Caspian 2004 Rescue Simulation Team Description

Mohammad Nejad Sedaghati, Nina Gholami, Ehsan Rafiee, Omid Mehdi Zadeh, Leila Pakravan Nezhad, Sommayeh Zahedian, Hamid Hamraaz, Shams Feyzabaadi, Reza Seyed Khamooshi, Sina Iravanian, Bardia AghaBeigi, Mehrdad Senobari, Mohammad Jafar Abdi, Arash Alikhani, Mohammad Saeed Tabatabaee, Mohammad Reza Kangavari

mnsedaghat@yahoo.com

Intelligent Systems Lab, Computer Engineering Department
Iran University of Science and Technology, Tehran, Iran

http://caspian.iust.ac.ir

Abstract. The RoboCup Rescue simulation environment is a challenging multi-agent domain where task need to be done collaboratively by heterogeneous agents. This paper has provided an overview of *coordination strategy* in Caspian Rescue Simulation Team. Our coordination strategy is a combination of *centralized coordination* and *standardized coordination* to organize disaster rescue operation. We have used a task scheduling algorithm called "*Preemptive Priority Scheduling*" which has been successfully applied to police center agent. We have also take advantage of *partitioning* to determine a searching area for each agent in fire rescue team. This can be seen as a *social law* which helps us to achieve *standardized coordination*.

1. Introduction

The RoboCup Rescue simulation environment is a challenging multi agent domain where task need to be done collaboratively by heterogeneous agents [1]. There are three fundamental approaches to solve the coordination problem [2]: *Mutual adjustment* [3][4], *centralized coordination* and *standardization* [5][6]. *Mutual adjustment* means that each agent is trying to adapt its behavior to improve coordination [7]. *Centralized coordination* means the leader agent performs the task scheduling and allocation on the basis of its own information, and the knowledge about resource consumptions. Finally *standardization* means that there are some *social laws* enforcing the coordination among the agents [7]. We have used a combination of *centralized coordination* and *standardization* in each rescue team (a rescue team is composed of a center agent and the corresponding platoon agent.) That is each center agent organizes the arrived requests from the other center agents, and then performs the task scheduling process (*Centralized coordination)*. In addition, we take advantage of *partitioning* mechanism as a *social law* [5] which assigns a working area to each platoon agent (standardization.) Scheduling and planning is an important aspect of coordination [9]. To address this, we have utilized a task scheduling algorithm called "*Preemptive Priority Scheduling*" in the design of the police center agent. The idea of this scheduling algorithm comes from a CPU scheduling algorithm which is used in the design of Operating Systems. In this paper, first we describe the task scheduling algorithm, which is used to develop a *centralized coordination* in a police rescue team. Then we will focus on the coordination strategy of the fire rescue

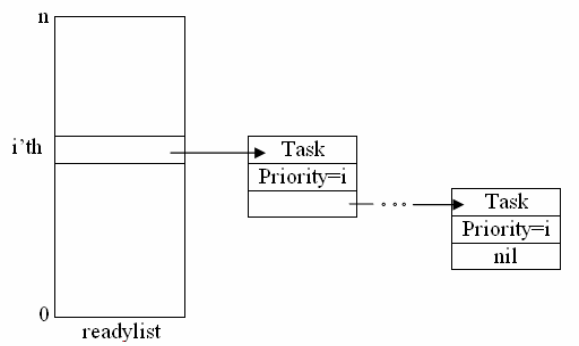

Figure 1. An overview of the Police Center Task Scheduling Algorithm

team, with the main emphasis on environment *partitioning* and the *task allocation mechanism*. Finally we study the results and the potential improvements to the current coordination strategy in the Conclusion and Future Works section.

2. Police Center Task Scheduling Algorithm

Scheduling has been defined as "the art of assigning resources to tasks in order to insure the termination of these tasks in a reasonable amount of time" [8]. In the police center task scheduling problem, police forces represent resources and the arrived requests from the other centers represent the tasks. We define a `task` as a structure which encapsulates the arrived request properties, including the `taskStatus`, `roadId`, `policeId`, `submittedTime`, and `priority`. `taskStatus` can be either `ST_READY` or `ST_RUNNING`. When a task is first submitted, the `taskStatus` is `ST_READY`, until it is assigned to a police force, then the `taskStatus` turns into `ST_RUNNING`. `roadId` is the id of the road which is mentioned in the arrived request. `policeId` is the id of the police force who is responsible for the request and will be assigned later. `submittedTime` is the time that the request arrives, which is used for monitoring and control purposes. In current implementation task priority is equal to the static priority of the requested road, which is calculated using the Floyed algorithm. This priority is calculated at the beginning of the simulation. Once the encapsulation process is finished, the `task` structure is inserted into a priority based list called `ReadyList`. i'th element of the `ReadyList` is a pointer to a `task` queue whose members are `task` structures with `priority` equal to i. Therefore, a `task` with `priority` equal to i is inserted at the end of the queue which the i'th element of the `ReadyList` is pointing to. (Figure 1)

After insertion of the `task` into the `ReadyList`, the `scheduler()` function is called. The pseudo-code of the scheduler algorithm is shown in listing 1.

```
void scheduler ( )
{
  FindHighestPriorityTask(highestPriorityReadyTask)
```

```
  AllocateIdleProcessor(lowestPriorityRunnig Task,
                                isIdleProcessorFound)

  if(! isIdleProcessorFound)
          preemption(highestPriorityReadyTask,
                          lowestPriorityRunningTask)
}
```

Listing1. The pseudo-code of the scheduler algorithm

`FindHighestPriorityTask()` searches the `ReadyList`'s `task` queue to find the `task` with the ready status which has the highest priority, and then returns the selected task. In the next step `allocateIdleProcessor()` is called. Considering that a police force agent in rescue simulation scenario is equivalent to a processor in Operating Systems concept, this function assigns an idle police force to the highest priority ready task. For this purpose, an array called `processors` is considered with the element that each encapsulates the corresponding police force state, including `policeId`, and `currentTask`. `policeId` is the id of the police force which represents this processor. `currentTask` is referring to the current assignment of the corresponding police force. A null reference means that the corresponding police force is not currently assigned to a specific mission. Having police forces current status in the `processors` array, the `AllocateIdleProcessor()` searches the `processors` array for an idle police force. If there is an idle police force, it will be assigned to the highest priority ready task. On the other hand, if no idle police force is found, this means all processor have a running task on them, task with the lowest priority which is assigned to a police force is returned. Finally the `preemption()` function decides whether the lowest priority running task goes on or it should be preempted by the highest priority ready task. The `scheduler()` function is also called in another situation when a police force reports his mission is finished. This is when the corresponding task will be removed from the ready list and the `scheduler()` is called to assign the set free resource to another task.

3. Coordination Strategy in Fire Rescue Team

Coordination Strategy in fire rescue team is a combination of centralized coordination and standardization. At the beginning of the simulation, the environment is divided into several partitions so that each partition is assigned to two fire brigades. The partitioning mechanism is angle-based and relative to the center of the simulation map. From two fire brigades which are assigned to same partition, one is located at farthest point in the partition relative to center of map, and the other is located at the nearest point (relative to center). Partitioning and the initial location is considered as two social rules which help to achieve standardized coordination. In the next step, each fire brigade starts searching his assigned area to find fire sources. This is done during a certain period of time which is called "initial searching time", and the agents reports the observed fire sources to the fire station and identify the fire sites.

Each fire site is defined as the group of fiery buildings which are close to each other. When "initial searching time" is over then fire station comes into play to organize the "fire rescue team" using a "centralized task allocation" approach. That is, first most important fire sites are selected by fire station based on the fire sites which are collected by fire brigades during the "initial search time". The number of fire sites which are processed concurrently depends on the number of fire brigades. We have found out from several experiments that a group of fire brigades with less than four members could hardly handle a fire site. After picking the most important fire sites, each one is announced as a mission to a suitable group of fire brigades. The considered parameters in forming the group are distance of fire brigades to fire sites, the fire brigade status including the hp and water quantity, and the state of fire site like average fieriness.

4. Conclusion and Future Work

Cooperative multi-agent system in which agents must interact together to achieve the goal is a very active field of research [10]. This paper has provided an overview of cooperation strategy in Caspian Rescue Simulation Team. Our coordination strategy is a combination of centralized coordination and standardized coordination to organize disaster rescue operation. We have used a task scheduling algorithm called "Preemptive Priority Scheduling" which is inspired from a CPU scheduling algorithm in Operating System concepts. This algorithm has been successfully applied to police center agent task scheduling. There are some potential improvements to this approach including developing a mechanism to assign a reasonable priority to the in-coming tasks. Parameters like the sender agent type, number of requests for the same task, and how time critical the request is, should be taken into account. Current task scheduling approach suffers from a potential starvation for the task with low initial priority. That is, a low priority task in the ready list may wait forever as long as there are some tasks with higher priority. This can be solved by applying the aging mechanism. Aging mechanism means that the initial priority of a ready task in the ready list is increased as the waiting time extends. Actually this is moving from static priority assignment to a dynamic priority assignment in ready list structure. We have also take advantage of partitioning to set a searching area for each agent in fire rescue team. This can be seen as a social law which helps us to achieve "standardized coordination".

References

1. Kitano, H.,Tadokoro, S. et al. RoboCup-Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research, Proc. of IEEE SMC. 1999.
2. Mintzberg, H.: *The Structuring of Organizations*. Englewoods Cliffs (1979)
3. Sugawara, T., and Lesser, V. R.: Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments. *Machine Learning*, Vol. 33 (1998) 129-153
4. Tambe, M.: Towards flexible teamwork. *Journal of Artificial Intelligence Research*, Vol. 7 (1997) 83–124
5. Shoham, Y., and Tennenholtz, M.: On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the National Conference on Artificial Intelligence*, San Jose, CA. (1992) 276–281

6. Stone, P., and Veloso, M.: Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence* (AIJ), Vol. 100, number 2, June (1999)
7. Paquet, S., Bernier, N., Chaib-draa, B.,: Comparison of Different Coordination Strategies for the RoboCup Rescue Simulation. In Proceeding of RoboCup Symposium 2003
8. Dempster, M., Lenstra, S,. and Kan R,. Deterministic and stochastic scheduling: introduction. *Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems*, D. Reidel Publishing Company: 3-14, 1981.
9. Coates, G., Whitfield, R., Duffy, D., and Hills, B.,: Coordination Approaches and Systems – Part II: An Operational Perspective. Research in Engineering Design (2000) 12:73-89. 2000 Springer-Verlag London Limited.
10. Wooldridge, M.: *An Introduction to MultiAgent Systems*. Wiley (2002)