for a predefined interval, whenever robot i receives a help request from robot k requesting to cooperate in task j. If in this period the value of *impatience_reset* is not zero, the motivation value will be multiplied by a value bigger than one. (Notice that faulty robot broadcasts help request periodically until its task is finished.)

Acquiesence$_{ij}$(t): The same as that in ALLIANCE architecture.

Initial values of the parameters and their changes deeply depend on the general requirements of the mission, robots type and different performance indexes used. Therefore, the initial values must be optimized for each task.

3.2 Robots Coordination in Help Process

After deciding to help, robots must coordinate to perform the shared task. The coordination strategy depends on the special characteristics of the task and there is no general method to do this. But, the coordination methods are generally based on the information about activities of the other robots. Then, we used the communication method used in (Asama, 1992) to obtain information about others tasks, activities, and their progress in order to coordinate the robots.

In Asama’s protocol, nine procedures are defined to communicate information (Asama, 1992). We selected seven of them to coordinate the team:

- Report for ready state,
- Declaration of task selection,
- Request for Cooperative task,
- Acceptance of Cooperative task,
- Report of completion of preparation for cooperation,
- Command for start of cooperative task,
- Report of task completion.

4 EVALUATING THE ARCHITECTURE

Task assignment and adaptive behavior of a cooperative team that uses help mechanism is evaluated in a box pushing problem. Box pushing is a common test bed in the field of cooperative robotics. It is assumed that some boxes are distributed in an environment and some robots must push them to the front wall.

There are two kinds of boxes. Some of them are light and can be moved by a single robot. Others are heavy such that one robot is not able to transfer them alone. Each robot selects a box to transfer while it has no information about weight of the box. Whenever the robot detects that the selected box is heavy and it's not possible to move it alone, it will broadcast a help request.

Robots have some inexact information about the position of the boxes, so they must search for them. At the beginning, the robots assume that all of the boxes are light. So after selecting a box, the robot goes towards it and tries to move it. If the box is heavy, the robot broadcasts a help request, and waits for other's responses.

Experiments show that the team can manage existing resources to complete the mission in cases that some robots are not able to perform their assigned task.

5 SUMMARY

In this paper, we have introduced a help supporting architecture that focuses on task allocation in cases that some of the team members are not able to complete their tasks. This architecture supports fault tolerance in cooperative missions that have various tasks with different criticalities. In this method, the team tries to redistribute the tasks among members by processing help requests from disabled robots in order to use all robots capabilities. The suggested architecture supports adaptive action selection and let's the group to perform its mission in cooperation. In our method the robots are committed unless some critical tasks are not assigned and they are individualistic unless some robots require help. The architecture is evaluated in a box pushing mission and results show acceptable performance.

ACKNOWLEDGEMENTS

This work is partially supported by Control and Intelligent Processing Center of Excellence, University of Tehran.

REFERENCES

Ahmadabadi, M. N. and Nakano E., "A Constrain and Move approach to distributed object manipulation", *IEEE Trans. on Robotics and Automation*, Vol. 17, No. 2, pp. 157-172 , April 2001.

Ghaderi, F., Ahmadabadi, M. N. "Distributed Cooperative Fault Tolerance In A Team Of Object Lifting Robots", In *IEEE Proc. of 1996 Int. Conf. on Intelligent Systems and Robots*, 2002, pp. 2721-2727.

Mataric, M. J., Sukhatme, G. S., Østergaard, E. H., "Multi-Robot Task Allocation in Uncertain Environments", *Autonomous Robots* 14, 255–263, 2003, Kluwer Academic Publishers.

Gerkey, B. P., Mataric, M. J., "A Framework for studying multi robot task allocation", In *Multi-Robot Systems: From Swarms to Intelligent Automata*, Volume II, Pages 15-26, the Netherlands, 2003, Kluwer Academic Publishers.

Ahmadabadi, M. N., Ghaderi, F. "Distributed cooperative load redistribution for fault tolerance in a team of four object-lifting robots", *Advanced Robotics*, Vol. 18, No. 1, pp. 61–81, 2004.

Vig, L., Adams, J. A., "A Framework for Multi-Robot Coalition Formation" In *Proc. of the 2nd Indian International Conference on Artificial Intelligence*, 2005, India.

Brooks, R. A., "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol.2, No.1, March 1986 pp. 14-23.

Parker, L. E., "ALLIANCE: An Architecture for Fault tolerant Multi-robot Cooperation", *IEEE Trans. robotics and automation* vol. 14, No. 2, pp. 220-240, April 1998.

Parker, L. E., "Heterogeneous Multi-Robot Cooperation", *Ph.D. Thesis*, Massachusetts Institute of Technology, Cambridge, MA, Feb.1994.

Kasahara, H., Takadama, K. , Nakasuka, S., Shimohara, K.,"Fault Tolerance in a Multiple Robots Organization Based on an Organizaional Learning Model" *The IEEE 1998 International Conference On Systems, Man and Cybernetics (SMC'98)*, pp. 2261-2266, 1998.

Asama H., Ozaki K., Matsumoto A., Ishida Y., and Endo I. , "Development of task assignment system using communication for multiple autonomous robots", *Journal of Robotics and Mechatronics*, pp. 122-127, 1992.