Distributed Task Allocation In Dynamic Multi-Agent System

Thesis Report submitted in partial fulfillment of the
requirement for the degree of
Master of Technology

In

Computer Science & Engineering

Under the Supervision of

Deepak Dahiya

By

Vaishnavi Singhal (132211)

Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that thesis report entitled "**Distributed Task Allocation in Dynamic Multi-Agent System**", submitted by **Vaishnavi Singhal** in partial fulfillment for the award of degree of Master of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Supervisor's Name………………

Designation……………………

ACKNOWLEDGEMENT

This is my long but the task of my thesis work both theoretically and practically may not have been completed without the help, guidance and mental support of the following persons.

Firstly I would like to thank my guide **Deepak Dahiya,** Professor, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, who provided me the related material and idea for the project proposal. He indeed guided me to do the task for my thesis in such a way that it seems to be research work, encouraged me a lot for doing my thesis in very smooth manner. Even if I made the mistake sometime he always tried to correct those mistakes and endeavor always to take me in the right direction.

Secondly, I would like to thank **my parents** who have always been with me for inspiring me that I can do the good thesis task with the hard work. Their instigation always helped me to grow my mind focused towards the hard work for implementation of thesis with having the research work in the mind.

Thirdly, I would like to thank **GOD** for keeping me enthusiastic, energetic and healthy every time due to which I could complete my thesis work successfully.

Once again thanks a ton to all mentioned people in my life.

Date: ……………………… Signature………………………

Name …………………………

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Distributed task allocation has been the hot research topic from the last few years. It is the heart of multi-agent systems. In multi-agent system, the agents coordinate and cooperate with other agents to accomplish the complex task which cannot be completed by an individual agent. Here, a distributed task allocation approach is proposed in constrained cooperative multi-agent environment (dynamic, real-time and uncertain). Agent allocates the task to multiple agents by considering the spatial, temporal and communicational constraints of the environment. The proposed approach considers the negotiation-based task allocation approach where the main agent announces the task and then other agents sends their respective bids for the received task. Best bid is chosen from all the received bids and then task is allocated to winning agent or group of agents. The main objective is to minimize the waiting time for a task to be accomplished and the number of messages transferred among agents for task allocation process. Furthermore, due to uncertainty of dynamic environment where the environment gets evolved at any point of time and plan gets failed, a re-planning algorithm is proposed which enables the agents to re-coordinate their plans when environment problem avoid it to fulfill them. The proposed approach is applied to the fire-fighting multi-agent environment where the allocation of fire-brigade agents is done to extinguish the fire in an efficient and effective manner. The approach is simulated in a multi-agent framework JADE and the result shows that the proposed approach requires less number of messages and less waiting time for the successful task allocation.

CHAPTER 1

INTRODUCTION

This chapter introduces the work presented in this thesis. Particularly, the motivation, objectives of the research work is described briefly. The chapter concludes with an overview of structure and content of the thesis.

1.1 Overview

The multi-agent systems are composed of intelligent entities called agents. Multi-agent system enables us to study the dynamic environments those are very similar to the real-life systems. There are various applications where multi-agent system plays an important role like transportation, disaster scenarios, coordinated defense systems, networking and mobile applications in order to achieve high scalability, dynamic load balancing and self-healing networks. The interaction of the agents can be selfish or cooperative. That is, the agents can pursue their own interest or can share the common goals of the system. It is very difficult for a single agent to achieve the system's goals individually so it grouped to form a multi-agent system.

Distributed task allocation and coordination have been the hot research topic in multi-agent system. In order to accomplish any task, there is requirement of a proper task allocation scheme. When the task is so complicated that it is very difficult to accomplish it by single agent then group of multiple agents have to be formed. To achieve the system goals, these agents must have to coordinate with each other. The coordination among the agent must have to be optimized so as to get an optimized task allocation strategy. The distributed task allocation process may be more difficult if the environment is dynamic, uncertain and real-time. Dynamic and uncertain means that the environment may evolves at any point of time. The agents cannot know with certainty that how the environment will evolve and what is the impact of its action on the environment. And real-time systems are those which involve some sort of constraints like spatial, temporal and communicational. To design any task allocation process for real-time systems, these constraints must have to be kept in mind.

In this sense, it is very important to design the task allocation process includes spatial, temporal and communicational constraints. Spatial constraint is related to the location of either agent or the task; temporal constraint is concerned with the deadline to start any task. To accomplish any complex task, the agents need to communicate with other agents. In multi-agent system, this communication is generally done via message passing. Thus communication constraint is concerned with the number of messages transferred for the allocation of the task among group of agents. This thesis addresses a task allocation approach for dynamic environment with temporal, spatial and communicational constraints. This also focuses on the coordination mechanism i.e. how the agents coordinate with each other so as to accomplish complex task. Here market-based auction strategy is used for the coordination problem. The communicational constraint is applied during coordination so as to achieve optimized task allocation process.

Briefly this thesis proposes:

- A task allocation algorithm which coordinates the agent using auction-based negotiation. Here the task is delivered to that agent who can accomplish it in lesser time. The agents are considered to be heterogeneous in the sense that, their implementation and functionalities are same but capabilities are different. Thus if the chosen agent cannot fulfill the task's requirement then negotiation is done. The participated agents submit their bids and the group of agent having best bid will be chosen for the task accomplishment.
- A trust model is also used for the task allocation purpose. On the basis of the trust factor, the most trust worthy agent is chosen for the task accomplishment first.
- Re-planning of the task allocation is done when agents face problems in accomplishing the assigned task due to uncertainty of environment.

1.2 Motivation

Disaster management has become an important and challenging issue in last few years. Disaster management coordinates a large number of rescuers to rescue the people or infrastructure so as to save them. A disaster environment is a dynamic environment where the environment conditions are unpredictable. The Disaster management includes various rescue activities like extinguishing the fire; rescue the patients to hospitals, cleaning beaches etc. The disaster management is responsible for allocating the rescue teams to accomplish these tasks for optimal recovery from the disaster.

Let us consider the case of fire-fighting. When any fire incident occurs in the society, the time to allocate the appropriate fire brigade to extinguish the fire is very crucial. If the allocation process is not optimal then it may results in a severe damage to the infrastructure as well as people. Multi-agent systems enable us to study such type of dynamic real-time systems.

Thus we devise a task allocation approach for fire-fighting multi-agent environment. Various intelligent fire-brigade agents cooperatively perform the rescue operation to extinguish the fire. The simulation is done on the Java Agent Development Framework (JADE), which facilitates the development of multi-agent systems. The main challenge involved in fire-extinguishing scenario is the time to response fire event by allocating the fire-brigades in less time.

This thesis discusses the approach of allocating the fire-brigade agents to the location where the fire event has been occurred with minimum waiting time. The fire-brigade agents are of heterogeneous in nature as they possess different capacities of the water tanks. To fulfill the requirement of any fire event, a group of fire-brigade agents have to be communicated with each other. The communication is done by message passing. Thus the task allocation approach which is discussed in this thesis considers the communication constraint along with spatial and temporal constraints. The proposed approach allocates the fire-brigade agents at the location of fire event with minimum waiting time and lesser number of message transfers.

1.3 Objectives

The main objective is to develop the algorithm that efficiently allocates the task in a constrained-cooperative multi-agent environment. Particularly, the focus is on the constrained environment of extinguishing the fire. The main aim of the proposed work is to allocate the appropriate fire-brigade for the fire-event as soon as possible and with less communication cost. The proposed algorithm must take the advantage of distributed approach to allocate the task in a co-operative fashion and to be real-time so as to allow the agents to face the changes in the scenario. To accomplish this objective, this thesis deals with the following specific objectives:

1. To apply the negotiation based technique used by the contract-net protocol with the inclusion of communicational constraint in order to improve the task allocation among the fire-brigade agents.
2. To apply the trust model during task allocation process, so as to allocate the task to most trust-worthy fire-brigade agent in order to minimize the failure of the fire-event.
3. To design a re-planning algorithm in order to allow agents to face the changes occurred in the environment due to the dynamic nature of environment.

1.4 Thesis Outline

This thesis is organized in the next 7 chapters:

Chapter 1 gave the introduction, motivation and the objectives of the research work.

Chapter 2 gives the overview of the background information related to the thesis work i.e. introduction of software agents, multi-agent system and agent communication and interaction protocols.

Chapter 3 presents the most relevant work related to the distributed task allocation in multi-agent system. It also presents the critical review on the various existing approaches of task allocation in multi-agent system.

Chapter 4 gives the brief overview of JADE multi-agent framework.

Chapter 5 describes the formalization of the task allocation approach and the proposed approach is also explained in this chapter.

Chapter 6 shows the experimental setup for this thesis work.

Chapter 7 shows simulation results and observations of the proposed approach.

Chapter 8 gives the author contribution to the society.

Chapter 9 provides the conclusion and outlines the most promising directions for the future work.

CHAPTER 2

BACKGROUND

This chapter introduces the basic concepts of software agents, multi-agent system, agent coordination and communication and Multi-agent interaction protocols.

2.1 Distributed Artificial Intelligence

Distributed Artificial Intelligence (DAI) is the study, application and construction of multi-agent systems. Multi-agent system is a system in which multiple intelligent agents interact with each other in order to achieve some set of goals. DAI addresses the research of developing the automated intelligent systems with an effective interaction.

DAI field is broadly divided into two research areas: Distributed Problem Solving (DPS) and Multi-agent system. DPS emphasizes on the problem and how to solve this problem by multiple intelligent entities, working together in an efficient manner, i.e. programmed computers. In multi-agent systems, the components are the intelligent agents which have some autonomous properties. These agents cooperate with each other in order to achieve the system goals. Contrariwise to the study on DPS, the multi-agent systems possess the property of reasoning out the coordination problem among the agents themselves.

There are various applications domain for multi-agent systems for example: manufacturing system, industrial procurement, crisis management, and network routing and airport traffic management. All of these applications require some autonomous entities i.e. agents that efficiently and effectively coordinate with each other to meet their design objectives in uncertain and dynamic environments [31].

2.1.1 What is an Intelligent Agent

Now-a-days, the intelligent software agents are popular research objects in the field of psychology, sociology and computer science. Software agents have their roots in work conducted in the fields of software engineering, computer-human interaction and the artificial intelligence.

Selker (1994) defines agents as "computer programs that simulate human relationship by doing something that another person could do for you". Smith defines it as "persistent software entity dedicated to a specific purpose".

According to [Wooldridge and Jennings, 1995] [1], an agent is a computer system which is situated in some environment and capable to perform the actions autonomously in order to meet its design objectives.

Janca (1995) introduces agents as "a software entity to which tasks can be delegated".

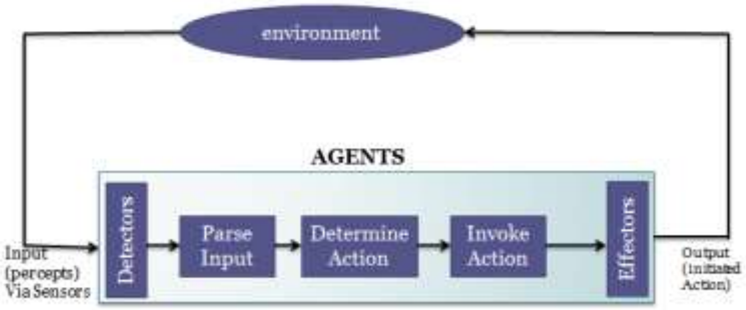


Figure 1- working of software agents

In above figure, agents perceives some input from environment and then parse the input using its environment knowledge (beliefs) and select a plan from plan library which is acquired to achieve the desired goal. The action is then invoked and performed back to the environment.

The software agents possess basic four properties i.e. autonomy, proactive, social ability and reactive [2]:

•**Autonomy**, agents operate without the direct intervention of human or others, and make their own decisions

•**Proactive**, agent exhibit goal-directed behavior by taking the initiative.

•**Social ability**, agents interact with each other via some kind of agent communication languages.

•**Reactive**, agent respond immediately to change in the environment.

Generally intelligent agents are dependent on each other. They interact with other agents in order to meet their design objectives. Thus agent forms group to achieve the system goals. This grouping constitutes the multi-agent system. Agents in cooperative multi-agent system coordinate their actions with other agents to fulfill its goals. For cooperative multi-agent systems, task allocation is an important requirement. It enables agents to know their individual goal so as to improve the overall system goals. The difference between the traditional system and the multi-agent system is shown in the Table 1.

Table 1- Traditional System vs. Multi-agent system

Traditional System	**Multi-agent System**
Sequential execution of operations	Parallel execution of the operations
Hierarchies of large programs	Large networked of small agents
Centralized decision	Distributed decision
Data driven	Knowledge Driven
Predictability	Self-organization
Instruction from top to bottom	Negotiations
Striving to reduce the complexity	Striving to thrive with the complexity

2.1.2 Existing architectures of intelligent agent

According to [3], there are four classes of agents:

i. Logic based agent architecture

In this architecture, the decision making is done through logical deduction

ii. Reactive agent architecture

In this architecture the direct mapping from situation to action is done for decision making

iii. Belief-Desire-Intention agent architecture

The agent is represented using belief-desire-intention model. Belief stands for knowledge about the world which can be incomplete knowledge, Desire stands for event or the task which agents want to perform and Intention stands for the plan which agent follows to accomplish its desire.

iv. Layered architecture

The decision making is done at different level of abstraction via various software levels.

2.1.3 Agent execution cycle

According to [4], actions, percepts, events, goals, plans and beliefs are the key components used to implement decision making of the agent. Agent's execution follows sense-think-act cycle. That is, when any event occurs in the environment, the agents' first sense that event then it thinks about the action which has to be performed and then perform the action.

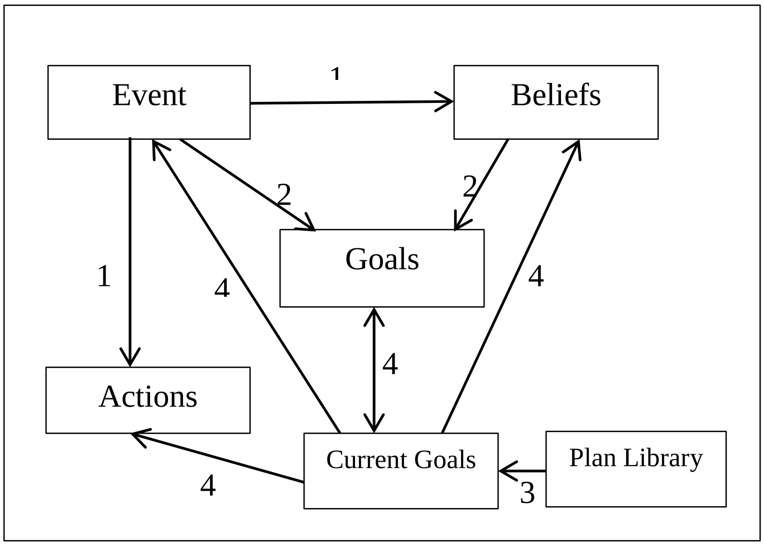


Figure 2- Agent Execution Cycle [4]

The agent execution cycle includes following steps:

1. To update the beliefs, events are processed and immediate actions are then generated.
2. Updating the goals by generating the new goals and achieved and impossible goals are dropped.
3. Available goals are achieved according to the plan from plan library
4. The plan is then executed.

2.1.4 Agent environments

The various type of environment from which agent can receive percept and performs the corresponding action are [4]:

- **Accessibility,** whether the complete information about the environment can be gathered or not?
- **Determinism,** whether effect of the action on the environment is definite?
- **Dynamic,** whether the entities can influence the environment at any moment of time?
- **Discreteness,** whether the entities in the environment are finite?

- **Episodicity,** whether the action of one agent influenced the other over some time instance?
- **Dimensionality,** whether the agents consider the dimensionality constraints of the environment?

2.2 Multi-Agent System

Multi-agent system is a system in which multiple agents interact with each other to achieve their goals. Multi-agent system is a very active field of research as it enables us to study the real-time applications in a more effective and efficient manner. This section introduces the multi-agent system, its characteristics and the application.

2.2.1 Introduction

Imagine there is an agent who involves in e-commerce i.e. tracking available goods on various e-shopping sites for sale and purchasing some items on the behalf of you. For successful operation, the agent will cater your knowledge related to your preference, your budget, and the environment where you want to use it and so on. For this the agent will have to exemplify your knowledge with other agents like store agent, transport agent and so on. Such agents collectively form the multi-agent system.

Multi-agent systems are composed of multiple software agents who interact with each another by exchanging messages through some computer network arrangement [2].

In order to successfully interact, these agents will thus require 3 Cs.

- **Coordinate**; agents achieve a common goal by coordinating each other.
- **Communicate**; agents pass messages for the interaction among them.
- **Cooperate**; by cooperating with each other, agents achieve the common goal.

Multi-agent System focuses on system of autonomous agents who are self-motivated and act in order to achieve their own personal task and increase their own personal gain [4].

In multi-agent system, the agents coordinate their knowledge and activities with each other to accomplish their desire and coordinate their knowledge. Thus the main research challenges in multi-agent system are problem decomposition, coordination and communication.

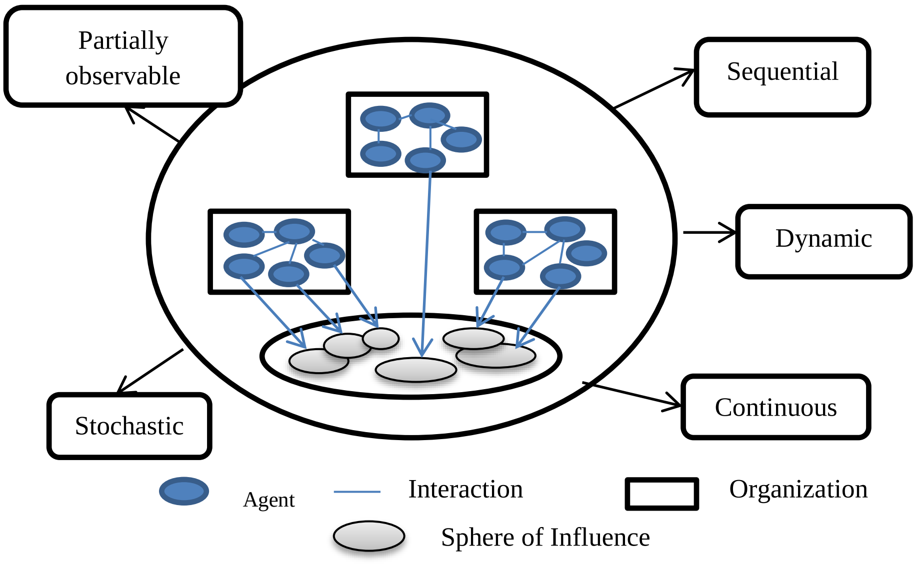


Figure 3- Environment of Multi-agent Systems [31]

2.2.2 Characteristics of Multi-agent Systems

The best way to depict the distributed computing systems is Multi-agent systems. There are several characteristics of multi-agent systems given by [3]:

- An infrastructure with communication and interaction protocol is provided by the Multiagent environment
- Multiagent environment doesn't require any centralized designer.
- Multi-agent systems are open and dynamic in nature.
- The agents those comprise the multi-agent system are autonomous and distributed in nature.

There are numerous concerns in the multi-agent execution environment that can be reckoned as the possible characteristics of multi-agent system.

Table 2- Characteristics of Multi-agent Systems [3]

Properties	**Values**
Design Autonomy	Platform / Interaction protocol
Communication infrastructure	• Shared Memory or Message-based • Connected or Connectionless • Point-to-point/ multicast/ broadcast • Push or pull

	• Synchronous or Asynchronous
Directory Services	White pages/ Yellow Pages
Message protocol	• KQML • HTTP / HTML • OLE/ CORBA/DSOM
Meditation services	Ontology based/ Transaction
Security services	Authentication/Time-stamp
Remittance services	Billing/ currency
Operation support	Archiving/ redundancy/ restoration/ accounting

2.2.3 Applications of MAS

There are various industrial and commercial applications for multi-agent systems. Such applications are:

- **E-Commerce**, where “buyer” and “seller” agents are used to purchase and sell the products on the behalf of users
- **Student-scheduling system**, here three agents namely student agent, lecture agent and scheduling agent communicate for schedule decision
- **Automatic- target recognition,** the agents sense the target and communicate with each other for the computation.
- **Traffic-monitoring**, agents are also used for traffic-monitoring. The traffic agents sense the traffic and communicate with driver agent.
- **Disaster-rescue operation,** various agents communicate and coordinate with other to perform the rescue operations.

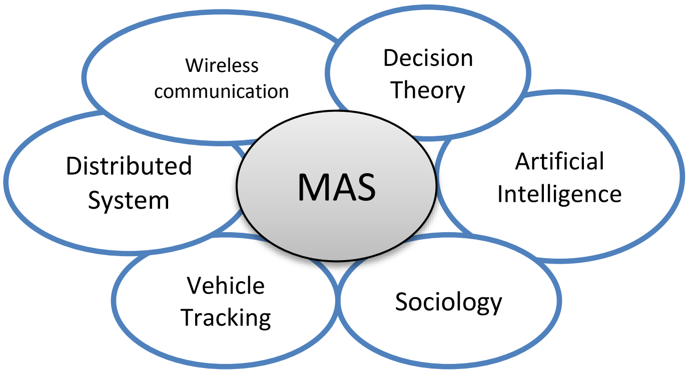


Figure 4- Various Domains of multi-agent system

2.3 Agent communication language

To represent the properties of communicating concurrent systems, much formalism have been developed in computer science. There are number of key issues that have tended to focus when dealing with systems that can interact with one another. Consider a scenario of agent-oriented programming. There are two agents ‘I’ and ‘j’, where ‘I’ has some capability to perform action ‘a’. But there is no concept for agent ‘j’ to invoke the method of i, because of its autonomous property. It can’t be taken for granted that agent ‘i’ will perform the action ‘a’ because agent ‘j’ want it to get performed [2].

Generally an agent can’t force the other agent to perform some action. This doesn’t mean that they can’t communicate however they can perform communicative action i.e. an attempt to influence other agents [2]. Agents communicate in order to achieve their goals or system goals. By communication, agents can coordinate their action and behavior, resulting in the systems that are more coherent. Coherence is how well a system behaves as a group [3].

2.3.1 Speech Acts

The communication among the computational agents can be done by modeling spoken human communication. Speech Act Theory [3] is a basis for analyzing human communication. In Speech Act Theory, the human natural language is considered as actions which can be a request, suggestions, commitments and replies. Speech Act theory has three main aspects namely, location (speaker’s physical utterance), illocution (speaker’s utterance meaning) and per-locution (locution’s result action).

2.3.2 Knowledge Query Manipulation Language (KQML)

KQML is a protocol that exchanges information and knowledge [3] [2]. The beauty of KQML is that the information to understand the content of message is included in the communication itself.

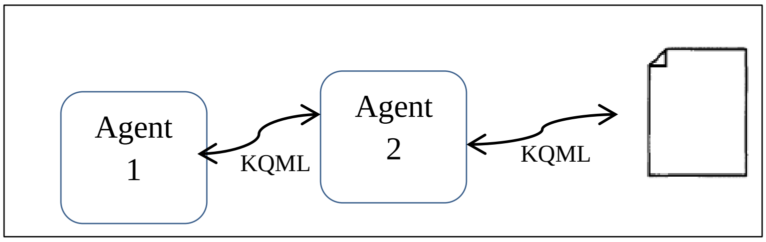


Figure 5- KQML working

Basic structure of KQML is:

```
(KQML-performative
                    :sender        <word>
                    :receiver      <word>
                    :language      <word>
                    :ontology      <word>
                    :content       <expression>
…)
```

KQML “wraps” the message in such a format that can be understood by any type of agent.

2.4 Agent interaction protocol

To send a series of messages, interaction protocols play an important role. The agents communicate by exchanging messages in order to accomplish the desired goals. The self-interested agents try to maximize their own utility but in case of common goal for all the agents, the objective is to maximize the overall system utility. The important aspects involved during the interaction are determining the shared goals and common tasks, avoid the conflicts those are unnecessary and collect knowledge and evidence. Various protocols are discussed in [3]:

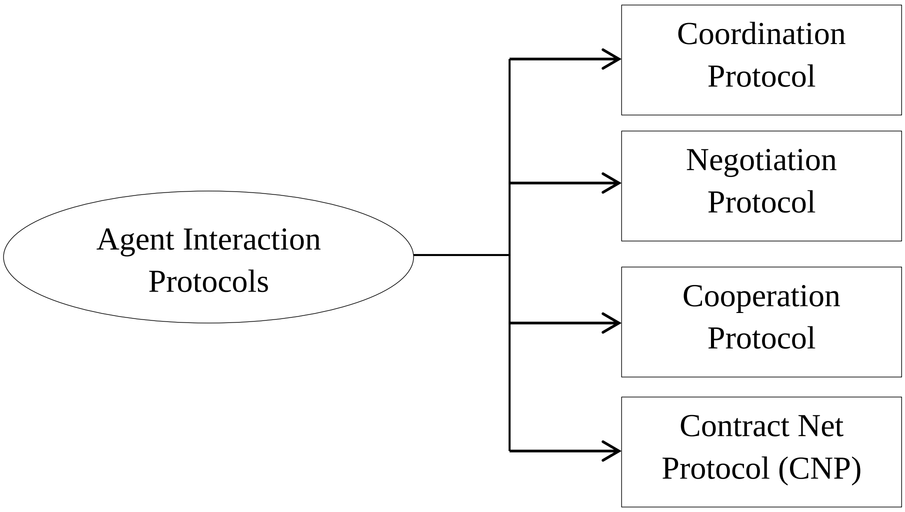


Figure 6- Existing Agent Interaction Protocols

2.4.1 Coordination Protocol

Coordination protocol allows the agent to satisfy both the individual and group goals. Coordination among the agents is required to maintain the dependencies between the agents or to achieve system goals or when agents have no sufficient competence, capability or information. These dependencies, actions and the required resources are represented by the AND/OR goal graphs.

2.4.2 Cooperation Protocol

The Cooperation protocols follow the strategy of Divide-and-conquer. The task is first decomposed and then distributed to multiple agents for its completion. There are various methodologies to decompose and distribute the task such as game theory approach, markov-decision based approaches, negotiation, auction-based market approach, and Swarm intelligence based methods. These methods will be explained in next chapter.

2.4.3 Negotiation Protocol

Negotiation is a process in which two or more agents reach to an agreement for achieving some desires or objective. The main features of negotiation protocols are the set of rules governed by the agents, language used for the negotiation purpose and the criteria for the agreement. Negotiation can be done in two manners: agent-centric and environment-centric. In environment-centric negotiation, the main emphasis is on the rules followed by the agents instead of agent’s capabilities. In agent-centric negotiation, agents are designed so as to fit in the existing environment. During negotiation, an agent may fall into one of the three states namely, conflict, compromising or cooperative [36]. In conflict state, the agent will act individually without any negotiation. In compromising state, the agent is forced to act so as to achieve the system goals and in cooperative state, the negotiating agents accepts all the requests and acts accordingly if they are capable to perform that task.

2.4.4 Contract-net Protocol (CNP)

The Contract-net protocol (CNP) is commonly used for the distributed task allocation in multi-agent system. CNP exists between the initiator agent (IA) and contractor agent (CA). CNP is based on the negotiation process where a task is announced by the initiator agent for completing the task. It assumes that the communication network is available for the agents to talk. FIPA has standardized contract net protocol. The flow diagram showing the working of CNP is depicted in figure 7.

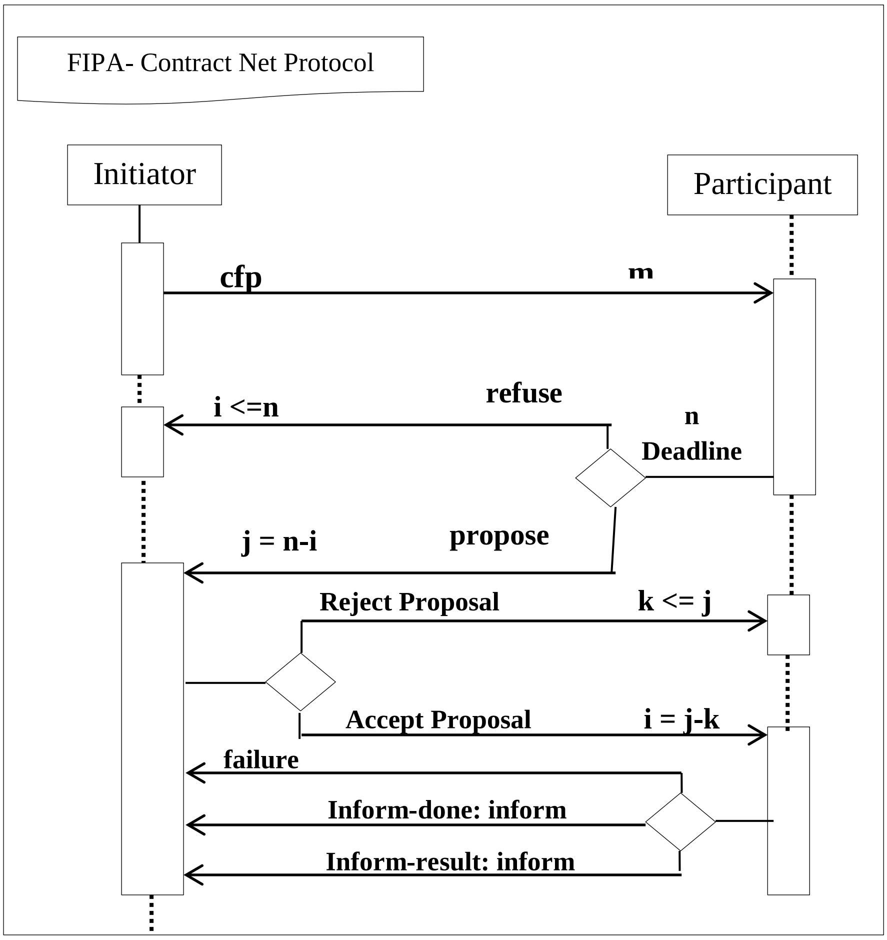


Figure 7- FIPA specified CNP [36]

CNP follows four phases for the task allocation namely, task announcement, bidding, awarding and task execution [37].

In task announcement phase, the initiator agent broadcasts the task announcement message to all the contractor agents for the required resources of new task. In bidding phase, CNP enables the contractor agents to evaluate the received task announcement message and decides whether to submit the bid for the respective task completion or not and sends the bidding message to sender accordingly. If the initiator agent doesn't receive any bid then it will repeat the task announcement phase again otherwise, it will go for awarding phase. In awarding phase, the winning contractor agent is selected on the basis of highest ranking bidder and the award message is sent to that winning contractor agent. After receiving the winning message the contractor will go for task execution.

CHAPTER 3

RELATED STUDY

A number of distributed task allocation algorithm for the dynamic multi-agent system have been developed: namely, OPGA [8] based on markov game theory, Auction and market based approaches [9], [10], DCOPs solution based approaches like LADCOP [11], SDPOP [12], distributed anytime algorithm [13] based on FMS, negotiation based approaches which include constraints optimization like CFSTP [14], and swarm intelligence based approach [15].

In recent years, many centralized and decentralized algorithms have been proposed for task allocation in cooperative multi-agent environment. The problem of task allocation and its relationship with overall system performance is a major research issue in distributed multi-agent system. The objective of each of the researchers was to find the solution of task allocation problem which gives maximum system utility and the successful accomplishment of task.

3.1 Task-Allocation

Task allocation is an important and challenging problem in Multi-agent systems. The problem is to assign a set of tasks to a set of agents in order to accomplish the maximum number of tasks successfully. There will be more tasks than agents thus agents need to schedule themselves to attempt each task in turn. In case of heterogeneous agents, where each agent may have different capabilities, agents communicate and negotiate with other agents and form the group of agents , called coalition, so as to successfully accomplish the requested task.

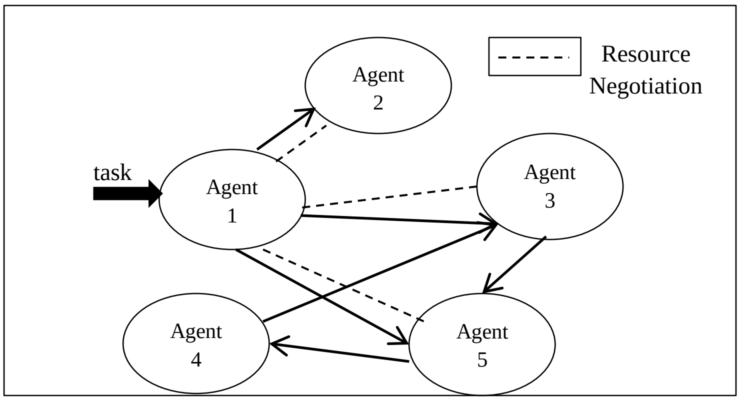


Figure 8- Task allocation Problem

In dynamic environment, coordination among the agents is essentially important because it enables a group of heterogeneous agents to find the best possible solution as the environment evolves. Task allocation can be done in two ways [5]:

- Centralized task allocation
- Distributed task allocation

In centralized approach, a central agent is used to allocate the tasks to cooperative agents. Here, single point of failure is usually inevitable which results in decreasing robustness of the system.

In Distributed approach, the task can be arrived at any agent and the agents communicate amongst themselves to complete the task and achieve the goals.

Example of task allocation problems include the allocation of sensing tasks to robots [6] and rescue tasks to ambulances [7]. The researchers gave both the centralized and distributed approaches for task allocation in static or dynamic environment. [6], [7] gave the centralized approach of task allocation where they didn’t consider the fact that agents or tasks may change over the time. Thus if the task allocation problem changes due to the arrival of a new agent or task, it need to be recomputed solutions from scratch. The main research factors in task allocation problem are:

- Coordination problem, after receiving the task, how to coordinate with other agent in an optimal way so as to fulfill the resources required for the completion of task.
- Coalition Formation, how to form an optimal group of agents so that the task is accomplished without any conflict.

Many researchers gave various approaches for finding the optimal task allocation in multi-agent systems. These are:

- Game-theory based approach
- Allocation based on markov decisions
- Auction based task allocation
- Negotiation based approach
- Distributed constraint optimization
- Swarm intelligence based approach

3.1.1 Game-Theory based approach

Here, each agent will be treated as a player and the process of allocating task to the coalition is strategy. The goal is to find the best strategy in the nash-equilibrium condition. For each player, the aim is to choose the strategy which will give its best payoff [5]. When each agent will choose its best strategy, no one will wish to deviate from their current strategy because they can't do any better than that. This is called nash-equilibrium condition.

In [8], Chapman defines a game-theoretic technique for decentralized planning to address dynamic task allocation named as OPGA. They considered that each agent has to perform a sequence of tasks where the tasks may require more than one agent for their successful completion. They considered that task is arriving dynamically in the environment. They formulated the task allocation problem as Markov game. But due to this formulation the agent's utility function became difficult to derive. Agent utility is the reward gained by the agent after performing the task and the global/system utility is the payoff gained by the whole system after accomplishing the task. They approximated the global utility using a series of static potential game and derive the agent's utility function. They also used the Distributed Stochastic Algorithm to find equilibrium in these games. Implementation was carried out on RoboCup Rescue simulator. The result shows that this approach outperformed the centralized and decentralized greedy approach and is robust to restrictions on the agents' communication and observation range. But this algorithm requires the continuous negotiation and doesn't consider the environmental changes.

3.1.2 Markov Decision based approach

The agents take the decisions on the basis of markov theory. Given the current state at particular time instant, the agent must have to take the action which results in optimal next state. For the markov game approach, agent must have either global or the partial view of the system.

Many researchers solved the task allocation problem of multi-agent system by using Markov Decisions Processes. In [16] the author presented a system designed for task allocation, staff management and decision support for scalable systems. The task is allocated to workers according to the user's requirements, different goals of the management, permanent staff and contractors. The system is designed on the basis of Contract Net protocol, belief theory and Markov Decision Processes

3.1.3 Auction Based Task Allocation

The task allocation can also be done on the basis of auction based market theory. Auction based task allocation is a type of centralized task allocation. There is a central auctioneer that is responsible for the task handling and allocation. When any task arrived at the central auctioneer than the auctioneer auctions for that task. Agents those are interested to perform that task sends their contribution to the central auctioneer. Then central auctioneer choose the winning agent whose contribution maximizes the overall system utility. The winning message is then sent to the winning agent to inform about the task execution.

In [17], the market-based allocation of the heterogeneous tasks to the heterogeneous agents was discussed. The authors have presented a heterogeneous task model and the metric task coverage for generating good heterogeneous teams. They used the sequential auction with the Team-Fit bidding mechanism.

3.1.4 Negotiation Based Approach

The agents negotiate with the other agents via some communication link for the efficient task allocation. The initiator agent if not capable to accomplish the received task individually then it negotiates with other agents in the system. Agents via negotiation form the coalition and then the coalition which maximizes the system utility has been chosen for the task allocation.

O. Shehory and S. Kraus presented an anytime algorithm in [7] for task allocation among computational agents via coalition formation. Here, the agent contacts to each other agents for their capability and make some agreement of coalition then choose the best coalition among disjoint and overlapping coalition. They also considered the task precedence ordering and allocate the task only when all its' predecessor tasks have assigned some coalition. This approach was implemented on RETSINA. The actual performance was 0.9 time the optimal performance. In worst case, the actual performance declined fast to less than 0.5 times of optimal performance.

In [18], the author constrained the agents' cooperation domain within a community i.e. the agent can only negotiate with its intercommunity member agents. This approach is inspired by the social sites like twitter or Facebook. They present their approach in three phases. First, task selection where the desirable task is to be selected preferentially. Second, allocation to community i.e. allocating the selected task to community based on significant task-first heuristics. Third, allocation to agents where the negotiation of resources for the selected task is done based on the non-overlap agent first and breadth first resource negotiation mechanism. In this community-aware model, because of dense intra-community connections, it is easy for a community member to cooperate, which will produce less system communication cost compared to the global-aware task allocation model. They concluded that their community model can be exploited well in large-scale applications because of the lower time complexity of the proposed algorithm. In this paper, the community was fixed during the task allocation however in reality the communities can be dynamic.

3.1.5 Distributed Constraint Optimization Problems

In DCOPs problem, each agent is given with a variable which has some assigned value whose domain is the action that an agent can perform. The objective function is to optimize some global constraint. From the literature surveyed there are various constraints that can be used in dynamic multi-agent systems. Like spatial constraint, temporal constraint, Communicational constraint etc. there are various DCOPs approaches like max-sum, Fast-Max-Sum, ADOPT, LADCOP etc.

A new Algorithm, Fast-Max-Sum (FMS) was proposed in [20]. The FMS algorithm is an extension of max-sum algorithm. It defines new function on variable and factor nodes. This reduces the number of states over which each factor has to compute its solution. Furthermore, the FMS algorithm allows each variable to decide when to send messages to other connected factor, when the factor-graph changes.

The author has further extended the FMS algorithm by applying online domain pruning and branch-and-bound methods as a novel approach [13]. This novel approach achieved 23% more utility, 31% less time and 25% less messages than other existing approaches in dynamic environment.

In [14], Ramchurn et. al. build the case for coalition formation with spatial and temporal constraint. They gave the MIP formulation for various constraints like completion constraint, deadline constraint, starting time, routing and service constraint etc. they also devised a new anytime heuristic for task allocation. They defined the set of feasible assignments and choose the best allocation which can accomplish the task in less time and can participate in more number of future tasks. CFTSP completes 97% tasks for the larger problems having 20 agents and 200 tasks.

In [21], ADOPT algorithm is proposed that converge to the optimal solution by considering only localized and asynchronous communication. This algorithm is based on the three key ideas, 1) agents explore the asynchronous partial solutions locally by using distributed backtrack searching. 2) For more efficient search, it uses backtrack threshold, 3) built-in termination detection. These ideas are responsible for the bounded-error approximation for performing trade-offs between solution quality and time-to-solution.

3.1.6 Swarm Intelligence based Approach

Swarm Intelligence has become a new field in the AI research, which is inspired by the social insect behavior that displays intelligence on the swarm level with simple interacting individuals. The swarm intelligence can be used for the task allocation in multi-agent system. In [15], the author presented the swarm based approach of task allocation. They implemented ant allocation algorithm for task allocation in random dynamic environment and perform task re-allocation when working condition changes. The author used hybridization of two approaches. For task selection, Honeybee model was used and then ant colony optimization is used. First of all, each agent is initialized with some response threshold. When task arrives at the system, the probability of selecting a task by the agent is calculated on the basis of response threshold. If Less response threshold then greater will be the chance of selecting that task. After finishing the task, the response threshold is updated similar to ant colony optimization.

3.2 Critical Review

We have studied various approaches for the task allocation in multi-agent system. As every system has some pros and cons so these approaches also have some benefits as well as shortcomings. Table 2 shows the critical review of the various task allocation approaches proposed by the researchers.

Table 3- critical Review

S.No.	Paper Title	Approach	Contribution	Shortcomings
1	Decentralized Dynamic Task allocation: A practical Game theory Approach, AAMAS, 2009	Overlapping Potential Game Algorithm	-Decentralized task allocation -tractable mechanism -consider future effect of agent's current action for decision window	-No partial contribution of agents -Continuous negotiation -doesn't consider the environmental change

2	Adaptive Task Allocation in multi-agent systems ACM, New York, 2001	Computational Market system	-Dynamic env. -Heterogeneous agents -Fairness in resource allocation - Adaptive MAS - Considers the type, deadline & priority of tasks	-Centralized approach -Communication overhead -resource manager overhead -reorganization cycle is fixed
3	A Distributed Anytime Algorithm for Dynamic Task Allocation in MAS AAAI, 2011	Fast-Max-Sum approach	-Dynamic env. -Heterogeneous agents -Less communication overhead -Less computation Overhead	-doesn't consider task preference -doesn't consider impact of future task -spatial & temporal constraints are not considered
4	Coalition Formation with Spatial and Temporal Constraints AAMAS, 2010	Mixed Integer Programming	-include spatial constraints -include temporal constraints -future task affect by CFLA -minimize comp. time of task and working time of agents	-homogeneous agents -one coalition can perform only one task at a time -static env.
5	Task Allocation in Multi-Agent Systems with Swarm Intelligence of Social Insects (ICNC-2010)	Hybridization of Honeybee Selection model & Ant colony optimization	-Random working env. -diff cost for diff category of tasks -learning method	-doesn't consider global maxima -time consuming approach for task completion
6	Community-Aware Task Allocation for Social Networked Multiagent Systems IEEE Transactions , 2014	Social Networked Multi-Agent Systems	-consider community constraint -significant-task first, non-overlap agent first and breadth-first heuristic is utilized -reduce communication Cost	-cooperative agents -centralized Algorithm -fixed community

The game theory approach outperforms the static applications rather dynamic application. The computational complexity in game theory approach is also very high. Robustness, scalability and adaptability are difficult to achieve in game-theory approach. The auction based approach depends on the communication link used for the negotiation between the auctioneer and the other agents. It leads to slow decision-making in case of unreliable communication line. Markov Decision processes results in more time consuming approach. As it searches all the possible states which give exponential time complexity. MDP also requires the complete view of the system which can’t be possible in dynamic environment. DCOP approaches require less communication overhead as compared to auction-based approach and MDP-based approach. Swarm-intelligence based approach considers the local maxima only but in our problem we require the optimization of global maxima.

3.3 Multi-agent System for Disaster Scenario

In this section the research done in task allocation for disaster scenario is going to be discussed.

Farinelli et.al [38] developed a multi-agent system based on RoboCup Rescue Simulator that allows the monitoring and the decision support needed for the rescue scenario. The authors developed a cognitive agent development kit that provides the ability of information fusion, planning and coordination required for the agent development. They performed a set of systematic simulation with different rescue scenarios so as to plan the actions whenever a prompt action is required in typical emergency scenario because of the partial information about the situation.

In [39], authors presented a multi-agent based framework that oriented towards the fire-fighting and suppression. They proposed a web-based fire-control system that assists fire-fighters and suggests the most optimal and feasible solution for controlling the fire. The overall architecture of the proposed framework works as follows: There is a user-interface agent that accepts the user request and forwards it to the global-cooperative agent. The global-cooperative agent is responsible for finding out the expert-agent for executing the requested task and forwards the request to Expert-system coordination agent. The ECSA reacts to the external request by selecting the appropriate expert agent for the task and assigns the task to that expert agent. The expert multi-agent system used in proposed approach comprises of house-fire agent, petroleum-fields fire agent, storehouse fire agent, petroleum tank fire agent and electronic station fire agents. The architecture also includes the external information agents like weather agent and traffic agent to give the information related to the weather and traffic to the other agents. The authors concluded that this prototype helps the user manager fire by enhancing the decision process and deriving the optimal response.

Yunbo lu and his colleagues developed an agent-based model to study the fire-fighting team’s performance [40]. They focused on the relationship between the distributions of fire-fighting team’s authority and its performance. They considered two types of authority distribution factor namely, the supervisor-centered factor (rescue factor and fire-control factor) and self-management factor (fire putout factor). The authors showed that the high performance can be obtained only when the supervisor-centered factors are in the state of supervisor-centered and the self-managing factors are in the state of self-management. They also showed that the relationship between the authority distribution and the team performance is non-linear and self-managing factor has a greater impact on the team performance.

In paper [41], the authors proposed a new algorithm based on the earliest deadline first for the coalition formation. They grouped the rescue teams for various rescue missions. They also presented the ungrouping of team after performing the assigned rescue mission and then create the new rescue teams on the basis of new situations of the environment. They used the earliest deadline first algorithm for solving the ambulance problem and coalition formation. In ambulance problem, the task is rescuing the victims and the task deadline is the time to death for a victim. For rescuing the victim, they sorted the civilian victims based on the time to death and the first candidate is selected for the rescue operation. Calculate the coalition size i.e. the number of ambulances needed to rescue the civilian on time. If it is possible to rescue that civilian according to the time then go for rescuing it otherwise remove the candidate from the victim list and go for selecting the next civilian.

Beatriz Lópaz and his colleagues presented a multi-agent system for coordinating the ambulances in emergency medical services [42]. The system is responsible for assigning the most appropriate ambulance vehicle for the emergency patient transportation. In this paper, the authors combined the auction protocol with trust model and fuzzy filter. The trust model deals with the driver’s expertise. This results in inclusion of more number of variables in the decision process. They also improved the decision making regarding the ambulance distribution by maintaining a region coverage strategy. The proposed system ensures that the patient receives the proper treatment by providing the quick response to the emergency request. In the proposed architecture, the ambulance coordinator agent receives the request from the external agents like patient’s location, first-aid of patient, transporting the patient to appropriate hospital. On receiving the service request, the coordinator agent assigns the services to the appropriate ambulance team agent. The assignment of the ambulance team to appropriate service has been done by using the contract-net protocol. Here, the coordinator announced the service request to the team agents. The team agents respond the request by sending the estimated arrival time with a bid. Using the winner determination algorithm, which choses the best ambulance team for the requested service, the coordinator selects the ambulance team agent to which it will assign the service. If the human coordinator agrees with this suggestion then coordinator will informs the team agents and external agent about this ambulance assignment.

By reviewing these papers, we devise a distributed task allocation approach for fire-fighting scenario.

CHAPTER 4

MULTI-AGENT SYSTEMS: JADE FRAMEWORK

Agent based technologies are widely used in distributed environment to design the complex distributed systems with less effort. Agents are autonomous in nature i.e. they take their own decisions without any user interventions. When agents work together to achieve the common goal then the system is known as multi-agent system. A lot of frameworks are available to develop the agent based systems like FIPA-OS, JADE, JACK Intelligent Agent, and JLAC. These frameworks provide some pre-defined agent tools and models to help the developer to design the multi-agent system easily.

4.1 FIPA Specification

The Foundation for Intelligent Physical Agent (FIPA) is a non-profit International association of organizations and companies which was registered in Geneva, Switzerland. They aim to produce the standards for generic agent technologies. FIPA was originated to produce the standard specifications for the agents which interact with one another and are heterogeneous in nature. FIPA is not only applicable for a specific application rather it is a generic technology for different applications. It is a set of basic technologies which is integrated by the several developers in order to develop the complex systems with high interoperability. FIPA was officially accepted by IEEE as its eleventh standard committee on June 8, 2005 [32].

Table 4 shows FIPA-97 and FIPA-98 specifications and their parts [33].

Table 4- FIPA-97 and 98 Specification [33]

FIPA- 97 Specification		
	Normative	**Informative**
Part1	Agent Management	
Part2	Agent Communication Channel (ACC)	
Part3	Agent Software Integration	
Part4		Personal Travel Assistance

Part5		Personal assistant
Part6		Audio-Visual Entertainment and Broadcasting
Part7		Network Management and Provisioning
FIPA-98 Specification		
	Normative	**Informative**
Part8	Human Agent Interaction	
Part9		Product Design and Manufacturing
Part10	Agent Security	
Part11	Agent Mobility	
Part12	Ontology Service information, application, specification	

The first output document if the FIPA specification was FIPA-97. FIPA-97 described the reference model for agent platform. This model is shown in the Figure 9.

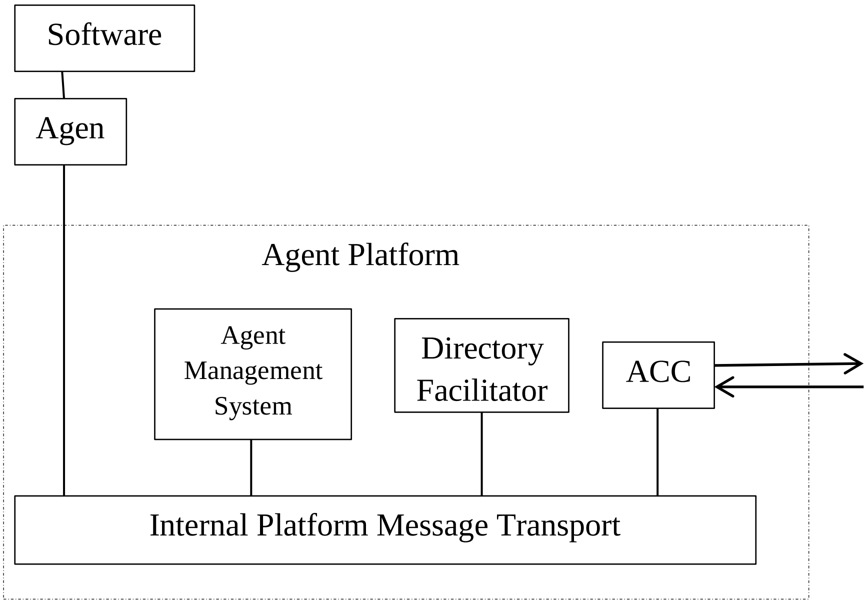


Figure 9- Reference model of FIPA-97 specification [34]

FIPA-97 includes seven parts. The first three parts namely Agent Management System (AMS), Agent Communication Channel (ACC) and Directory Facilitator (DF) are of normative type [34]. They emphasize on the technical aspects of multi-agent systems. It identifies the roles of key agents that are required for the platform management and specifies the agent content language for its management and its ontology. AMS supervises the control to use and access the platform. It controls the registration and authentication of resident agents. ACC enables the communication between the agents inside and outside of the platform. It supports IIOP for the interoperability between the different agent platforms. DF provides yellow page services to the agents. The next four parts of FIPA-97 explains the use of AMS, ACC and Agent/Software integration to implement the applications like Personal Travel Assistance, Personal Assistant, Audio-Visual Entertainment and broadcasting and Network Management and provisioning [33].

FIPA-97 also specifies the Agent Communication Language for allowing the communication among agents [36]. It is based on the message-passing scheme where agents communicate with each other by formulating and sending messages. FIPA ACL specifies the encoding, semantics and the pragmatics of messages required for the agent communication. The syntax of ACL is very similar to the existing communication language KQML.

The second version of FIPA-97 is launched in 1998 named FIPA-98. It describes 6 parts [33]. Out of 6, the normative specifications are Human/Agent Interaction, Agent Security, Agent Mobility and ontology Service whereas the informative specifications are product design and manufacturing and FIPA-97 Developers' guide.

4.2 Java Agent Development Framework (JADE)

JADE is a software framework which allows the development of agent- based applications. It compliances with FIPA standard therefore achieves high interoperable intelligent multi-agent systems. JADE makes the development simpler through a complete set of system services and agents. The following list of features is offered by JADE so as to achieve an inter-operable intelligent multi-agent system [34]:

- FIPA- compliant Agent Platform, it includes three normative-type key agents namely, AMS, ACC and DF. These agents are automatically activated with the start-up of agent platform.
- Distributed Agent Platform, distributed environment can be achieved by splitting the agent platform into several hosts. A single JVM will be executed on each host. The agents are implemented similar to the java threads and parallelism can be achieved by executing the several task s by a single agent in parallel. These parallel tasks are scheduled in a more efficient manner than JVM.
- In order to implement multi-domain application, a number of FIPA-compliant DFs can be started at run time.
- To simplify the registration of agent services with more than one domain, a programming interface is provided.
- To send/receive messages to/from the agents, transport mechanism and interfaces are provided.
- Different platforms are connected via FIPA-97 IIOP protocol.
- Light-weight transport of ACL messages within the same agent platform
- Libraries are specified to access FIPA interaction protocols
- AMS registers the agent automatically
- At the start-up, agents obtain their Global Unique ID (GUID) from the platform.
- To manage the agents and agent platform, graphical user interface is provided.

JADE Agent Platform agrees with FIPA-97 specifications. It includes all the mandatory agents that manage the agent platform. The communication among the agents are done via message passing through Agent Communication Language i.e. Agent ACL.

The coexistence of the multiple JVMs is the basis for software architecture of JADE. The communication between different VMs and event signaling within a single JVM relies on Java RMI (Remote Method Invocation). In JADE, a multi-threaded execution environment is provided by the Agent Container [34]. Each agent is corresponds to one thread and Message dispatching is done through system threads those are spawned by the RMI runtime system.

The software architecture of one JADE agent platform is shown in the fig. 10.

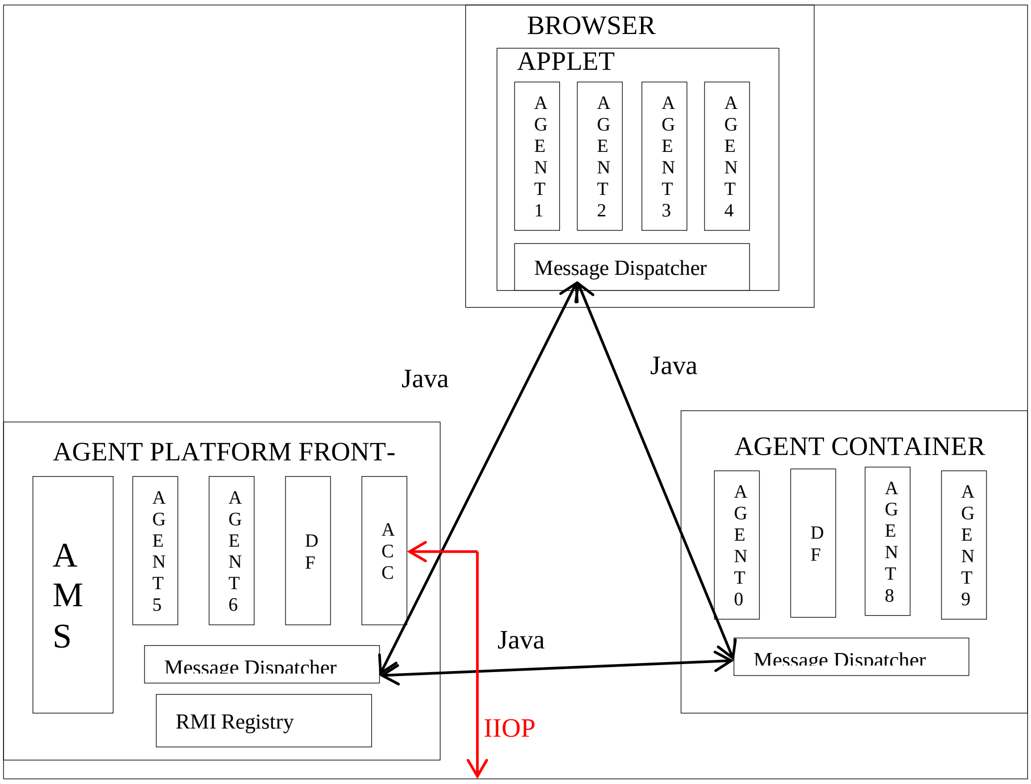


Figure 10- Software Architecture of single JADE Agent Platform [34]

4.2.1 Agent Container

Agent Container is a RMI server object which manages the set of agents locally. It is responsible for executing an agent. The life cycle of agent consists of four stages namely, agent creation, agent suspension, agent resuming and agent killing. The communication aspects like dispatching of incoming ACL messages, routing the message to destination, store them into message queue of private agent, outgoing messages are also handled by the Agent container.

4.2.2 JADE Communication System

Whenever a new Agent Container is created, it registers itself in a RMI registry which is maintained by the JADE front-end container. This registry is then stored in the Agent Container Table. An Agent Global Descriptor table is also maintained to store the name if each agent with its AMS data and RMI object reference of its container. When a new front-end begins, an internal RMI registry is created on the current host. This registry listen the specific TCP/IP port and start with the FIPA agents system. Whenever a container sends a message to another container, it caches the object reference of that container. It increases the performance of the system by avoiding the looking-up of the Agent Global Descriptor table each time whenever a message is sent.

The three cases can be possible when JADE agent send a message [34]:

1. Within same agent container, the message is passed in the form of a Java object using an event object without any message translation.
2. Within the same JADE platform but different container, the Java RMI framework is used to send the ACL message.
3. For the different agent platforms, FIPA compliance-standard IIOP and OMG IDL interfaces are used to send the ACL messages.

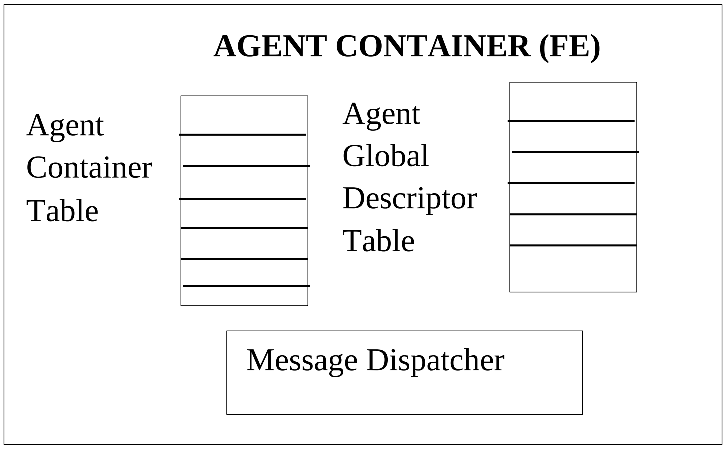


Figure 11- Front-End Agent Container [34]

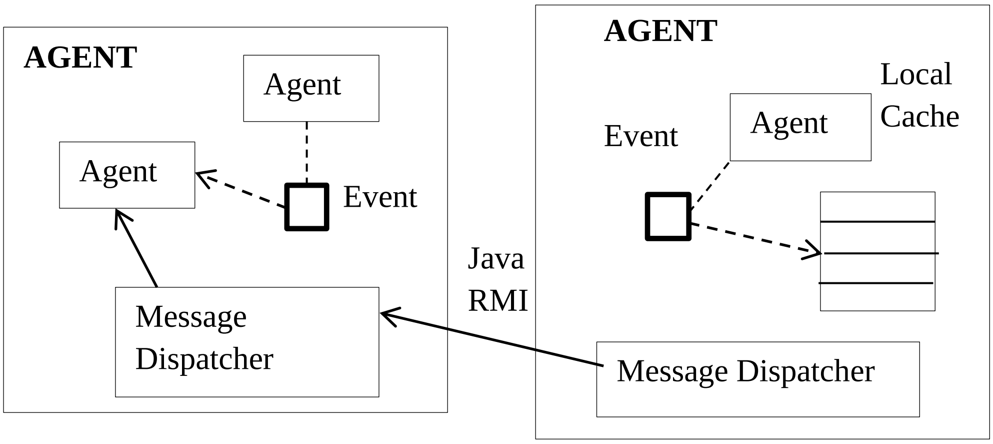


Figure 12- JADE intra-platform communication model [34]

4.2.3 Agent Execution Model

The actual task that an agent can perform is carried out within “Behaviour” class and agents instantiate their behaviours according to the requirements and capabilities. JADE runs the agent platform by using the thread-per-agent concurrency model instead of thread-per-behavior which results in less no. of threads generation. To execute a task, agent creates an instance of corresponding Behaviour subclass and call the addBehaviour() method of the Agent class. Each Behaviour class must implement two methods namely, action() method and done() method. action() method represents the “true” for the task that must be performed by the specific Behaviour class. done() method is used by the agent scheduler which returns “true” if action of behavior is finished and can be removed from the queue otherwise returns “false”.

On the basis of tasks executed by the agent, several types of behaviours are defined in JADE framework. These are as follows [35]:

1. SimpleBehaviour: This is used to implement simple actions of the agent.
2. ComplexBehaviour: This is used to implement those Behaviours which are composed of several sub-Behaviours. Agent scheduler follows the FIFO policy i.e. selects the top-most task for the execution. After accomplishing the top-most task, it assigns the control to next task in the ready queue.
3. OneShotBehaviour: The actions which must have to be accomplished only once are modeled by this class.
4. Cyclicbehaviour: It models those atomic actions that never ends and executed until the agent is killed.
5. SequentialBehaviour: It is ComplexBehaviour that executes the sub-behaviours in a sequential manner.

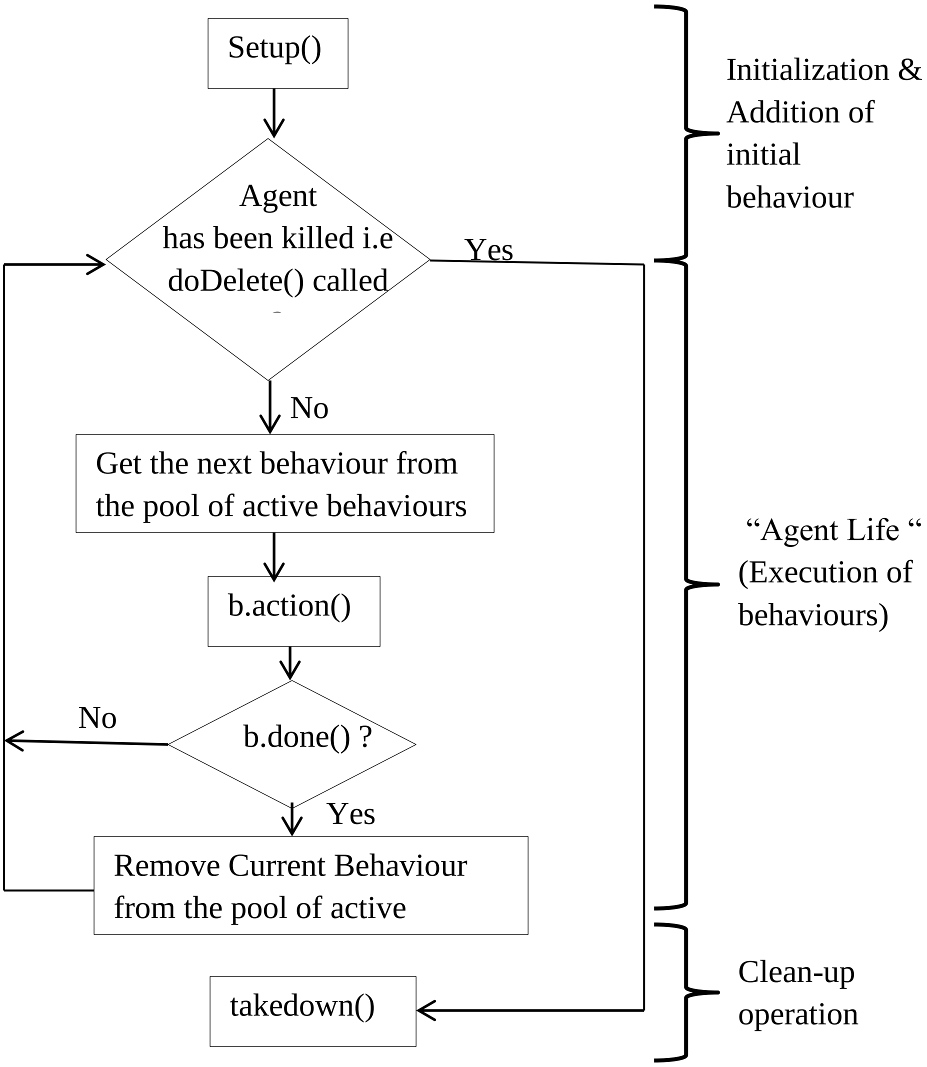


Figure 13- Agent Execution model [35]

The development of JADE is still growing. Further implementations, enhancements and improvements have already been discussed. For example, the support for agent mobility has been included in FIPA-98 specification. JADE enables the agent developer to develop the complex multi-agent system in a very effective, easy and efficient manner.

CHAPTER 5

PROPOSED APPROACH: TASK ALLOCATION IN FIRE FIGHTING

This chapter presents the proposed task allocation approach for fire-fighting scenario. This approach aimed at improving the waiting time and the communication cost during the task allocation in constrained-cooperative multi-agent environment. The fire-fighting scenario is considered for the research work. The problem statement & description, formulization and algorithms for proposed approach are discussed in this chapter.

5.1 Problem statement

The task allocation problem in multi-agent systems is the problem of allocating task to the agents so as to maximize number of successfully accomplished tasks and the system utility. In case of complex task where the task can’t be accomplished by a single agent, a group of agents are formed which requires some sort of coordination and negotiation between multiple agents. Thus the main issue in task allocation problem is the coalition formation and coordination among multiple agents. To optimize the task allocation problem, there is need of appropriate coordination and coalition formation mechanism.

The main problem addressed in this thesis is to improve the coordination and coalition formation mechanisms in order to optimize the task allocation algorithm in constrained-cooperative environment. Due to bad coordination among agents, these environments result in lower performance. The lack of coordination among agents in multi-agent systems is caused due to the inefficient task allocation among agents. The task allocation guarantees agents an efficient determination of goals and successful execution of tasks which permits agents to achieve their goals in a cooperative way. Therefore, it is necessary to create and design the new task allocation and coordination mechanism so that the agents can make efficient decisions in such complex systems.

5.2 Problem Definition

The problem addressed in this thesis is to optimize the task allocation problem in constrained-cooperative multi-agent system by improving the coordination and coalition formation mechanism. The fire-fighting scenario has been taken for the proposed approach. This scenario is highly dynamic, uncertain and real-time in nature. When any fire incident occurs in the environment, the time to allocate the appropriate fire brigades is very crucial. It is required to allocate the fire-brigade which will take less time to reach at the destination. Thus spatial, temporal and communicational constraints are considered in the proposed task allocation algorithm.

The proposed approach optimizes the task allocation approach by performing the following objectives:

1. **Coordination mechanism**

   In multi-agent systems, the coordination is done via message passing. Various coordination approaches have been proposed which are already discussed in chapter 2. In the proposed approach, Contract-net protocol (CNP) is used to allow the coordination and negotiation among the multiple agents with some improvements. In the proposed approach, the coordinator will send the task request initially to only that agent which is nearest to the event location instead of sending the request to all the agents available in the system. If the receiving agent is capable to accomplish the task alone then it will inform to the coordinator and the task is assigned to that agent otherwise coalition will be formed by the receiving agent according to the CNP mechanism. This approach results in less number of message transferred than conventional CNP.

2. **Coalition formation**

   In case of fire-fighting scenario, time to allocate the fire brigade agent is very crucial. The proposed approach considers the two factors while forming the coalition i.e. earliest start time and the trust model. The agent or group of agents which can arrive to the event location early and has largest trust factor will be chosen as the winner. The trust factor determines the driver’s expertise of the fire-brigade agent. It results in less waiting time and maximizes the number of successfully extinguished fire events.

3. **Re-planning algorithm**

   In dynamic environment, the execution errors may occur due to the uncertainty and failure of action. An essential part of the planning system is re-planning. In fire-fighting system, the action may get failed due to some obstacle arrived when a fire-brigade is travelling towards the event location like road blockage. In such cases, re-planning is required. In the proposed approach, whenever an obstacle is detected, the fire-brigade agent will re-start negotiation for the required capability with the other agents in the system. If no set of agents will satisfy the requirement of the init-agent (agent who detected the obstacle) then this init-agent will follow some alternate route to reach to the event's destination. This will maximize the success rate of fire events.

5.3 Problem Formulation

The proposed fire-fighting multi-agent environment consist of three types of agents namely, fire station agent (FSA), fire-brigade agent (FBA) and fire-event agent (FEA). For each and every fire-brigade vehicle, fire-brigade agents are created. Fire-brigade agent is concerned with the location of its respective fire-brigade vehicle, its capacity of water, status and the local view of the environment in its surrounding. The fire-station-agent has the global view of the environment that is the knowledge of event-agent like location and the required capacity of water to extinguish the fire-event and the knowledge of fire-brigade like location, number of success and failure of events for the fire-brigades available in the system. The fire-brigade agents are heterogeneous in nature in the sense that the capacities of water of each fire-brigade agents are different to one another.

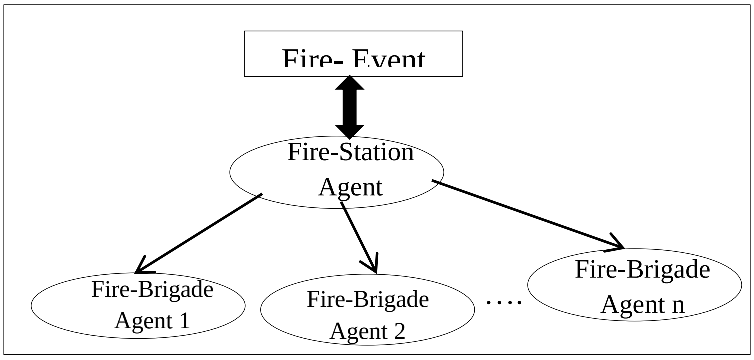


Figure 14-A multi-agent System architecture for assigning Fire-Brigades to fire event

Whenever a fire event occurs, FEA get generated with the event location and event's required water capacity and FSA gets called. It will get the event location and required capacity. It assigns a unique name to this event for the sake of coordination and task allocation problem. It coordinates with the available FBAs and informs FEA about the assigned FBAs. FEA then reports either success or failure for the event. And then that FEA gets killed.

Assumptions:

- The agents are heterogeneous in nature in terms of the capacity of water tank they have.
- Agent can perform only one task at a time.
- The occurrence of fire-event is dynamic and the arrival of fire-event follows the Poisson distribution.
- Task allocation is done in a distributed manner.
- The communication channel is considered to be reliable.
- Agents are cooperative in nature, means whenever they have required capacity and ideal, they will co-operate the other agent and after starting the execution of any task, the agent cannot leave the system before its completion.
- FSA has the global view of the system whereas FBA has the local view of the environment.
- Coordinate plane system is used to locate the fire-brigade and the fire-event in the environment for the sake of simulation.
- Fire-brigade follows the straight line to reach to the event location. There is only one route to reach to the event location.
- To calculate the distance between fire-brigade location and fire-event location, Euclidean distance is used.
- Only road-blockage condition is considered as an obstacle.

Let us consider the multi-agent system consists of one FSA and n FBA i.e. FBA= {FBA_1, FBA_2, FBA_3…. FBA_n}. Here each agent will possess a unique ID.

The pseudo-code of the proposed approach is given into the Appendix-B.

5.3.1 Agent Definition

This section presents the Agent formulation for the proposed approach.

a) Fire-Station Agent

Fire-station agent has global view of the system. It contains an agent list which stores the location of each fire-brigade vehicle provided by the corresponding FBA. Whenever an event occurs, it stores the location, required-water-capacity and arrival time of the event in a list named eventlist. FSA can be formulized as:

FSA-ID: the unique id of FSA generated by the agent platform
agentlist, the list of FBA's location,
<ID(FBA$_i$), x(FBA$_i$), y(FBA$_i$), no_of_success(FBA$_i$), no_of_failure(FBA$_i$)>
eventlist, list of event invoked in the system,
<event_nm, x(ev), y(ev), eventcap(ev), arrivaltime(ev)>

b) Fire-Brigade Agent

For each fire-brigade vehicle available in the system, FBA will be created. FBA can be formalized as:

FBA$_i$-ID: unique ID of FBA$_i$ generated by the agent platform
x(FBA$_i$): the location of corresponding fire-brigade at x-axis
y(FBA$_i$): the location of corresponding fire-brigade at y-axis
cap(FBA$_i$): the amount of water in the water tank of corresponding fire brigade
speed(FBA$_i$): the speed of corresponding fire-brigade vehicle
status(FBA$_i$): the status of corresponding fire-brigade. Here, three type of status has been considered namely, "active", "busy" and "inactive".
"Active", when fire-brigade is ready for assignment
"Busy", when fire-brigade is assigned for some other event
"Inactive", when fire-brigade is in refilling or recovery state
neighborlist, the list of neighbors of FBA$_i$.
<ID(neighbor$_i$), x(neighbor$_i$), y(neighbor$_i$), success(neighbor$_i$), failure(neighbor$_i$)>

5.4 Proposed algorithm

The proposed approach is divided into four phases namely,

- Task Arrival

- Resource Negotiation
- Coalition Formation
- Task Execution

The flowchart for the proposed algorithm is shown below:

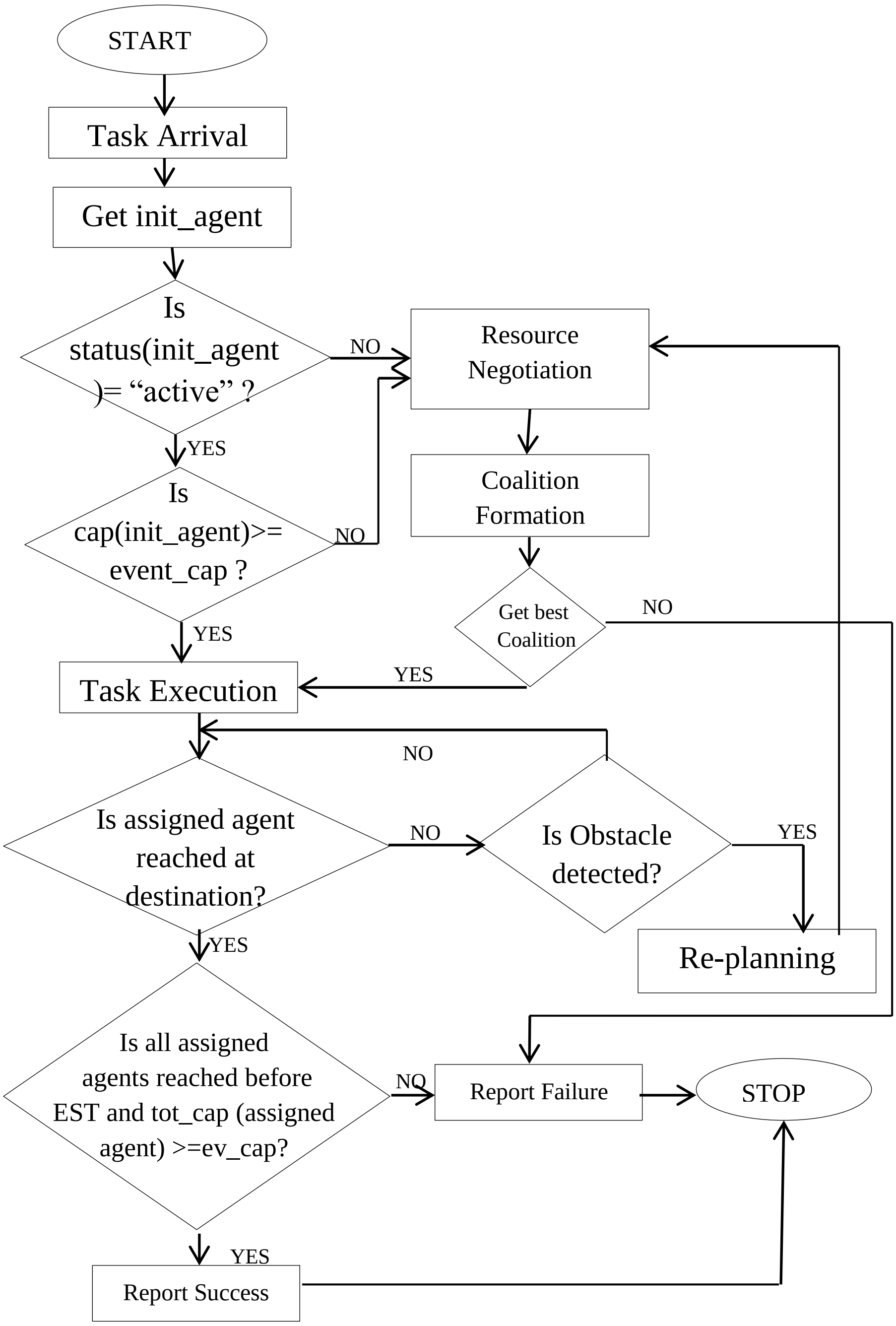


Figure 15- Flowchart for the proposed algorithm

Initially when the system gets started, FSA initializes the agent list with the location of each fire-brigade agent.

5.4.1 Task Arrival

This phase encounters when the fire-event occurs in the environment and invokes the fire-station agent.

Algorithm:

1. Event ev will invoke the FSA with its location and required capacity.
2. FSA find the nearest fire-brigade from its agentlist by calculating the minimum Euclidean distance between the event location and fire-brigade location and send the event request to corresponding FBA. This FBA is named as init-agent.
3. On receiving the event-request, init-agent will check its status.
    a. If the status is “active” then it will check its capacity in water tank.
        i. If the cap (init-agent) > = eventcap then init-agent will send OK message with its expected start time (EST), capacity to FSA. <EST, cap (init-agent)>
        ii. Else init-agent will go for “Resource Negotiation(eventcap – cap(init-agent))”
    b. Else init-agent will go for “Resource Negotiation(eventcap)”
4. If FSA receives OK message from the assigned agents, FSA will send CONFIRM message to init-agent and informs to FEA about the assigned agent and EST.
5. Else FSA will report failure.
6. On receiving the CONFIRM message, FBA will go for “Task Execution”.
7. If any obstacle detected by the assigned FBA the go for “Re-planning”.
8. If the assigned FBAs reached at the event location before the EST, then event agent will report success and inform to FSA which shows the successful task allocation and FSA will record this time as completion time.
9. Else it will report failure to FSA.
10. FSA also record the number of success or failure of the assigned agent on the basis of success or failure of the event. This record is used to evaluate the trust factor of respective FBA.
11. The waiting time for the event is calculated as

$$\text{Waiting time} = \text{completion time} - \text{arrival time} \qquad (1)$$

The numbers of message transferred are calculated by counting the message during the communication among multiple agents in multi-agent systems.

5.4.2 Resource Negotiation

This phase encounters when the init-agent is not able to fulfill the event's capacity. In this phase the agent will negotiate with other agent. The resource negotiation mechanism used in this proposed approach is based on the CNP protocol.

Algorithm

1. Init-agent sends the event request to its neighbor FBAs with the required capacity of water.
2. The receiver FBAs will check their status.
3. If the status is "active" then it will send the ACCEPT message to the init-agent with its EST and capacity.
4. Else it will send the REJECT message to init-agent
5. On receiving the response from all the FBAs, init-agent will go for "Coalition Formation" for the agents who have sent the ACCEPT message.
6. The best coalition will be chosen from all the possible coalition.
    a. If no coalition is possible that satisfies the required capacity then init-agent will send CANCEL message to FSA
    b. Else go for step-7 to step-13.
7. Init_agent send the INFORM message to all the member of winning coalition.
8. On receiving the INFORM message, the receiving agent will check its status again.
9. If the status is "active" then it will send the OK message to the init-agent.
10. Else send the PRONE message to init-agent.
11. If all the winning agent responds with OK message then init-agent will send the CONFIRM message to those agents and OK message to FSA with the set of assigned agent and its EST.
12. Else init-agent will send the CANCEL message to all the winning agents and to the FSA.
13. On receiving the CONFIRM message, the FBA will go for "Task Execution" phase.

Here, one assumption has been taken that once a FBA send the OK message for one event, it will not allow sending OK message for another event until it receives the CANCEL message or accomplish the assigned task because agents are co-operative in nature.

5.4.3 Coalition Formation

If the required capacity of fire event is not fulfilled by the init-agent then coalition will be formed. In this proposed approach the coalition is formed on the basis of earliest Expected Start Time (EST) and the trust factor of the fire-brigade agent.

Algorithm

```
1. Make the power set of all the agents who sent ACCEPT message to init-agent.
2. For all the set $S_i$ Є powerset
     a. For all $FBA_j$ Є $S_i$
     b. $tot\_cap(S_i) += cap(FBA_j)$
     c. If $tot\_cap(S_i) >= required\_cap$
     d. Add $S_i$ to the coalition C // coalition which satisfies the event's capacity
3. End for loop
4. For all $S_i$ Є C
     a. For all $FBA_j$ Є $S_i$
          i. Chose the maximum starttime among all the $FBA_j$ and set it as Starttime ($S_i$)
     b. End for loop
     c. Arrange $S_i$ according to ascending order of the starttime($S_i$)
5. End for loop
6. Choose the set $S_i$ having minimum starttime. //coalition having min. EST
7. If there are more than one sets having same and minimum starttime
     a. Then chose the coalition with smallest size. It is done because if less number of agents is engaged in performing a particular task then chance of assigning other agents to future task will become high.
          i. If there are more than one sets having same and smallest coalition size
          ii. Then the set having maximum trust factor will be chosen. The trust factor is used to determine how much a particular agent is trust-worthy for accomplishing the task according to the driver's expertise.
          iii. Return the chosen set as the best coalition.
     b.Else return any set having less  number of agent
8. Else return any set having minimum starttime($S_i$)
```

5.4.4 Task Execution

This phase will encounter when FSA assigns the fire-event to the chosen FBA. On receiving the CONFIRM message from the sender, the FBA will go under this phase. FBA will set its status as busy and assigned for the received fire-event.

Algorithm

1. When winning agent receives the CONFIRM message from the init-agent then it sets its status as “busy”.
2. The fire-brigade corresponding to that FBA will move towards the event’s location.
3. If any obstacle is detected then go for “Re-planning”.
4. If assigned FBA reached at the location before the Expected Start Time
    a. if tot_cap(reached_agent) >= ev_cap
        i. Then FEA will report success to the FSA
    b. Else FEA will report failure.
5. Else FEA will report failure
6. After extinguishing the fire, the assigned FBAs will change their status as “inactive” and go for refilling.
7. After refilling phase, the fire-brigades reach to their base location and update their state as “active”.

5.4.5 Trust Model

The trust, a Fire-Station Agent has in its Fire-Brigade Agent, is its faith that the agent can accomplish the assigned task successfully. The fire-brigade agent with a “skilled” driver should have highest trust factor because it is expected that it can reached to the destination more easily whereas a “beginner” driver could have a lowest trust value. “Skilled” or “beginner” is related to the knowledge of area/regions covered by the fire-brigade.

To calculate the trust factor of FBA, Jigar Patel and his colleagues’ applied the probabilistic approach to trust. This trust model is used in the proposed approach in order to get the best coalition for the requested event. They defined trust as a value in the [0, 1] interval, 0 means completely untrustworthy agent and 1 means complete reliability. Due to the insufficient information for defining the probability of trust, the authors propose using the expected values given in the previous experience of all interaction outcomes. Thus, the trust value t_j for the FBA_j can be calculated as:

$$t_j = \frac{\alpha}{\alpha + \beta} \tag{2}$$

where,

$$\alpha = s_j + 1$$

$$\beta = u_j + 1 \tag{3}$$

Here s_j is the number of past successful task accomplished by FBA_j and u_j is the number of unsuccessful task assigned to FBA_j.

5.4.6 Re-planning Algorithm

The re-planning algorithm will be called when any obstacle is detected by the FBA while travelling toward the event-destination.

Algorithm

1. FBA will send the event request to neighbor FBAs for the capacity equal to its own capacity
2. FBA will form the coalition for all the agents sent ACCEPT message.
3. FBA will choose the group of agents, C which satisfies the required capacity.
4. For all S_i Є C
    a. Chose the coalition C_b for which
        i. starttime(S_i) <= EST(FBA) and smallest coalition size
5. end for loop
6. if C_b is non-empty
    a. Then agent will send the INFORM message to all FBA_j Є C_b
    b. On receiving the OK message form all FBA_j Є C_b, FBA will send them CONFIRM message
    c. On receiving the CONFIRM message, FBA_j will go for "Task Execution".
7. Else FBA will choose some alternate route and go for the event's execution.

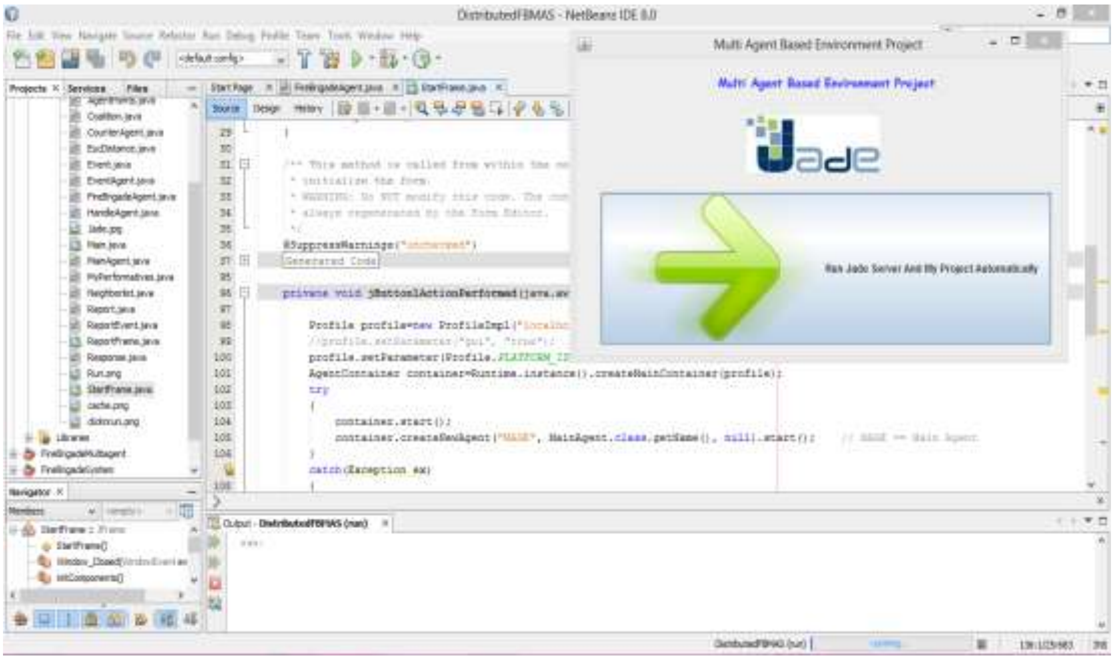
Instead of sending the CANCEL message directly after detecting the obstacle, The FBA will go for re-planning so as to maximize the success rate.

CHAPTER 6
EXPERIMENTAL SETUP

This chapter represents the snaptsshots of the experimental setup which was used for simulating the proposed approach.

6.1 Start-up Frame

Figure 16, is the startup frame generated with the intiatialization of Agent Platform. As the project starts, the agent container gets started in JADE framework. The Agent platform starts the agent container. This container is then create the Fire-Station Agent. The Fire-station agent gets created and obtains a unique ID from agent platform.



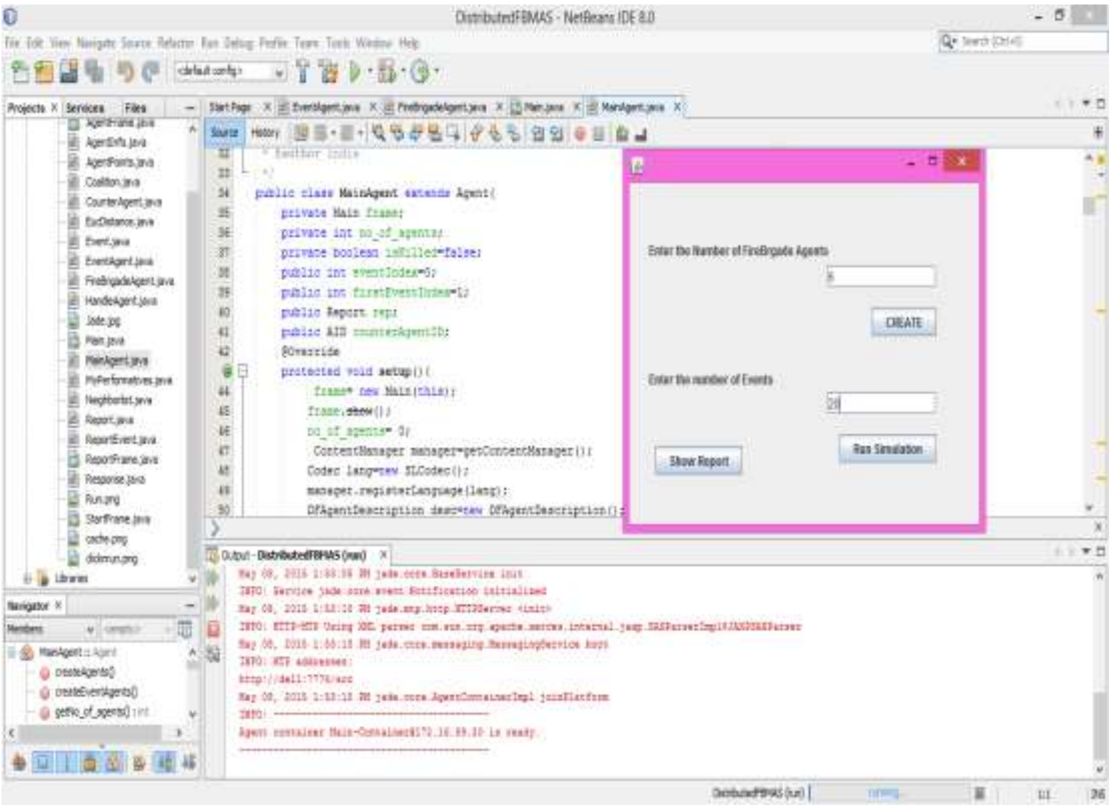
```
@SuppressWarnings("unchecked")
Generated Code

private void jButton1ActionPerformed(java.aw

    Profile profile=new ProfileImpl(
    profile.setParameter(Profile.PLATFORM_ID
    AgentContainer container=Runtime.instance().createMainContainer(profile);
    try
    {
        container.start();
        container.createNewAgent("MASE", MainAgent.class.getName(), null).start();
    }
    catch(Exception ex)
    {
```

Figure 16- snapshot of start-up frame

In the figure 16, on clicking the arrow, the Fire-Station agent gets created in the Agent Platform. Fire-Station agent is responsible for the monitoring the global view of the environment. It keeps the record of all the fire-brigade agents, their location, and their success count and failure count. Any fire event call is received by the fire-station agent. The fire-station agent then allocates the appropriate fire-brigade agent to extinguish the fire with minimum response time.

6.2 Main Frame

When the Agent platform gets intitialized, the Fire-Station agent will be created. The Fire-Station Agent invokdes the “MAIN” frame window . This is the main frame in which user will enter the number of fire-brigade agents available in the environemnt and the number of fire-event for which the simulation has to be done. This is shown in figure 17.



```
     * @author india
     */
public class MainAgent extends Agent{
    private Main frame;
    private int no_of_agents;
    private boolean isKilled=false;
    public int eventIndex=0;
    public int firstEventIndex=1;
    public Report rep;
    public AID counterAgentID;
    @Override
    protected void setup(){
         frame= new Main(this);
        frame.show();
        no_of_agents= 0;
         ContentManager manager=getContentManager();
        Codec lang=new SLCodec();
        manager.registerLanguage(lang);
        DFAgentDescription desc=new DFAgentDescription();
```

Figure 17- snapshot of Main-Frame and Fire-Station agent

The fire-station agent is responsible for receiving the fire-brigade agent details whenever FBA gets created into the system. FSA also reduces the number of fire-brigade agents when any FBA gets killed. The success and failure of the fire-brigade allocation is also reported by the fire-station agent. Thus fire-station agent is responsible for coordinating the whole simulation environment.

After the creation of Fire-Station Agent, the agent will call the “MAIN” frame window. In main frame window, there are two input fields, one is for entering the number of fire-brigade agents for the fire-brigade vehicle available in the environment and second is the number of fire-events for which simulation has to be done. First of all number of fire-brigade agents will be entered.

6.3 Fire-brigade Agent Frame

On clicking the “create” button, the fire-brigade agent will get created. Figure 3 shows the Fire-Brigade agent implementation and the fire-brigade frame. When the fire-brigade agent gets created, it will randomly generate the (x,y) coordinate for the location of fire-brigade vehicle, water-tank capacity and the speed of the fire-brigade vehicle. The fire-brigade agent frame is shown in figure 18.

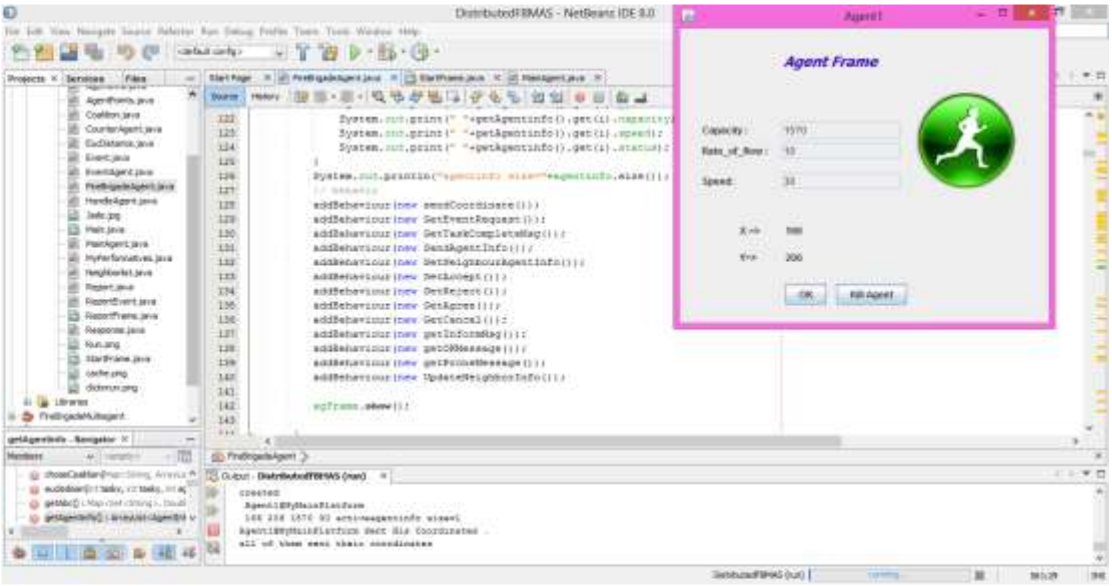


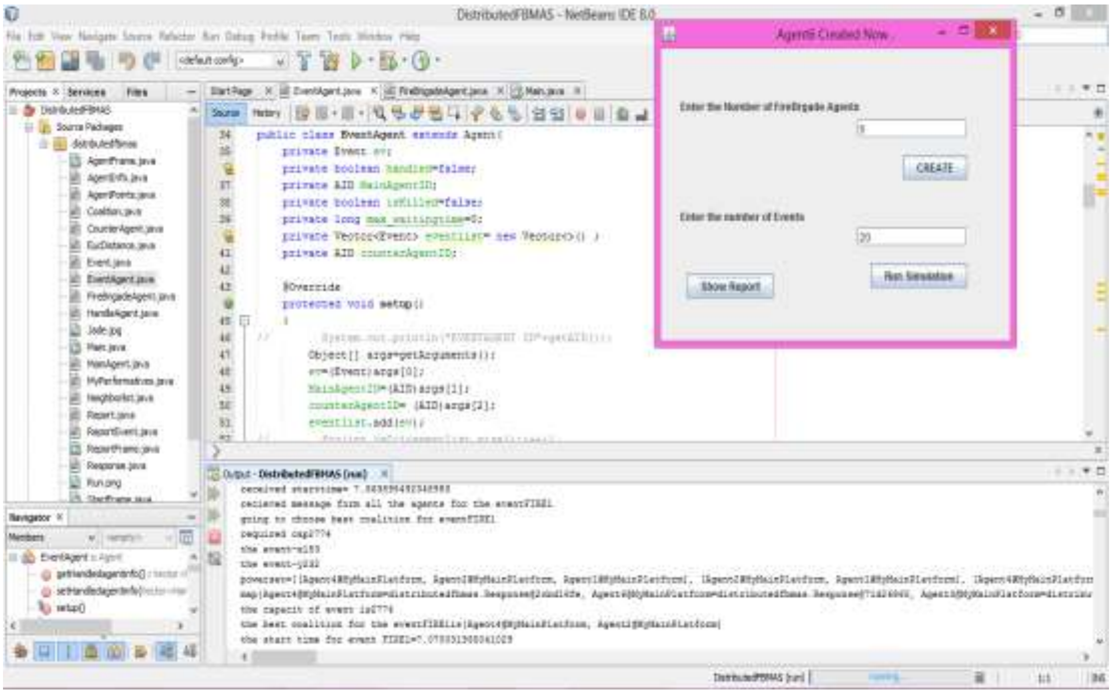
Figure 18- snapshot for fire-brigade agent

Here, “Kill Agent” button is used to terminate the respective fire-brigade agent. The behaviors for which fire-brigade agent is responsible are:

1. sendCordinate, sends the details of their coordinate to FSA
2. GetEventRequest, when FSA sends request to FBA for the fire-extinguishing.
3. SendAgentInfo, sends the bid to FSA
4. GetTaskCompleteMsg, gets the success report of the task for which they are assigned
5. GetAccept, if sender FBA/FSA receives the ACCEPT message. After receiving the ACCEPT message, the sender FBA goes for coalition formation.
6. GetReject, if sender FBA/FSA receives the REJECT message.
7. GetInformMsg, when agent receives the INFORM message which acknowledge the winning agent for the requested event.
8. GetConfirmMsg, when agent receives CONFIRM message, it will undergo for the Task Execution phase.

6.4 Task Allocation Processing

After initializing the fire-brigade agents, fire-fighting scenario is ready for the simulation. The number of events for which simulation is going to be done is entered into the respective text field and starts the simulation. The location in terms of (x,y) coordinate and its required water capacity is generated randomly and then FSA will start the task allocation for the requested event. The snapshot of fire-brigade frame is shown in figure 19.



Enter the Number of FireBrigade Agents: 5

Enter the number of Events: 20

```
public class EventAgent extends Agent{
    private Event ev;
    private boolean handled=false;
    private AID MainAgentID;
    private boolean isKilled=false;
    private long max_waitingtime=0;
    private Vector<Event> eventlist= new Vector<>( );
    private AID counterAgentID;

    @Override
    protected void setup()
    {
//          System.out.println("EVENTAGENT ID="+getAID());
        Object[] args=getArguments();
        ev=(Event)args[0];
        MainAgentID=(AID)args[1];
        counterAgentID= (AID)args[2];
        eventlist.add(ev);
```

Output - DistributedFBMAS (run)

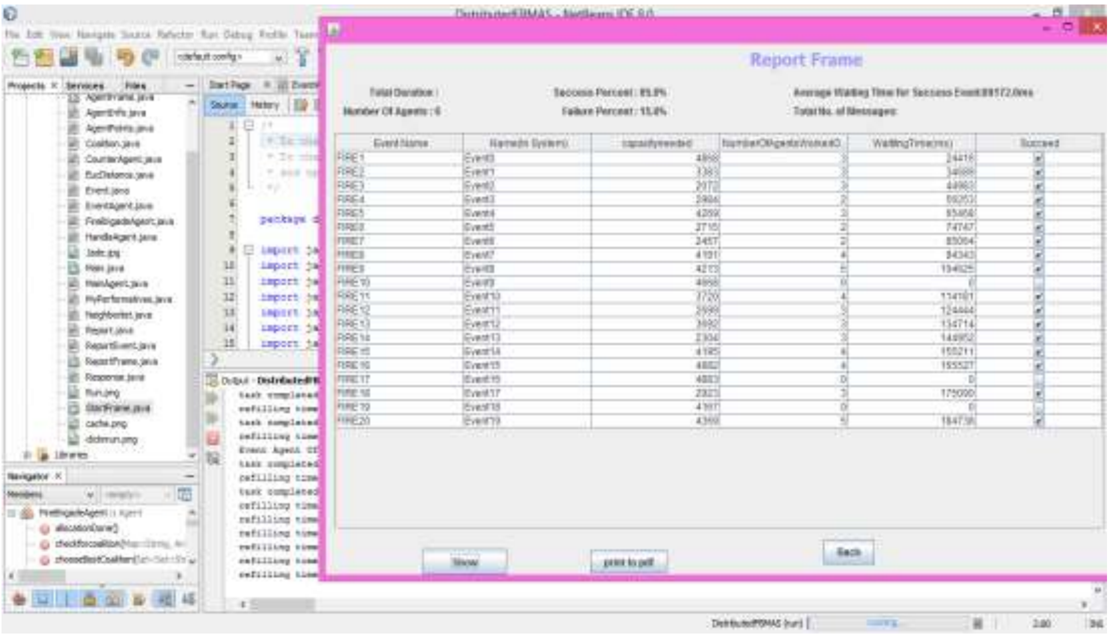
```
received starttime= 7.843595482348988
recieved message form all the agents for the eventFIRE1
going to choose best coalition for eventFIRE1
required cap2774
the event-x153
the event-y232
the capacit of event is2774
the best coalition for the eventFIRE1is[Agent4@MyMainPlatform, Agent1@MyMainPlatform]
the start time for event FIRE1=7.07083130806102
```

Figure 19- snapshot when task allocation is going on

After the successful task allocation, the FSA informs the respective FEA about the assigned agents and their expected start time. If the assigned agents reached at the event location then the event will be reported as success otherwise it is reported as failure of the event. When the event gets completed, the assigned FBA informed about completion and it will undergo for refilling.

6.5 Report Frame

When the simulation gets completed, the report will be generated after clicking on the “Show Report” button. The report will be displayed for each fire-event. The report will include the fire-event name, its require capacity, the number of agents assigned for that event completion, the waiting time for the event and the success or failure of the event. The average waiting time and the success % is calculated on the basis of waiting time and the success or failure of each and every fire-event. Figure 20 shows snapshot of the “Report Frame”.



Report Frame

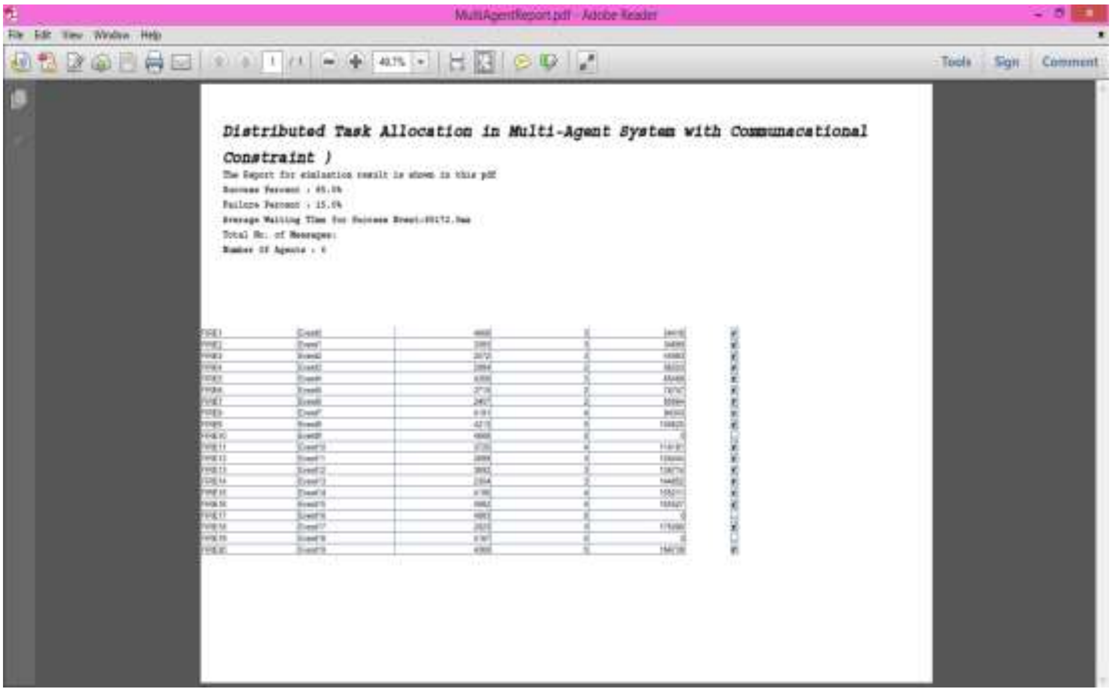
Total Duration :
Number Of Agents : 6
Success Percent : 85.0%
Failure Percent : 15.0%
Average Waiting Time for Success Event:89172.0ms
Total No. of Messages:

Event Name	Name(in System)	capacityneeded	NumberOfAgentsWorked	WaitingTime(ms)	Succeed
FIRE1	Event0	4868	3	24416	☑
FIRE2	Event1	3383	3	34686	☑
FIRE3	Event2	2072	3	44983	☑
FIRE4	Event3	2904	2	59253	☑
FIRE5	Event4	4269	3	65456	☑
FIRE6	Event5	2715	2	74747	☑
FIRE7	Event6	2457	2	85064	☑
FIRE8	Event7	4191	4	94343	☑
FIRE9	Event8	4213	5	104625	☑
FIRE10	Event9	4898	0	0	☐
FIRE11	Event10	3720	4	114181	☑
FIRE12	Event11	2598	5	124444	☑
FIRE13	Event12	3692	3	134714	☑
FIRE14	Event13	2304	3	144952	☑
FIRE15	Event14	4185	4	155211	☑
FIRE16	Event15	4882	4	155527	☑
FIRE17	Event16	4883	0	0	☐
FIRE18	Event17	2323	3	175090	☑
FIRE19	Event18	4197	0	0	☐
FIRE20	Event19	4368	5	184736	☑

Figure 20- snapshot of the Report Frame after the simulation has been done

6.6 Event-Report.pdf Generation

The report is saved as the “Report.pdf” file in the computer system so as to analyze the results for making the decision on the number of fire-brigade to be included in the system so as to increase the success rate of event. Figure 21 shows the snapshot of the pdf generated after clicking on the button “print to pdf” on the “Report Frame”.



Distributed Task Allocation in Multi-Agent System with Communacational Constraint)

The Report for simlation result is shown in this pdf
Success Percent : 85.0%
Failure Percent : 15.0%
Average Waiting Time for Success Event:89172.0ms
Total No. of Messages:
Number Of Agents : 6

Figure 21- Event Report pdf snapshot

CHAPTER 7
OBSERVATIONS AND RESULTS

This chapter shows the simulation results and observation drawn for the proposed approach.

7.1 Simulation Results

For simulation, a simple simulator for fire-fighting multi-agent system is developed in JADE framework. The simulator consist three types of agents namely, fire-station agent, which handles the team of fire-brigade agents; fire-brigade agent, which is responsible for the fire-brigade vehicle in the system and fire-event agent, which is related to the fire-event and handles the success or failure of fire-event. Fire-brigade can move freely i.e. they go straight to the target and do not follow the roads. Fire-brigades have a limited amount of water they can carry. According to the strength of the fire, the fire-event may require more water which is fulfilled by more than one fire-brigade.

The environment is taken as a plane and the location of fire-brigade and fire-event is taken by the (x,y) coordinate of the plane. Initially, the system will create a fire-station agent and n fire-brigade agents. The number of fire-brigade agents is determined by the number of fire-brigade present in the system. As mentioned earlier that fire-station agent possesses the global view of the system and fire-brigade agent possesses the local view of the system. As soon as the fire-brigade agent FBA_i gets created, it sends its location and capacity to the FSA. The obstacles are inserted at the initialization phase by randomly inserting the coordinates representing the location of obstacles.

The simulation is done three times with different number of agents and different event details and corresponding waiting time and number of messages transferred for the task allocation is calculated.

For simulating the proposed approach, the experiment is done on 100 numbers of events. The distribution of arrival rate of an event is taken as a Poisson distribution. The experiment is done for 100 event request and the result is observed after processing of every 10 event request. The details for the fire-brigades and the fire-event location and required capacity are generated randomly.

Table 5 shows the simulation result for fire-fighting multi-agent system with 3 fire-brigade agents. Task allocation without communicational constraint and task allocation with communication constraint are simulated for the same dataset. The location of fire-brigade vehicle is taken as (x,y) coordinate of the plane. The capacity and speed of the fire-brigade agents are taken as inputs which are generated at random. The event request is generated at random with Poisson distribution which takes event's location and required capacity as input. The average waiting time and number of messages transferred are observed after the processing of every 10 fire-events.

Table 5- Simulation results for the experiment 1

No. of events	No. of message		Avg. waiting time of successful events (in seconds)		Success %	
	A	B	A	B	A	B
0-10	130	95	18.52	16.67	30.0	40.0
10-20	266	175	18.48	13.47	35.0	40.0
20-30	402	277	18.61	17.16	36.0	43.34
30-40	506	383	18.10	17.42	32.0	45.0
40-50	636	460	18.04	16.59	32.0	40.0
50-60	751	527	17.99	15.56	31.0	38.36
60-70	903	619	18.22	15.89	34.0	38.57
70-80	1061	693	18.54	16.35	37.49	40.0
80-90	1208	784	18.71	16.80	38.88	41.11
90-100	1333	844	18.64	15.78	37.99	39.0

No. of agents = 3

Agent capacity= (1000-2000) and event capacity= (1000-5000)

A, task allocation without communicational constraint

B, task allocation with communicational constraint

Table 6 shows the simulation result for 6 fire-brigade agents. The simulation is again done for 100 numbers of events and average waiting time and number of messages transferred are observed after the processing of every 10 events. The success % represents the percentage of number of events successfully accomplished by the assigned fire-brigades.

Table 6- Simulation results for the experiment 2

No. of events	No. of message		Avg. waiting time of successful events (in seconds)		Success %	
	A	B	A	B	A	B
0-10	251	167	18.89	11.89	50.0	60.0
10-20	502	325	18.95	11.72	50.0	60.0
20-30	701	476	18.26	10.23	43.33	56.0
30-40	983	611	18.78	9.51	47.49	55.0
40-50	1268	794	19.77	11.18	46.0	58.0
50-60	1580	983	19.92	11.40	46.66	58.33
60-70	1960	1187	20.54	12.38	51.42	60.0
70-80	2253	1324	20.62	11.79	52.0	58.75
80-90	2522	1525	20.81	15.08	54.44	63.0
90-100	2769	1690	20.76	14.82	54.0	63.0

No. of agents = 6

Agent capacity= (1000-2000) and event capacity= (2000-8000)

A, task allocation without communicational constraint

B, task allocation with communicational constraint

Table 7 shows the simulation result for 10 fire-brigade agents. Here, the simulation is again done for 100 numbers of events and result is observed after the processing of every 10 events.

Table 7- Simulation results for the experiment 3

No. of events	No. of message		Avg. waiting time of successful events (in seconds)		Success %	
	A	B	A	B	A	B
0-10	490	279	35.15	33.19	70.0	90.0
10-20	920	606	35.16	32.47	70.0	90.0

20-30	1370	936	34.99	32.79	70.0	90.0
30-40	1789	1159	29.63	26.51	67.0	70.0
40-50	2364	1501	29.85	26.14	68.0	70.0
50-60	2759	1817	28.85	27.18	68.33	81.67
60-70	3249	2104	28.93	26.79	68.57	81.43
70-80	3764	2423	28.96	26.58	68.75	81.25
80-90	4289	2810	30.96	28.27	68.89	83.33
90-100	4749	3113	30.93	28.69	69.0	84.0

No. of agents = 10

Agent capacity= (1000-2000) and event capacity= (2000-10000)

A, task allocation without communicational constraint

B, task allocation with communicational constraint

7.2 Observations

From the results obtained in simulation of three experiments, following observations has been drawn:

- The number of messages gets reduced when communicational constraint is applied in the task allocation algorithm. This results in less communication cost because in multi-agent system, the communication is done by message passing only.
- The average waiting time is also reduced. The time is very crucial parameter for any real-time systems. Especially for fire-fighting scenario. Because if the response time is very high then it may results in severe damage to people and infrastructure. It happens because of the inclusion of EST and trust factor for choosing best coalition.
- The success rate of the events is dependent on the number of fire-brigade agents in the system. As observed in experiment-1 where there are only 3 agents in the system the maximum success rate is 45% whereas in case of 3rd experiment the success rate is 90% with 10 fire-brigade agents in the system. Thus by analyzing the success rate of the incoming event-request, we can predict the optimal number of fire-brigades that must be present in the system so as to increase the success rate.

The results obtained from the simulation are visualized in fig.22 to fig. 29.

Fig. 22 shows the graph representing the number of messages transferred in proposed approach for simulation experiment 1 with and without communicational constraint respectively. This graph shows that the number of message transferred gets reduced in the proposed approach. This happened because instead of sending the message to all neighbor FBAs, proposed approach sends the message to the nearest FBA to the event location.

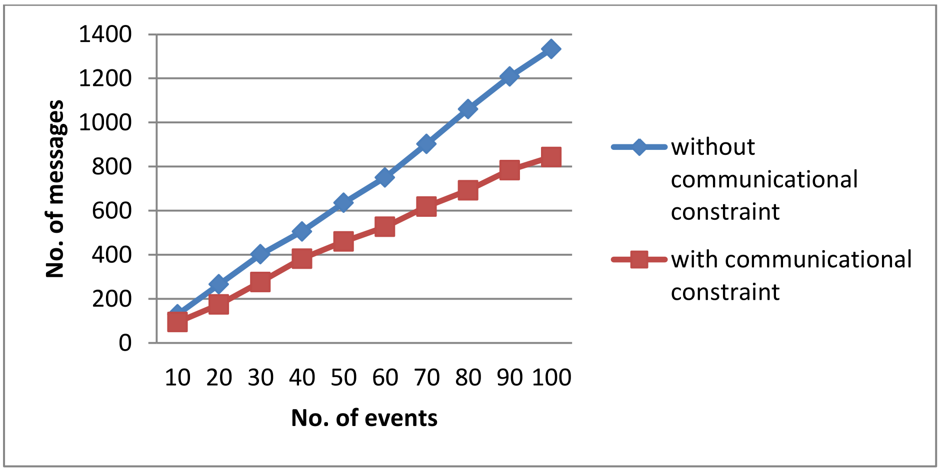


Figure 22- number of message transferred vs. total no. of events for experiment 1

Fig. 23 shows the graph representing average waiting time for successful events after the processing of every 10 events for task allocation for simulation experiment 1 with and without communicational constraint respectively. It shows that proposed approach gives better results.

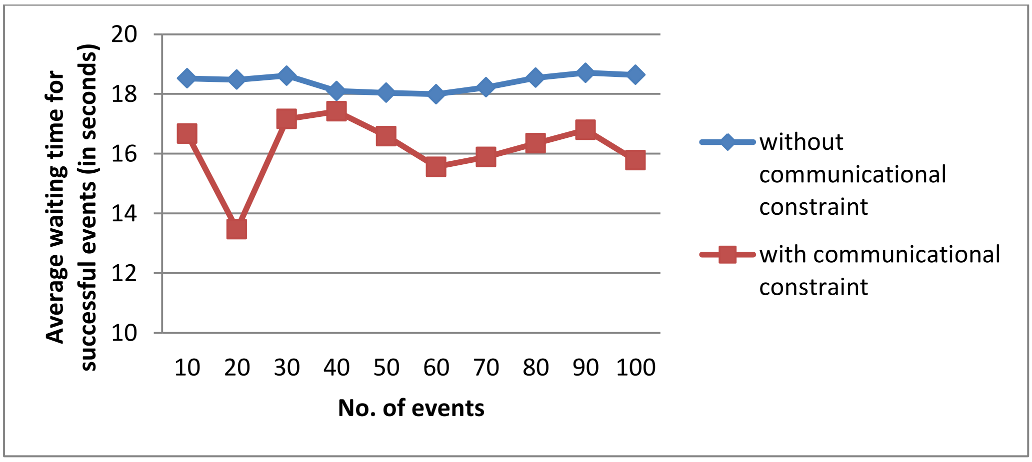


Figure 23- avg. waiting time for successful events vs. total no of events for exp. 1

Fig. 24 and 25 shows the simulation result of experiment 2 for both the parameters i.e. number of messages transferred and avg. waiting time. This shows that the coordination mechanism with communicational constraint outperforms the coordination without communicational constraint.

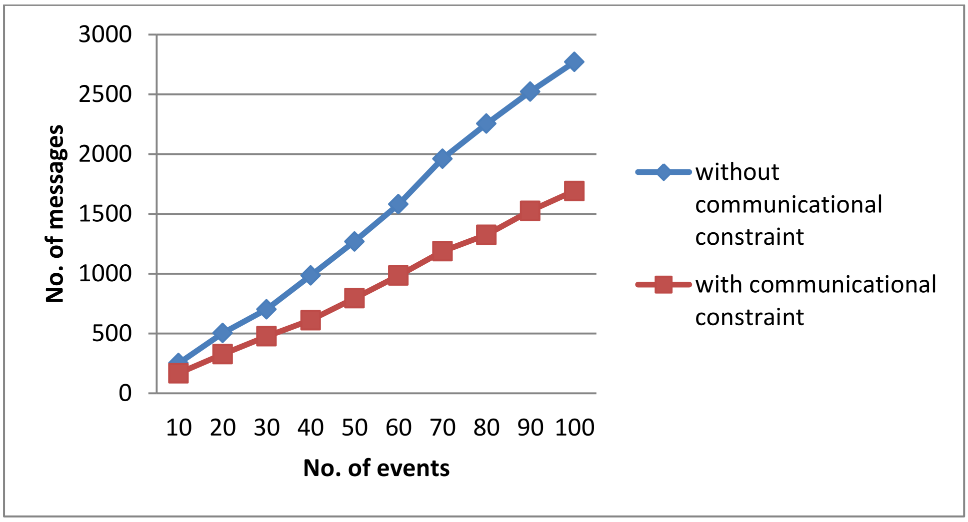


Figure 24- number of message transferred vs. total no. of events for experiment 2

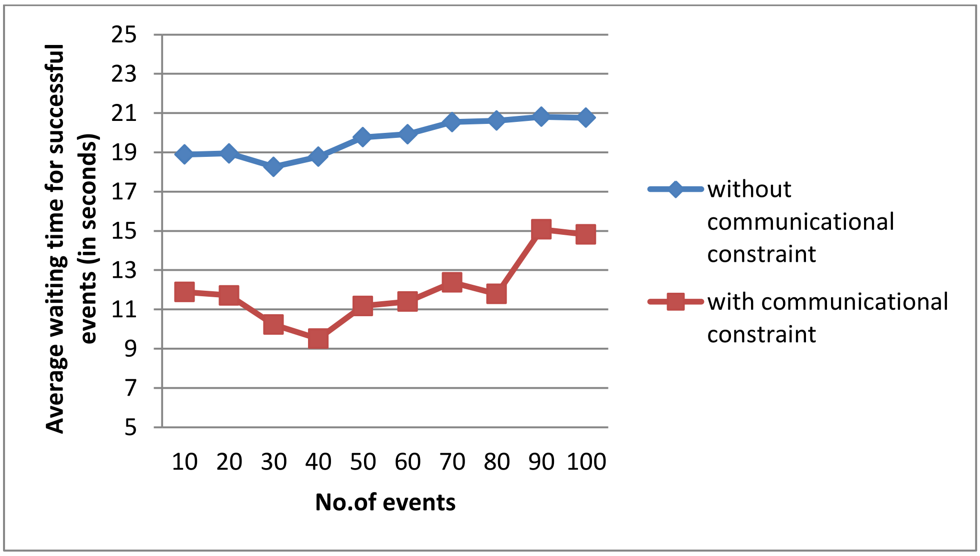


Figure 25-avg. waiting time for successful events vs. total no of events for exp. 2

Fig. 26 and 27 visualizes the simulation result of experiment 3 for both the parameters i.e. number of messages transferred and avg. waiting time.

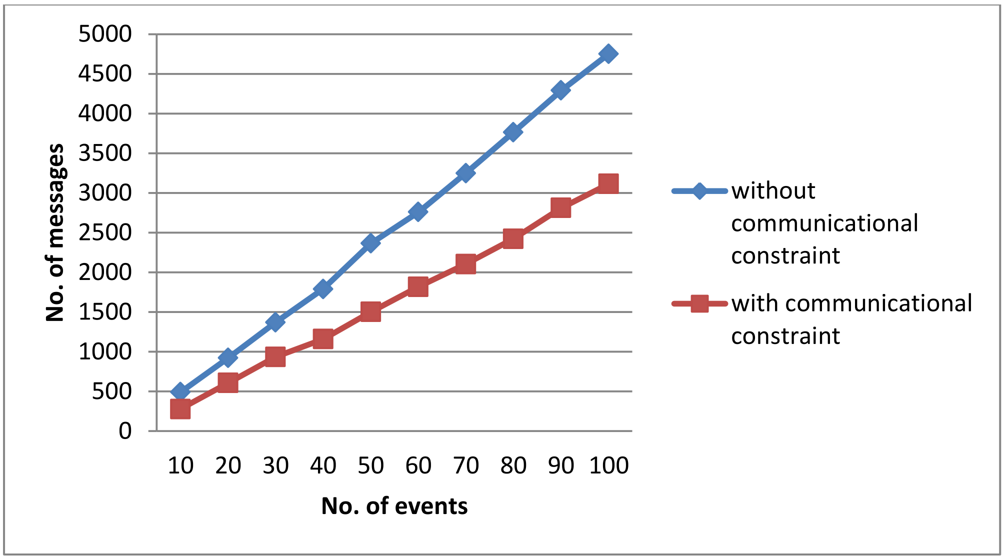


Figure 26- number of message transferred vs. total no. of events for experiment 3

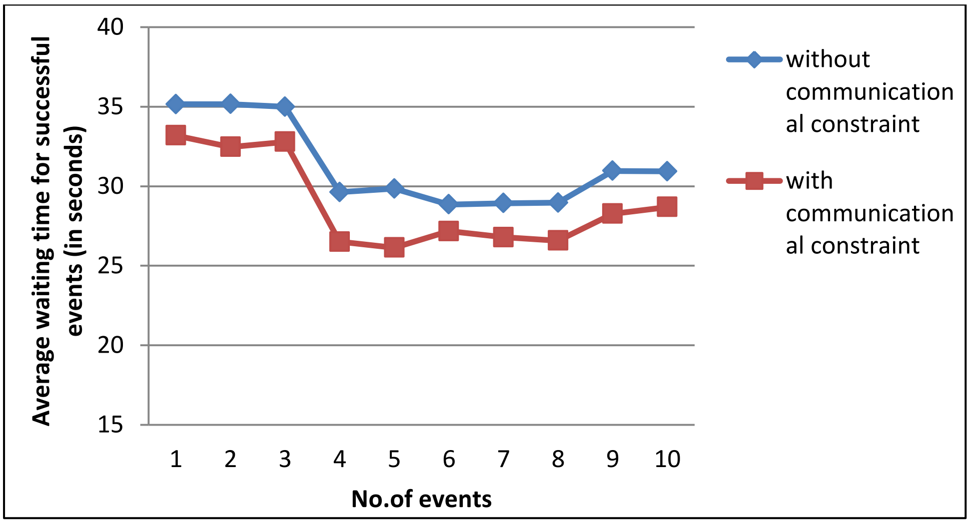


Figure 27- avg. waiting time for successful events vs. total no of events for experiment 3

All above graphs show that in all three simulations, where the numbers of agents are 3, 6 or 10, the proposed approach outperforms the existing approach for both the parameters i.e. number of messages transferred and waiting time.

Fig. 28 shows the success rate of proposed approach for all of the three experiments. The success% is also observed after the processing of every 10 events out of total 100 events.

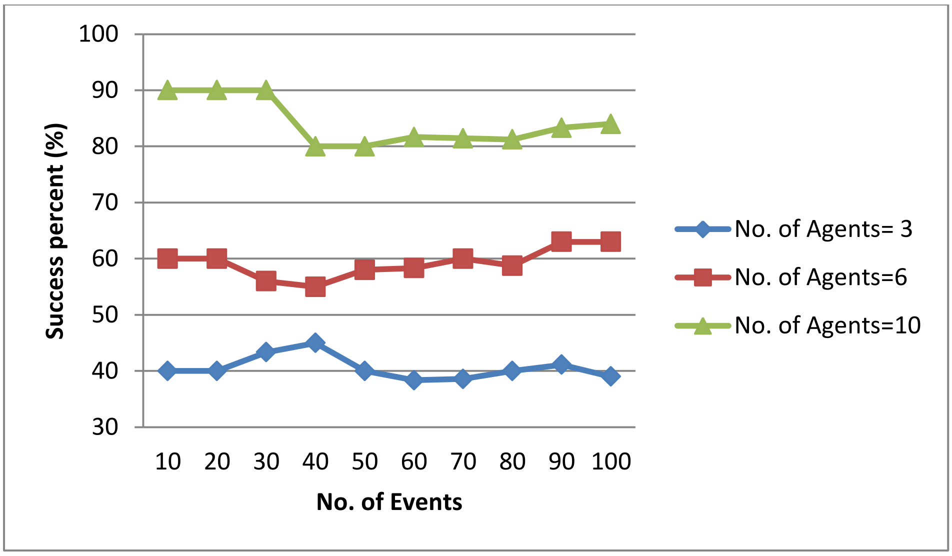


Figure 28- success rate of proposed approach for all the experiments 1, 2, and 3

It shows that the success rate increases with the increase of number of agents in the system. This is helpful for getting the optimal number of fire-brigades in the system on the basis of past history of success % of the fire-events.

From the fig. 28, it is observed that with 10 fire-brigade agents, the success % is approximately 82%. The success% depends on the coalition formation and re-planning algorithm of the proposed approach. As the coalition is formed on the basis of expected earliest start time and the trust factor so the chance of failure of task allocation will get reduced. The re-planning algorithm also results in increase in success %. Whenever any obstacle detected, then instead of sending the failure message the agent will perform the resource negotiation and if no response is received then only it will send the failure message.

CHAPTER 8

CONTRIBUTIONS

This research work proposes the task allocation approach to allocate the appropriate fire-brigade at the fire-event location in fire-fighting multi-agent system. In today's era researchers' main focus is to make the universe autonomous. With the same aim, I've chosen the multi-agent environment in which agents can autonomously think without any user intervention. Thus in this research work, the task allocation is done autonomously. The task arrives at the agent and agent will try to accomplish it successfully. To accomplish the complex tasks, which are not completed by an individual agent, group of agents are formed for its completion. The agents communicate and coordinate with each other to form the group of agents. Thus, the main research issue in task allocation problem is coordination and coalition formation. The task allocation will become more difficult when the environment is highly dynamic and uncertain like disaster scenario. To allocate the task in such type of environment some constraints need to be applied.

In this thesis, we devised an optimized task allocation approach by improving the coordination and coalition formation mechanism. This results in better task allocation with less waiting time and less communication cost. We also proposed a re-planning algorithm to handle the difficulties occurred due to dynamic nature of environment.

The proposed approach is divided into four stages namely, Task Arrival, Resource Negotiation, Coalition Formation, Task Execution. Whenever any request for extinguishing the fire arrives, the fire-station agent will go for fire-brigade allocation. The proposed approach allocates the most appropriate fire-brigade or group of fire brigades which fulfill the requested capacity of fire-event on the basis of earliest expected start time and trust factor. It results in increase in success rate of the allocation process. The time and the communication cost are considered as the primary factors for the allocation procedure. Thus the proposed approach allocates the efficient fire-brigade with less amount of time and less communication cost.

8.1 Contribution to the Society

The proposed approach is allocating the appropriate fire-brigade to the fire-event’s location in less waiting time. This approach can be helpful for the various application domains in our society.

1. The state of Himachal Pradesh is a very sensitive area in terms of seismic point of view. This area suffers from the earthquake very often. Thus, there is a very high requirement to automate the disaster management. The Fires occurs due to the broken gas lines or electrical lines are one of the common side effects of earthquakes. The things got more complicated when water lines were also broken. The San Francisco earthquake of 1906 results in 90% of damage by fire. To recover such type of disaster, the proposed approach can be efficiently used. Whenever there is a request for extinguishing the fire, the fire-station agent will be invoked and it will allocate the appropriate fire-brigade by communicating with FBA, within less waiting time.
2. To extinguish the fire in residential area or industrial area, the proposed approach can also be used and will give efficient results.
3. The proposed approach can also be used to allocate the emergency medical services like ambulances to the patient’s location so as to provide a quicker treatment to the patient. With a slight modification in the proposed approach, it can be used to allocate the ambulance. In case of medical services, in place of water tank capacity, we can consider the first-aid facilities available in the ambulance. Thus, according to the requirement of the patient, the appropriate ambulance will be allocated for the patient. In this system, no coalition will be required thus the init-agent will be chosen on the basis of Expected earliest start time first and trust factor among the ambulances which fulfills the required first-aid facilities of the patient.

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

Task allocation problem in multi-agent system is hot research topic from the last few years. The task allocation problem can be defined as allocating the task to the agents so as to achieve the system goals without any conflict. The main problem issues with distributed task allocation are coordination among multiple agents and coalition formation. Many researchers are working on the task allocation issues in multi-agent systems which are highly dynamic and uncertain in nature.

The main focus of this thesis work is to optimize the task allocation approach in dynamic and real-time multi-agent system by improving thr coordination and coalition formation mechanism.. This research considers the fire-fighting scenario for the task allocation process. This scenario has been chosen because it is highly dynamic, uncertain and real time system. The fire-fighting multi-agent system consists of one fire station agent and n fire-brigade agent. The fire-station agent has the global view of the system like the occurrence of event in the system and knowledge of the fire-brigade agent available in the system. The fire-brigade agent has the local view of the system i.e. the details of its respective fire-brigade vehicle and the knowledge of its surroundings. The task allocation problem thus defined as, allocating the appropriate fire-brigade or group of fire-brigades so as to fulfill the required capacity of water with less waiting time and less number of messages transferred for coordination among multiple agents. The proposed approach is divided into four modules namely, Task Arrival, Resource Negotiation, Coalition Formation and Task Execution.

The proposed approach optimizes the existing task allocation approach by making following changes:

1. Improved coordination mechanism, applies the communicational constraint during coordination. Instead of sending the request to all the participating agent and chose the winning coalition, the proposed sends the request message to the nearest agent of the event location only. If it is capable to fulfill the required capacity alone then it will send the OK message and assigned for that event. But in CNP, whether the nearest agent is capable to accomplish the

event, the request message sends to all participating agent. Though the winning agent will be same in both the cases but the number of message transferred increases as compared to the proposed approach.

2. Improvement in coalition formation, the best coalition will be chosen on the basis of earliest expected start time, trust factor and smallest coalition size. This is helpful to increase the success rate of task allocation approach.
3. Use the re-planning algorithm to handle the obstacles detected while travelling toward the event-location.

The proposed approach is simulated with different number of fire-brigade agent and following conclusion has been drawn:

- The proposed approach results in less waiting time
- The communication cost in terms of number of messages transferred for the coordination and cooperation is also reduced.
- The success rate of event is also depends on the number of fire-brigade agent available in the system.

The proposed approach thus results in an efficient task allocation process for dynamic and uncertain environment with spatial, temporal and communicational constraint. Though the proposed approach is very helpful for task allocation in dynamic environment but it still possesses several limitations:

- The proposed approach considers the cooperative nature of agent but agent can also possess selfish nature.
- The reliable communication channel is considered but in actual it is not possible to have reliable communication channel.
- It only considers a single route from the agent’s location to event’s location.
- This proposed work is limited to only single region having one fire-station agent. But it can be extended to the multiple regions where one fire-station of one region can negotiate with the fire-station of another region in case of failure.

Thus to overcome these limitation some work will be required in future. In future, I will also try to enhance this approach for rescuing the victims after extinguishing the fire. I’ll also integrate the Google map to simulate the approach on real data.

REFERENCES

[1] Nick Jennings, Michael Wooldridge, “Software Agents”, IEEE Review, January 1996, pp. 17-20

[2] Michael Wooldridge, “Introduction to Multi-agent Systems”, England: John Wiley & Sons Ltd., Aug. 2002.

[3] G. Weiss, “Multiagent Systems: A Modern Approach to Distributed Artificial intelligence” London: The MIT Press, 1999.

[4] L. Padgam, M. Winikoff, “Developing Intelligent Agent Systems”, England: John Wiley & Sons, 2004

[5] Kathryn S. Macarthur, “Multi-Agent Coordination for Dynamic Task Allocation”, PhD [Dissertation], Southampton: University of Southampton, Dec. 2011.

[6] Zheng X, and Koenig S ,”Reaction functions for task allocation to cooperative agents”, In Proceedings of the 2008 7th International conference on Autonomous Agents and Multi Agent Systems, AAMAS 2008, May, 12-16., 2008, Estoril, Portuga, pp. 559–566.

[7] Shehory O and Kraus S, “Methods for task allocation via agent coalition formation” Artificial Intelligence, vol. 101, pp. 165–200, May 1998.

[8] A.C. Chapman, R. A. Micillo, Ramachandra Kota and N.R. Jennings, “Decentralized dynamic Task Allocation: A Practical Game- Theoretic Approach,” In Proceedings of 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009,May 10-15, 2009, Budapest, Hungary, pp. 915-922

[9] S. Shaheen Fatime, Michael Wooldridge, “Adaptive task resource allocation in multi-agent systems”, In Proceedings of 5th International Conference on Autonomous Agents, AGENTS’ 01, ACM, New York, NY, USA, 2001, pp.537-544

[10] yan kong, Minjie Zhang, Dayong Ye, “A Group Task Allocation Strategy in Open and Dynamic Grid Environments”, presented at 7th International workshop on Agent based Complex Automated Negotiations, ACAN 2014, Paris, France, May 5-6, 2014.

[11] Scerri P., Farinelli A., Okamoto S. and Tambe, “Allocating tasks in extreme teams”, In Proceeding of 5th International Conference on Autonomous Agents and Multi Agent System, AAMAS 2005,Utrecht, Netherlands, 2005, pp. 727–734.

[12] Petcu, A., and Faltings, B. 2005. S-DPOP: Super stabilizing Fault containing Multiagent Combinatorial Optimization”, In Proceedings of the National Conference on Artificial Intelligence, AAAI-05,Pittsburgh, Pennsylvania, USA, July 2005, pp. 449–454.

[13] K.S. Macarthur, R. Stranders, S.D. Ramchurn and N.R. Jennings, “A Distributed Anytime Algorithm for dynamic Multi-Agent Systems”, In proceeding of 25th National Conference on Artificial Intelligence, AAAI-2011, Aug. 7-11, 2011, San Francisco, pp. 701-706.

[14] S.D. Ramchurn, M. polukarov, Alessandro Farinelli, Cuong Troung, “Coalition formation with spatial and temporal constraint”, In Proceeding of 9th International Conference on Autonomous Agents and Multi-agent System, AA-MAS 2010, May 10-14, 2010, Toronto, Canada, pp-1181-1188

[15] Jipeng Du, Ling Zhou, Peng Qu, Zhen Shi, Yang Lin, “Task Allocation in Multi-Agent Systems with Swarm Intelligence of Social Insects”, In proceedings of 6th International Conference on Natural Computation, ICNC-2010, Aug. 10-12, 2010, Yantai, China, pp-4322-4326.

[16] Philip H.P.Nguyen, Minh-Quang Nguyen and Ken Kaneiwa, “A Belief-Based Multi-Agent Markov Decision Process for Staff Management”, In Proceedings of International Journal of Energy, Information and Communication, Vol. 2, Issue 2, pp.23-39, May 2011

[17] George Thomas, Andrew B. Williams, “Sequential Auction for Heterogeneous Task Allocation in Multiagent Routing Domains”, In proceedings of the 2009 IEEE Conference on Systems, Man and Cybernetics, SMC-2009, Oct. 11-14, 2009, San Antonio, TX, USA, pp. 1995-2000.

[18] Wanyuan Wang, Yichuan Jiang, “Community-Aware Task Allocation for Social Networked Multiagent Systems”, IEEE Transactions on Cybernetics, vol. 44, No. 9, pp. 1529-1543, September 2014.

[19] Danny Weyns, Nelis Bouck´e, Kurt Schelfthout, Tom Holvoet , “Dyncnet: A Protocol For Flexible Task Assignment Applied in An AGV Transportation System”, In Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2007, Boston, MA, USA, July 9-11, 2007, pp. 1-4.

[20] S.D. Ramchurn, A. Farinelli, K.S. Macarthur, N.R. Jennings, “Decentralized coordination in RoboCup rescue", Published in The Computer Journal, Vol. 53, pp. 1-15, 2010.

[21] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, “ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees “, Artificial Intelligence, vol. 161, pp. 149-180, Sept. 2004.

[22] James Odell, “Agent Technology: What is it and why do we care? “, Enterprise architecture Advisory Services, Executive Report, vol. 10, No. 3, 2007. [Online]. Available: http://www.jamesodell.com/Cutter-Exec_Rpt-July_2007.pdf.

[23] Prithviraj Dasgupta, “A Multi-agent swarming System for Distributed Automatic Target Recognition Using Unmanned Aerial Vehicles”, IEEE Transaction on Systems, Man and Cybernetics, vol. 38, No. 3 , pp. 549-563, May, 2008.

[24] Sofia Amador, Steven Okamoto an Roie Zivan, “Dynamic Multi-agent Task Allocation with Spatial and Temporal Constraints”, In proceeding 13th International Conference on Autonomous Agents and Mutli Agent System, AAMAS-2014, May 5- 9,2014,paris, France, pp. 1495-1496

[25] Yongcan Cao, Wenwu Yu, Wei Ren and Guanrong, “An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination”, IEEE Transaction on Industrial Informatics, vol. 9, No. 1, pp. 427-438, Feb. 2013

[26] Frank R. Kshischang, Brendan J. Frey and Hans-Andrea Loeliger, “Factor Graphs and the Sum-Product Algorithm”, IEEE Transactions on Information Theory, Vol. 47, No. 2, pp. 498-519, Feb. 2001

[27] Zheng Xiao, Shengxiang Ma, Shiyong Zhang, “Learning Task Allocation for Multiple Flows in Multi-agent Systems”, In proceedings of International Conference on Communication Software and Networks, ICSSN’09, Feb. 27-28, 2009, Macau, pp. 153-157.

[28] Xing Su, Minjie Zhang, Dayong Ye, “A Dynamic Approach for Task Allocation in Disaster Environments under Spatial and Communicational Constraints”, AAAI workshop on Multiagent Interaction without Prior Coordination, MIPC 2014,July 2014.

[29] Yichuan Jiang, Zhaofeng Li, “Locality-sensitive Task Allocation and load balancing in networked multi-agent systems: Talent versus centrality”, Journal of Parallel Distributed Computing, vol. 71, no. 6, pp. 822-836, June 2011.

[30] Hiroki Kitano and Satoshi Tadokoro, “Robocup Rescue: A Grand Challenge for Multiagent and Intelligent Systems”, AI Magzine, SPRING 2001, Vol. 22, No. 1, pp. 39-52, November 1, 2001.

[31] S.A. Suárez Barón, “Dynamic Task Allocation and Coordination in Cooperative Multi-Agent Environments”, PhD [Dissertation], Girona: University of Girona,2010.

[32] Foundation for Intelligent Physical Agents Specification 1997. Available from http://www.fipa.org

[33] Mohammad Ubaidullah Bokhari, Sadaf Ahmad, Faheem Syeed Masoodi, “Development of Multi-Agent System Based on Frameworks, “In the proceedings of International Conference on Reliability, Infocom Technology and Optimization, ICRITO’2010, Nov. 4-6, 2010, Faridabad, India.

[34] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa, “JADE- A FIPA- compliant agent framework”, In proceedings of PAAM’99, April 1999, vol. 99, pp. 97-108 .

[35] F. Bellifemine, G. Caire, D. Greenwood, “Developing multi-agent systems with JADE”, England: John Willey & Sons Ltd., 2007.

[36] Rashmi Singh, Aarti Singh, Saurabh Mukherjee, “A Critical Investigation of Agent Protocols in Multiagent Systems”, International Journal of Advancements in Technology, IJoAT- 2014, vol. 5, no. 2, pp. 72-81 , March 2014.

[37] Yeung, W.L., “Short paper adapting the contract net protocol for publish/subscribe messaging”, In the proceedings of International conference on high performance computing and simulation, HPCS-2013, July 1-5, 2013, Helsinki, pp. 139-142.

[38] A. Farinelli, G. Grisetti, L. Iocchi, S. Lo Cascio and D. Nardi, “Design and Evaluation of Multi Agent Systems for Rescue Operations”, in the proceedings of 2003 IEEE/RSJ International conference on Intelligent Robots and Systems, October 2003, Las-Vegas, Nevada, pp. 3138-3143

[39] Abeer el-korancy and Khalid el-bahnasy, “A multi-agent Cooperative model for Crisis Management System”, In the proceedings of 7th International conference on artificial Intelligence, Knowledge engineering and Databases, AIKED’08, Feb. 20-22,2008, University of Cambridge, U.K. , pp. 170-175

[40] Yunbo Lu, Xu Yang and Xuebin Cui, “An Agent-Based Simulation Application on a Multi-Agent Firefighting and Rescue System”, in the proceedings of 8th international conference on Supply Chain Management and Information Systems, SCMIS-2010, Oct. 6-9, 2010, Hong Kong, pp. 1-5.

[41] Omid Amir Ghiasvand and Maziar Ahmad Sharbafi, “Using Earliest Deadline First Algorithms for Coalition Formation in Dynamic Time-Critical Environment”, International Journal of Information and Educational Technology, Vol. 1, no. 2, pp. 120-125, June 2011.

[42] Beatriz López, Bianca Innocenti and Dídac Busquets, “A Multiagent System for Coordinating Ambulances for Emergency Medical Services”, IEEE Intelligent Systems, vol. 23, no. 5, pp. 50-57, 2008.

APPENDIX- A

PUBLICATIONS

[1] Vaishnavi Singhal and Deepak Dahiya, "Distributed Task Allocation in Dynamic Multi-Agent Systems", accepted In the Proceedings of International Conference on Computing, Communication and Automation, ICCCA 2015, May 15-16, 2015, School of Computing Science and Engineering, Galgotias University, Uttar Pradesh, India.

[2] Vaishnavi Singhal and Deepak Dahiya, "Task Allocation in Fire-Fighting Multi-Agent Systems". [**under submission**]

APPENDIX- B
PSEUDO-CODE OF PROPOSED ALGORITHM

This chapter presents the pseudo-code for the proposed algorithm.

B.1 Pseudo-code for the proposed approach

The proposed approach improves the coordination mechanism and the coalition formation algorithm so as to allocate the appropriate fire-brigade to the fore-event location in more effective and efficient way.

The pseudo-code for the proposed approach is explained below.

Initialization:

1. Create fire-station agent, initialize the obstacles at some location
2. Create the fire-brigade agent with the location, cap(FBA), speed(FBA_i)
3. Set status(FBA)= “active”
4. No_of_sucees =0, no_of_failure = 0
5. Agent_list(FSA).add(x(FBA),y(FBA), no_of_success(FBA), no_of_fail(FBA)

Algorithm 1: Task Arrival

Input: <FE> = {ev_name, x(ev), y(ev), req_cap(ev), arrivaltime(ev)}

Output: wait_time, msg_count

Algorithm:

```
For all $FBA_i$ Є agent_list(FSA)
  Init-agent = min(Eucledean_dist(x($FBA_i$), y($FBA_i$), x(ev), y(ev)))
End for loop
send event_req (x(ev), y(ev), req_cap(ev)) from FSA to init-agent
increase msg_count by 1
if(Receive (event_req(x(ev), y(ev), req_cap(ev))) by init_agent
        If(status(init-agent) = “active”)
                if(cap(init-agent) > = req_cap(ev))
                  EST = currenttime + (eucledean_dist(x(init_agent), y(init_agent),
                     x(ev), y(ev)) / speed(init_agent))
```

```
            Send OK(EST(init_agent)) to FSA
            task_execution(init_agent, EST)
            increase the msg_count by 1
        Else resource_negotiation(ev,x(ev), y(ev), req_cap(ev)-cap(init_agent))
    Else resource_negotiation(ev, x(ev), y(ev), req_cap(ev))
End if
If( receive_OK(EST(<assigned_agentlist, EST)) from init_agent)
    Send Allocationdone(<assigned_agentlist>, EST) to event_agent
    increase the msg_count by 1
Else report failure
If (receive Allocationdone(<assigned_agentlist, EST)
    while (currenttime != EST(init_agent))
    If reached(FBA)
        Reached_agentlist.add(FBA)
    End if
    end while
    if (reached_agentlist.equals(assigned_agentlist))
        completion_time= current_time
        Send success(ev, reached_agentlist, completion_time)
        increase the msg_count by 1
    else send failure(ev)
If(receive success(ev, reached_agentlist, completion_time))
    For all $FBA_j$ Є reached_agentlist
        no_of_success +=1
    end for loop
    wait_time(ev) = completion_time – arrival_time(ev)
    Return wait_time and message count
Else report failure for event ev.
```

Algorithm 2: Resource Negotiation

Input: ev, x(ev), y(ev), req_cap(ev)

Algorithm:

```
For $FBA_i$ Є neighbor_list
    send event_req (ev, x(ev), y(ev), req_cap(ev)) to $FBA_i$
```

```
    increase msg_count by 1
end for loop
if (receive event_req(ev, x(ev),y(ev),req_cap(ev))) by $FBA_i$
    if(status($FBA_i$) = “active”)
        EST = currenttime + (eucledean_dist(x($FBA_i$), y($FBA_i$), x(ev), y(ev)) /
        speed($FBA_i$))
    Send ACCEPT(EST($FBA_i$), cap($FBA_i$))
    increase msg_count by 1
    Else send REJECT(ev) to sender and increase msg_count by 1
End if
If(receive ACCEPT(EST($FBA_i$), cap($FBA_i$))
    Add $FBA_i$ to S                    // S is the set of FBA who sent ACCEPT
<Best_coalition, EST> = Coalition_form(S, req_cap(ev))
If(! Best_coalition.isEmpty())
    For all $FBA_i$ Є Best_coalition
        Send INFORM(ev, EST) to $FBA_i$
        increase msg_count by 1
    end for loop
    If (receive INFORM(ev)) by $FBA_i$
        If(status($FBA_i$) = “active”)
            Send OK(ev) to sender and increase msg_count by 1
        Else send PRONE(ev) to sender and increase msg_count by 1
    End if
    If(receive OK(ev) from all $FBA_i$ Є Best_coalition)
        Send CONFIRM(ev) to $FBA_i$ and send OK(Best_coalition, EST) to
        FSA and increase msg_count by i+1
    Else send CANCEL(ev_name) to all $FBA_i$ Є Best_coalition
    If receiveCONFIRM(ev) by $FBA_i$ then go for task_exceution($FBA_i$, EST)
Else send CANCEL(ev) to FSA and increase msg_count by 1
```

Algorithm 3: Coalition Formation

Input: S, req_cap(ev)

Output: best_coalition, EST // set of winning agents

Algorithm:

```
X=powerset(S)
For all $x_i$ Є X
    For all $FBA_i$ Є $x_i$
        tot_cap($x_i$) +=cap($FBA_i$)
    End for loop
    If(tot_cap($x_i$) >= req_cap(ev))
        Add $x_i$ to C
End for loop
For all $x_i$ Є C
    For all $FBA_i$ Є $x_i$
        EST($x_i$)= max(EST($FBA_i$))
        coalition.put($x_i$ , EST)
    End for loop
End for loop
sort_coalition= sort(coalition) // sort coalition in ascending order of EST
for all $c_i$ Є sort_coalition
    if(EST($c_i$) <= min)
        min= EST($c_i$)
        min_est_coalition.add($c_i$) // chose coalition with minimum EST
    end if
end for loop
if(min_est_coalition.size() > 1)
    for all $c_i$ Є min_est_coalition
        if($c_I$ .size() <= min)
            min= $c_i$. size()
            min_size_coalition.add($c_i$) // chose coalition having FBAs
        end if
    end for loop
else return min_est_coalition, EST(min_est_coalition)
if(min_size_coalition.size() > 1)
    for all $c_i$ Є min_size_coalition
        for all $FBA_i$ Є $c_i$
            tot_trust($c_i$) += trust($FBA_i$)     // trust model
        end for loop
```

```
        if( t <= tot_trust($c_i$))
            t= tot_trust($c_i$)
            max_trust_coalition.add($c_i$)
        end if
    end for loop
    return max_trust_coalition($c_0$), EST($c_0$)
else return min_size_coalition, EST(min_size_coalition)
```

Algorithm 4: Task Execution

Input: FBA, EST

Algorithm:

```
If (receive CONFIRM(ev))
    Set status(FBA) = “busy”
    Start moving toward event location
    if(! reached(FBA))
        if( obstacle_detected)
            replanning(ev, x(ev), y(ev), cap(FBA), EST)
        else continue
    end if
    If (reached(FBA))
        reached_FB.add(FBA)
    end if
    wait until (currenttime != EST)
    if (assigned_agent(ev) = reached_FB) //all the assigned brigades reached
        if(curr_req__cap(ev) <= tot_cap(reached_FB))
            Send SUCCESS(ev, reached_FB)
        Else send failure(ev)
        if(fire_exitngushed)
            for all $FBA_i$ Є reached_FB
                set status($FBA_i$) = “inactive” go for refilling
                if refilling is over
                    reached to its base location
                    set status($FBA_i$) = “active”
                end if
```

```
                end for loop
            end if
        Else
            Send FAILURE(ev)
end if
```

Algorithm 5: Trust

Input: FBA

Output: trust_fac(FBA)

```
Set $\alpha$ = no_of_success(FBA)
Set $\beta$ = no_failure(FBA)
Set trust_fac(FBA) = $\alpha/\alpha + \beta$
Return reust_fac(FBA)
```

Algorithm 6: Replanning

Input : ev, x(ev), y(ev), cap, EST

Algorithm:

```
For all $FBA_i$ Є neighborlist
    Send event_req(ev, x(ev), y(ev)) and increase msg_count by 1
End for loop
If(receive event_req(ev, x(ev), y(ev))) by FBA
    If(status(FBA) = “active” )
        EST= current_time + Euclidean_dist(x(FBA), y(FBA), x(ev), y(ev))
        /speed(FBA)
        Send ACCEPT(EST(FBA), cap(FBA)) and increase msg_count by 1
    Else send REJECT(ev)
End if
If(receive(ACCEPT(EST(FBA), cap(FBA))))
    add FBA to S
end if
X= powerset(S)
For all $x_i$ Є X
    For all $FBA_j$ Є $x_i$
        tot_cap($x_i$) += cap ($FBA_j$)
```

```
        end for loop
        if(tot_cap($x_i$) >= cap)
                add $x_i$ to C
        end if
end for loop
for all $x_i$ Є C
        for all $FBA_j$ Є $x_i$
                EST($x_i$)= max(EST($FBA_j$))
                coalition.put($x_i$, EST)
        end for loop
end for loop
for all $c_i$ Є coalition
        if(EST($c_i$) <= EST)
                min_est_coalitio.add($c_i$)
        end if
end for loop
choose the coalition with smallest size and maximum trust factor as done in coaltiton formation algorithm
$C_b$ = min_est_coalition
Send INFORM($C_{b,}$ EST) and increase msg_count by 1
If (receive OK message form all $FBA_i$ Є $C_b$)
        send OK($C_b$, ev, EST) to FSA and increase msg_count by 1
else send CANCEL(ev) message to FSA
if(receive CANCEL(ev)) the set no_of_failure (FBA)+=1
report failure(ev)
```

Algorithm 7: Euclidean_dist

Input: x(FBA), y(FBA), x(ev), y(ev) **Output: dist**

Algorihtm: Return Math.sqrt(((x(ev)-x(FBA))*(x(ev)-x(FBA)))+((y(ev)-y(FBA))*(y(ev)-y(FBA))))