

DYNAMIC TASK ALLOCATION APPROACH TO MULTI-FORAGING IN GROUP OF ROBOTS

*Dissertation submitted to Jawaharlal Nehru University,
in the partial fulfillment of the requirement for
the award of degree of*

**MASTER OF TECHNOLOGY
In
COMPUTER SCIENCE & TECHNOLOGY**

By

Sunil Kumar

Under the Guidance of

Prof. K.K. Bharadwaj



**SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067**



जवाहरलाल नेहरू विश्वविद्यालय
SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067 (INDIA)

CERTIFICATE

This is to certify that the dissertation entitled “**DYNAMIC TASK ALLOCATION APPROACH TO MULTI-FORAGING IN GROUP OF ROBOTS**”, being submitted by **Mr. Sunil Kumar** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi in partial fulfillment of the requirement for the award of the Degree of **Master of Technology in Computer Science and Technology**, is a bona fide record of original work done by him under the supervision of **Prof. K.K. Bharadwaj**, during the 2006-2007.

The matter embodied in the dissertation has not been submitted in part or full to any other University or Institution for the award of any Degree or Diploma.

Sunil Kumar
(Student)

Prof. K.K. Bharadwaj
(Supervisor)

Prof. Parimala N:
Dean

School of Computer & Systems Sciences
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-11.067

Dean
School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi- 110067

ACKNOWLEDGEMENTS

The first person I would like to thank is my supervisor Prof. K.K. Bharadwaj. I have been his student since 2005 when I started my M.Tech. During these years I have known Prof. Bharadwaj as a sympathetic, helping, motivator, and principle-centered person. His overly enthusiasm and integral view on research and real life and his mission for providing only high-quality works, has made a deep impression on me. He could not even realize how much I have learned from him. I am sincerely thankful for his invaluable guidance for every thing during my M.Tech.

Now, I would like to convey my thanks to Prof. S. Balasundaram the Dean of the School of Computer and Systems Sciences for providing me the laboratory and other facilities.

Finally, I am highly thankful to my parents and family member for their emotional support to encourage my research interest. I also acknowledge my seniors, my friends for their cooperation during my dissertation work.

At last, my thank goes to the beautiful and round the clock environment of JNU Campus that always makes me feel good.

SUNIL KUMAR

SC&SS

JNU, New Delhi

ABSTRACT

Allocation of tasks to a group of robots working in unknown and dynamic environment is one of the fundamental problems in multi-robot systems. Dynamic task allocation is a category of task allocation in which allocation of task to group of robots is an ongoing process. Tasks allocation is adjusted according to the change in environment such as change in the state of robot, change in performance of robot, and change in the number of tasks or robots.

In this dissertation, we have tried to explore how the task allocation is applied in dynamic environment and developed a scheme for Multi-Foraging in a group of robots based on Dynamic Task Allocation. Several experiments are conducted using MissionLab-7.0 (A Multi-robot simulator) based on the proposed scheme with varying puck density, homogeneous and heterogeneous teams of robots on multi-foraging domain including objects of different sizes and colors.

We have proposed an allocation algorithm and appropriate architecture for solving the problem of task allocation in dynamic environment in Multi-Foraging domain.

With the help of simulation it is also shown how the problem of interference affects the performance of different group of robot. Experimental results are presented to demonstrate the effectiveness of the proposed scheme.

At last, we concluded with a summary of the contributions of this dissertation in multi-foraging domain with some discussion on future work.

CONTENTS

	Page
ABSTRACT	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
1. INTRODUCTION	1-5
1.1 Foraging.....	2
1.2 Multi-Foraging.....	3
1.3 Dynamic Task Allocation.....	3
1.4 Organization of Dissertation.....	5
2. BACKGROUND AND LITERATURE SURVEY	6-25
2.1 Multi Robot Task Allocations.....	6
2.1.1 Task Allocation.....	7
2.1.2 Static Task Allocation.....	7
2.1.3 Definition of Multi-Robot Task Allocation.....	8
2.1.4 Utility Function.....	8
2.2 Classification of Multi-Robot Task Allocation.....	9
2.2.1 Axes of Task Allocation in Multi-Robot.....	9
2.2.2 Organization of Tasks and Robots with respect to Assignment.....	10
2.3 Dynamic Task Allocation.....	11
2.4 Multi-Foraging.....	13
2.4.1 Multi-Foraging Task.....	14
2.4.2 Behavior-Based Control for Multi-Foraging.....	15
2.4.3 Multi-Robot Interference.....	18
2.5 Methodology of Task Allocation.....	19
2.5.1 Murdoch: Publish/Subscribe System.....	19
2.5.2 Auction Algorithm.....	20
2.5.3 Ant Algorithm.....	21
2.5.4 Task Allocation as a Free Market Architecture.....	21
2.5.5 ALLIANCE.....	22

2.5.6	Broadcast Local Eligibility (BLE) using Port Arbitration Behavior (PAB)	23
2.5.7	Team Formation-Based Task Allocation.....	24
3.	MULTI-FORAGING BASED ON DYNAMIC TASK ALLOCATION	26-35
3.1	The Overview.....	26
3.2	Algorithms for Multi-Foraging.....	27
3.2.1	Homogeneous Teams of Robot.....	28
3.2.2	Heterogeneous Teams of Robot.....	30
3.3	Robot Behavior Finite State Automata (FSA).....	32
4.	IMPLEMENTATION AND EXPERIMENTAL RESULT	36-55
4.1	The Simulator.....	36
4.1.1	Simulation Parameters.....	37
4.1.2	Team and Environment Configuration.....	38
4.2	Implementation.....	39
4.2.1	FSA and Activation Condition for Robot.....	39
4.2.2	Compilation and Execution.....	44
4.3	Simulation Results and Analysis	49
4.3.1	Comparisons of Homogeneous and Heterogeneous Group.....	49
4.3.2	Interference in Homogeneous Group.....	53
5.	CONCLUSIONS	56-57
	APPENDIX A	58-60
	REFERENCES	61-63

List of Tables

2.1	Behavior Activation Condition [Lerman et al., 2006].....	18
4.1	Simulation Parameters.....	37
4.2	Detail of the task and homogeneous group.....	38
4.3	Detail of the task and heterogeneous group.....	38
4.4	Behavior Activation Condition for Single Home.....	40
4.5	Behavior Activation Condition for Multiple Homes.....	41
4.6	Foraging Time for team of size two for single Home.....	50
4.7	Foraging Time for team of size three with single Home.....	50
4.8	Foraging Time for team of size three with three Home locations..	51
4.9	Foraging Time for team of size three with three Home locations..	51
4.10	Homogeneous Team Foraging Time (sec).....	55

List of Figures

2.1	Dynamic Task Allocation.....	12
2.2	ALLIANCE architecture [Parker, 1998].....	22
3.1	Finite State Automata for the robot for foraging task.....	34
4.1	FSA for Homogeneous Robot with single Home location.....	40
4.2	FSA for Homogeneous Robot with three Home locations.....	41
4.3	FSA for the Robot _{Red} Specialized to Collect the Red_Object to Home_Red	42
4.4	FSA for the Robot _{Green} Specialized to Collect the Green_Object to Home_Green.....	43
4.5	FSA for the Robot _{Blue} Specialized to Collect the Blue_Object to Home_Blue.....	44
4.6	Team of Six MRV2 Robots.....	45
4.7	Compilation of robot.....	45
4.8	Runtime Options.....	45
4.9	Execution of three Homogeneous Robots with single Home location.....	46
4.10	Team of three Homogeneous Robots with three Home locations and objects of different sizes	47
4.11	Execution of Six Homogeneous Robots with three Home locations..	47
4.12	Execution of three Heterogeneous Robots with three Home locations	

and objects of different sizes	48
4.13 Comparison of Heterogeneous and Homogeneous Groups size of two.....	50
4.14 Comparison of Heterogeneous and Homogeneous Groups size of two.....	51
4.15 Comparison of Heterogeneous and Homogeneous Teams of size three with three Home locations.....	52
4.16 Comparison of Heterogeneous and Homogeneous Teams of size six with three homes.....	53
4.17 Interference during the execution of foraging task.....	54
4.18 Foraging Time (sec) for Homogeneous group	55

1. Introduction

Since the early 1990s, a research on Group of Robots, operating in dynamic environment has been given a special attention by the researchers. The rapid development in hardware, and software associated with it, leads great interest in Multi-Robot research. The complex sensor device and robot controller improved the capability of robots for accomplishing the complex task such as localization, path planning, object transportation, object recognition and tracking etc, efficiently.

The important aspect of the group of robots is the task mission as a whole. The group of robots are well suited for several application domains, which require highly coordinated task to be performed in dynamic and dangerous environment for example space, military, and rescue operation etc.

Most research in multi-robot team has centered on the homogeneous teams, with works in heterogeneous systems focused primarily on mechanical and sensor differences. But team of identical robots are also interesting because they may be homogeneous or heterogeneous depending only on agent behaviors. The most important point of multi robot system is "*they can work in cooperative manner*" [Gerkey and Mataric, 2001]. In other words they have ability to achieve the task as compared to single robot. In this dissertation, We worked upon both types of teams, heterogeneous and homogeneous, to demonstrate the team behaviors of the robots.

The foraging has a strong biological basis. The term "Foraging" comes from nature, which is seen in many insects (e.g. Bees, Ants) for food gathering. In

ant species, for instance, perform the forage task as they gather food. The foraging task is divided in two parts: *simple foraging* and *multi-foraging* [Balch, 1999a]. The work presented in dissertation focus on Multi-Foraging.

1.1 Foraging

In simple foraging task, there is single type object and delivered on the single destination. The robot is to wander or search about the environment looking for object of interest or attractor. To encounter an attractor, the robots moves towards and grasp it. After attachment, the robot returns the object to specified home location [Balch, 1999a].

Foraging and collection task are the oldest and most studied problem in the team of robots. In these tasks, the robot or group of robots has to collect the objects scattered in the environment and assemble them on a predefined place called “home” location [Lerman and Galstyan, 2002]. Foraging robot teams are configured as either homogeneous or heterogeneous a priori, and then their performance is comparatively evaluated [Balch, 1999b] [Gerkey and Mataric, 2003].

There are several reasons to study foraging in a group of robots. Besides providing a test-bed for the design of physical robots and their controllers, foraging serves as a useful framework for exploring many issues in the design and implementation of multi-robot teams. Additionally, deploying a team of robots to perform a collection or a foraging task is often of practical importance: it introduces robustness and parallelism. Many robots working in parallel manner may complete the task faster and failure of single robot doesn't affect the performance [Lerman and Galstyan, 2002].

Foraging is reduced to a problem of transportation when the locations of the sources and sinks are known. When there are multiple sources and sinks, the transportation problem becomes the Multi-Robot Task Allocation problem where

transportation tasks must be allocated to robots in a way that optimizes overall performance [Goldberg and Mataric, 2000] [Stone and Veloso, 1998].

1.2 Multi-Foraging

In multi-foraging domain, different type of objects are randomly dispersed in the environment. The object can be of different color (i.e. red, green, blue etc). The task, multi-foraging, requires robot to collect different types of object and deliver them to different location according to their type [Gerkey and Mataric, 2004].

The performance in multi-foraging task is defined as the number of object collected and properly delivered in a fixed time. Several environmental parameters like interference, obstacle in environment, type of robot, number of robots, field size affect the rate at which robot collect and deliver [Lerman and Galstyan, 2002].

This is extensive use of multi-foraging is find in many real world problem like mining operation, explosive ordnance disposal, and waste or specimen collection, rescue operation in different environment etc.

1.3 Dynamic Task Allocation

This dissertation is specifically concerned with the dynamic allocation of tasks. This is an important requirement for multi-robot systems operating in an unknown environment. The allocation of the tasks to a group of robot in such a way, that robot has to change its state continuously in response to change in environment that will enhance the overall system performance and complete the mission efficiently.

The challenge faced by the designer is to devise a mechanism that will lead to a desired task allocation in a distributed Multi-Robot System that can work

robustly during the changing in environment. The challenge is made even more difficult by the fact that robots have limited sensing capabilities, due to the indirect communication with other robots and, therefore cannot acquire global information about the state of the world, the initial or current number of tasks (total or by type), or the initial or current number of robots (total or by assigned type). Instead, robots can sample the world (assumed to be finite) – for example, by moving around and making local observations of the environment. We assume that robots are able to observe tasks and discriminate their types. They may also be able to observe and discriminate the task states of other robots [Lerman et al., 2006].

In the context of multi-robot coordination, dynamic task allocation can be viewed as the selection of appropriate actions for each robot at each point of time so as to achieve the completion of the global task by the team as a whole. From a global perspective, in multi-robot coordination, action selection is based on the mapping from the combined robot state space to the combined robot action space. Task allocation plays an important role to handle the Multi-Foraging in the different teams of robots. In multi-foraging task, there are different types of objects, can be of different colors, to be arranged on different random locations. In the task allocation mechanism robots use local observation of the environment to decide their task assignment.

In this dissertation, I tried to explore how the task allocation is applied in dynamic environment. It includes development of a scheme for Multi-Foraging in a group of robots based on Dynamic Task Allocation. The experiment would be conducted using the proposed scheme with:

- Varying puck density
- Homogeneous and heterogeneous teams of robot
- Extended multi-foraging task including objects of different shape

in addition to different colors.

1.4 Organization of Dissertation

The rest of this dissertation is organized as follows. In the next chapter, we discuss about the Background and Literature survey of the Dynamic Task Allocation, Multi-Foraging etc.

Chapter 3 describes the Dynamic task allocation on Multi-Foraging in which we proposed the architecture of the problem and task allocation algorithm.

In Chapter 4, we describe the implementation and result of the purposed work. We conclude in Chapter 5 with a summary of the contributions of this dissertation and future work.

2. Background and Literature Survey

One of the important trends in Multi-Robotic system is to study the group of robots as a team which works in coordinated manner. Most important advantage of the multi-robot system is that it can improve the ability of a single robot by cooperation and coordination. Extensive research has been done in this particular area. [Maja J. Mataric, 2000, 2002] research on control and learning in behavior-based multi-robot systems and dynamic task allocation, [Lynne E. Parker, 1997] designed the ALLIANCE that works in dynamic action selection for situation in which the robot fails if the mission changes, the robot team composition changes, or the environment changes, Ronald Arkin and Tucker Balch did work on behavior-based team formation in multi-robot [Yingying *et al.*, 2003].

Research performed under such titles as multi-agent robotics, distributed robotic system, decentralized robotics, swarm robotics (e.g., Fukuda, Nakagawa, Kawauchi & Buss (1988), Deneubourg, Theraulaz & Beckers (1991)), and cellular robotics etc, has focused on the investigation issues and applications of system composed of group of robots. Multi-robot teamwork is a complex problem consisting of task allocation, foraging, coordination and cooperation, communication etc. [Baghaei and Agah *et al.*, 2002]

2.1 Multi Robot Task Allocations

As a result of the growing focus on multi-robot systems, multi-robot coordination has received significant attention. In particular, *multi-robot task*

allocation (MRTA) has recently risen to prominence and become a key research topic in its own right. As researchers design, build, and use cooperative multi-robot systems, they invariably encounter the fundamental question: “*which robot should execute which task?*” (e.g. Brian P. Gerkey, Maja J Mataric) in order to cooperatively achieve the global goal. By “*task*,” we mean a sub-goal that is necessary for achieving the overall goal of the system, and that can be achieved independently of other sub-goals (i.e., tasks). Tasks can be discrete (e.g. deliver this package to room 157) or continuous (e.g. monitor the building entrance for intruders) and can also vary in a number of other ways, including timescale, complexity, and specificity [Gerkey and Mataric, 2004].

2.1.1 Task Allocation

The Complicated problem is divided in sub-problems, so that they can be solved easily and efficiently. For example foraging problem can be divided in avoid obstacle, pick object, search object etc sub-problems. These sub-problems are known as a task, which is necessary to achieve the overall solution of the given problem. The process of assigning a task to a robot, according to its type, and to the environment, is known as task allocation. The allocation of task is made in a way that reduce the overall mission completion time, this means that it enhance the overall system execution time.

According to the allocation of the task, it can be divided in two parts that is Static Task Allocation and Dynamic Task Allocation.

2.1.2 Static Task Allocation

In static approach, tasks allocation is method of robot team formation, to perform during the system design. All robot has predefined and similar task (e.g. R.C. Arkin, T. Balch). The robot has to work according to its pre-allocated way. If

the environment changes, the system can fail. The information about the tasks and environment must be known in advance, like number of tasks, numbers of robots etc. If the complete information is not available, than the static task allocation is not possible.

2.1.3 Definition of Multi-Robot Task Allocation

Let us suppose, we have given m robots and n tasks. Each task required one robot. The performance of each robot is estimated and rank is given by some non negative number. The goal is to Assign(i,j) means i^{th} task to j^{th} robot in a way that can increase the overall performance of the system taking into account the priority of the task and the rank of the robot [Gerkey and Mataric, 2004].

But this definition has some limitations. There are problems of dynamic decision as there can be chances of robot failure and environment is dynamically changing very frequently [Gerkey and Mataric, 2003].

2.1.4 Utility Function

The utility function is defined in order to provide a quantitative measure of the effectiveness estimated by each robot [Iocchi *et al.*, 2003]. Each robot has to estimate the value or cost, or the fitness of action which is going to be performed. The estimation mainly include following factors [Gerkey and Mataric, 2004]:

- Expected Cost of Resources, given the spatio-temporal requirement of the task.
- Expected Quality of task execution, given method and equipment to be used.

Let us suppose we have a robot R capable of executing a task T . Let C_{RT} cost factor and Q_{RT} quality factors of task T . The Utility function can be defined as [Gerkey and Mataric, 2004]:

$$U_{RT} = \begin{cases} Q_{RT} - C_{RT} & \text{if R is capable of Executing T and} \\ & Q_{RT} > C_{RT} \\ 0 & \text{Otherwise} \end{cases}$$

The definition of Utility function is an important issue used in Multi-Robot System design. Utility estimates are not useful sometimes due to the noise, general uncertainty and change in environment, as they give wrong values.

2.2 Classification of Multi-Robot Task Allocation

While designing a multi-robot system, it is essential to understand what kind of task allocation problem is present in order to solve it in a principled manner.

2.2.1 Axes of Task Allocation in Multi-Robot

According to number of tasks and number of robots, [Gerkey and Mataric, 2004] defines following three axes for use in describing MRTA problems

i. Single-Task Robots (ST) vs. Multi-Task Robots (MT)

ST means that each robot is capable of executing at most one task at a time, while MT means that some robots can execute multiple tasks simultaneously.

ii. Single-Robot Tasks (SR) vs. Multi-Robot Tasks (MR)

SR means that each task requires exactly one robot to achieve it, while MR means that some tasks can require multiple robots to accomplish it for example Box-Pushing.

iii. Instantaneous Assignment (IA) vs. Time-extended Assignment (TA):

IA means that the available information concerning the robots, the tasks, and the environment permits only an instantaneous allocation of tasks to robots, with no planning for future allocations.

TA means that more information is available, such as the set of all tasks that will need to be assigned, or a model of how tasks are expected to arrive. The complete information about the task and environment is required before allocation.

2.2.2 Organization of Tasks and Robots with respect to Assignment

The taxonomy of task allocation is given as the combination of axes which are already explained in section 2.4.1. The taxonomy of Multi-robot task allocation is elaborated in following section which may be used as prescription of solutions [Gerkey and Mataric, 2004].

- i. Single-Task Robot, Single-Robot Task, Instantaneous Assignment
- ii. Single-Task Robot, Single-Robot Task, Time-Extended Assignment
- iii. Single-Task Robot, Multi-Robot Task, Instantaneous Assignment
- iv. Single-Task Robot, Multi-Robot Task, Time-Extended Assignment
- v. Multi-Task Robot, Single-Robot Task, Instantaneous Assignment
- vi. Multi-Task Robot, Single-Robot Task, Time-Extended Assignment

- vii. Multi-Task Robot, Multi-Robot Task, Instantaneous Assignment
- viii. Multi-Task Robot, Multi-Robot Task, Time-Extended Assignment

2.3 Dynamic Task Allocation

Allocation of tasks to a group of robots working in unknown environment is one of the fundamental problems in multi-robot system. The real world is dynamic in nature. To cope with such environment is very difficult as it do not have any centralized system for controlling the allocation of tasks in group of robots. In distributed multi-robot, the problem of task allocation is more complex due to the lack of global information about the environment as local sensing limitation of each robot. Robot works in group on the basis of local sensing information and local interaction [Lerman *et al.*, 2006].

Dynamic task allocation is category of task allocation in which allocation of task to group of robots is an ongoing process. Tasks allocation is adjusted according to the change in environment such as change in the state of robot, change in performance of robot, and change in the number of tasks or robots. Each robot works independently in coordination of other robot as a group. The appropriate behavior is selected for each robot at each point of time so that overall accomplishment of mission can be achieved. Let's assume that we have an environment having number of available task. Robot chooses any task from that available task pool based on some rules and regulation, as given in figure 2.1 is an example of Dynamic Task allocation.

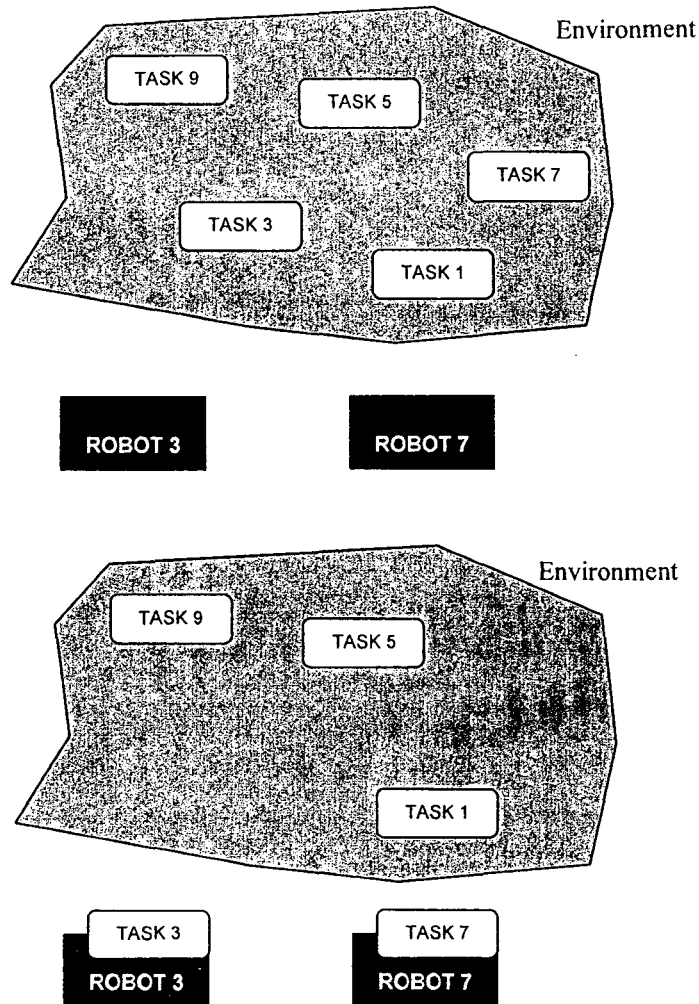


Figure 2.1: Dynamic Task Allocation

To achieve the dynamic allocation, every robot has to work till all the tasks have not been accomplished. In homogeneous teams of robot, all robot works together to accomplish the mission as a group task without any specialization and differences in the behavior of robot. Each robot perform task in same manner as other robot in team. But in case of heterogeneous teams of robot, each robot is specialized to perform a particular type of task. Each robot involves in particular type of task at each point of time. If one robot completes one type task, it coordinated with others and involves in other task of different type to improve the overall performance of the system. From Multi-robot coordination perspective, dynamic task allocation is a mechanism of selecting and appropriate

action for each robot at each point of time so as to achieve the completion of the global task by the team as a whole [Østergaard *et al.*, 2002] .

2.4 Multi-Foraging

Multi-robot foraging has been studied by many researchers. In robotics, foraging is very important field of study to use in solving of many real world problems. Goldberg & Mataric and Parker works on design of behavior-based controller and Holland and Melhuish on reactive controller, for foraging and task collection [Lerman and Galstyan, 2002]. Arkin and Balch investigate several behaviorally strategy for robot foraging and impact of communication in foraging team (homogeneous and heterogeneous) [Balch, 1999]. There are following previous work as a contribution given by the different researcher in foraging [Lerman and Galstyan, 2002] :

I. Task Type

- a. Collection [Beckers, Holland, and Deneubourg, 1994; Martinoli Ijspreert A.J. and Gambardella L.M., 1999)
- b. Foraging [Mataric, 1992; Goldberg & Mataric, 2000; Nitz, Arkin, & Balch, 1993]

II. System type

- a. Individual Robot
- b. Group of Robots (Homogeneous [Goldberg and Mataric, 2000] and Heterogeneous [Goldberg and Mataric, 2000; Parker, 1994])

III. Controller type

- a. Reactive (Holland and Melhuish, 2000; Martinoli Ijspreert A.J. and Gambardella L.M., 1999)
- b. Behavior-Based [Mataric, 1992; Goldberg and Mataric, 2000]

- c. Hybrid [Nitz, Arkin, and Balch,1993]

IV. Communication

- a. No Communication (Goldberg and Mataric, 2000)
- b. Direct (Nitz, Arkin, and Balch,1993; Sugawara and Sano, 1997)
- c. Stigmergetic (through modification of the environment) (Holland & Melhuish, 2000; Vaughan, Sty, Sukhatme, and Mataric, 2000b)

Foraging is one of the oldest and well studied problem in robotic. It is the modified form of robot collection problem. In robot collection problem robot or group of robot has to collect the object scattered in the environment on some random location. But in foraging, robot or group of robot has to collect the object on predefined location called “home”.

2.4.1 Multi-Foraging Task

In simple foraging task, individual robot or group of robot has to collect the object scattered in environment on particular location. Multi-foraging task is the variation of simple foraging. It consists of different type of objects scattered in the environment. Robot or group of robots has to collect these objects on some particular location or different location according to types of objects. The *multi* means different type of objects not different types of robot. The object scattered in the environment is called “*Puck*” and location on which these object are assembled are called “*Home*” location. [Lerman *et al.*, 2006] uses two types of objects (pucks) in Multi-foraging domain that is $Puck_{Red}$ and $Puck_{Green}$. These objects are distinguished by the color.

More formally we can define Multi-Foraging task in two steps [Goldberg and Mataric, 2000]

- A. Lets suppose we have n robots, where n is greater than and equal to 1, search selected regions of space for certain objects, and

- B. Find the Puck_{*i*}, and brought that Puck_{*i*} to a Home location using some form of navigation.

Where *i* is type of puck according to color.

[Balch, 1999a] gives three types of strategies for multi-foraging. These are implemented and evaluated for performance and diversity:

i. Behavior Homogeneous:

In this strategy all the robot collect different type of objects and deliver them on corresponding type of home location.

ii. Territorial or Behavior Heterogeneity:

In this scheme, [Balch, 1999a] use two type of robot. One type robot is responsible for searching and delivering of the object near the boundary of the Home zone. And second type robot is responsible for collecting the object placed on boundary of home region. This robot called sorting agent.

iii. Specified By Color:

In this scheme, some robot is specialized to collect one type of objects and rests of the robots are responsible for other type of object, in case of two color objects. It is also heterogeneous robot strategy.

2.4.2 Behavior-Based Control for Multi-Foraging

“*Think the way you act*” [Mataric, 2001]: The behaviors are observable patterns of activity emerging from interactions between the robot and its

environment. The behavior-based controller has strong biological inspiration. Most of the researchers working on multi-robotic system try to apply the social characteristics of insects on coordination in group of robots. The control policies find in various biological societies mostly in ants, bees, and birds, are used to the development of similar behaviors in cooperative robot systems [Mataric, 2001].

The Behavior-based controller consists of set of behaviors. [Lerman *et al.*, 2006] purposed a behavior based controller for Multi-Foraging consisting of following mutually exclusive behaviors:

I. The Wandering:

The behavior cause the robot to move forward, left or right randomly in the environment for searching. Sometime we called this behavior searching for the object or puck.

II. The Avoiding:

This behavior causes the robot to turn either left or right direction randomly if there is any obstacle in the path of robot.

III. The Puck Servoing:

This behavior causes the robot to detect and move toward the detected object or desire type. The type can be on the basis of color or size or shape. [Lerman *et al.*, 2006] foraged only Puck_{Red} puck and Puck_{Green} puck. If robot detect Puck_{Red} puck then the foraging state of that is Robot_{Red} and for green is Robot_{Green}.

IV. The Grasping:

This behavior causes the robot to pick up the desired type of puck in the gripper of the desire type of robot.

V. The Observing:

This behavior causes the robot to observe the environment and store the information related to robots and puck information.

All robots have same behavior-based controller. Depending on the relevant sensor input and state value, robot selects the appropriate behavior to activate from the above listed behavior. [Lerman *et al.*, 2006] purposed the activation conditions for behaviors of the robot are described as follow:

I. The Puck_{*i*} Detected:

The activation condition is true when robot detect an object or puck and robot current foraging state is Robot_{*i*} and a puck of type Puck_{*i*} where *i* can be either Red or Green. Robot uses some sensor that sensor and its range varies from robot hardware to hardware.

II. The Obstacle Detected:

This activation condition is true when robot detect some obstacle in a particular sensor range. If robot is holding puck and it detect another puck, in that situation that detected puck is considered as an obstacle.

III. The Gripper Beak-Beam On:

There is a sensor in the jaws of robot to sense the availability of the Puck between the jaws of robot [Lerman *et al.*, 2006]. The sensor *on/off* state is depending on the presence of puck in the jaws of robot. If the Puck is in the jaws then the sensor state is *on* otherwise *off*. The Gripper Beak-Beam On activation condition is true when break-beam sensor detect the puck inside the jaws.

IV. The Observation Signal:

This activation condition is true after the particular distance traveled by the robot from the last time the Observing behavior was activated.

The relation between activation and behavior can be shown as in following table:

Obstacle Detected	Puck_{det} Detected	Gripper Break-Beam On	Observation Signal	Activation Behavior
×	×	×	1	Observation
1	×	×	×	Avoiding
0	1	0	0	Puck Servoing
0	×	1	0	Grasping
0	×	×	×	Wandering

Table 2.1: Behavior Activation Condition [Lerman *et al.*, 2006]

× Denotes the activation condition is irrelevant for the corresponding behavior.

2.4.3. Multi-Robot Interference

The robot works as a team to improve the overall all system performance. The group robots working in parallel may increase the collection of tasks. Since we increase the number of robots, the degrees of interference between robots increase accordingly. In the extreme case environment becomes crowded. In this way, the most of time and energy of robot is wasted in avoiding other robots.

[Balch, 1999a] defines interface as it is measured as time spent by the robot to avoid each other, when two robots attempt to occupy same object at the same time. Since interference may slow down the system performance. Several approaches are used to minimize the interference like communication and

cooperative strategies such as trail function, the bucket brigade etc. The [Goldberg *et al.*, 2002] suggests pack and caste negotiation method to decrease interference and for gathering efficient behavior. In Pack approach, each robot has assigned an order in pack hierarchy, the robot having higher order in hierarchy pick and deliver the object before the other robots. The Case approach is a Territorial strategy in which one robot is responsible for the final delivery of the objects from the boundary of home to home and rest of the robots are involved in collecting the objects on the boundary of the home. This interference time is minimized in heterogeneous pack system [Uchibe *et al.* 2001].

2.5 Methodology of Task Allocation

In some cases, it will be possible to construct provably optimal solutions, while in others only approximate solutions are available. There are also some difficult Task Allocation problems for which there do not currently exist good approximations. Task allocation methodologies are explained in following section:

2.5.1 Murdoch: Publish/Subscribe System

Murdoch is auction based MRTA architecture used in heterogeneous team of robots. For Murdoch, the robots might have characteristics like multipurpose, can communicate, cooperative, distributive control and can determines its own progress etc. This architecture provides a tradeoff between communication overhead and solution quality, producing greedy solutions at a low cost, in a distributed manner [Brian *et al.* 2001].

It is a Dynamic Task Allocation mechanism in which each robot subscribes to a set of tasks based on available resources. Robot can subscribe to different message based on their capability and state information.. Murdoch declares and defines tasks' subjects referred to as subject-base addressing

including robot capabilities and robot states. The system then publishes that message. This message is addressed by contents and the robot who subscribed to that subject will receive the associated message. The Best Fit selection algorithm is used to choose the best robot from the registered robots [Baghaei and Agah *et al.*, 2002]

2.5.2 Auction Algorithm

Auction algorithms are task allocation algorithm basically used in homogeneous teams of robot. These algorithms are guaranteed to produce optimal allocations, although they can incur significant communication overhead and take many rounds to converge to stability. This algorithm can be utilized in multi-robot task allocation applications, particularly suitable for parallel computation [Baghaei and Agah *et al.*, 2002] There are many auction algorithms, but Bertsekas(1992) gives one of the best-known algorithm as below :

Algorithm 5.2 : (Bertsekas Algorithm)

1. Initially perform the following:
 - a. Randomly assign Task to Robot
 - b. Prices the tasks
2. Randomly select Robot_{*i*}, which don't have task any more.
 - a. If no such robot then Market has reached stability
Exit
3. Find the Task_{*i*} that increase the profit for Robot_{*i*}
4. Swap tasks between Robot_{*i*} and robot that is currently assign the Task_{*i*}.
5. Increment the price of Task_{*i*}.
6. Return to Step 2.

This auction algorithm is guaranteed to terminate with assignment of task with optimal utility [Baghaei and Agah *et al.*, 2002]

2.5.3 Ant Algorithm

This method is on the basis of biology in ants. When ants move, they leave some pheromone from their tails. Amount of the pheromone affects the activity of other ants as they work in coordination through pheromone. The probability of following is depends on the higher pheromone on the tails. The Multi-robot is a distributed system in which robot can work alone or as a team member.

The basic idea of ant algorithm is based on the adaptability of the group of ants for their environment changes. This approach can be considered as task allocation [Baghaei and Agah *et al.*, 2002] There are number of task and ants, task allocated to the ant depends on the probability function. The more difficult task has higher pheromone than easier one, hence higher probability. More pheromone attracts more robots toward that difficult task to cooperate and to accomplish it. An important feature of this approach is the indirect communication between ants, resulting in emergent behavior [Baghaei and Agah *et al.*, 2002].

74-14674

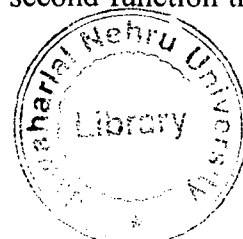
2.5.4 Task Allocation as a Free Market Architecture

This task allocation methodology is based on the free market system. In this technique, robot is self centered and managed as an economical entity. Each robot acts based on its own benefit. These separate benefits are added and generate total team profit. Higher the team profit, better the group performance. To achieve the better system performance, robots have to be cooperated to improve the overall benefits.

The system performance is measured based on the Revenue/ Cost balance.

$$\text{Benefit} = \text{Revenue} - \text{Cost}$$

The ultimate goal is to maximize the benefits. This methodology is based on two function, cost function and revenue function. The revenue function maps the result of each action to a revenue value and a second function that maps each



method for performing a single task to its cost values. The calculation of the minimum value of the difference between the two functions results in a factor for selecting the most suitable task. Dynamic task allocation, group learning, and minimum communication dependability are some important features of this approach [Baghaei and Agah *et al.*, 2002]

2.5.5 ALLIANCE

L.E. Parker proposed ALLIANCE which is fully distributed, behavior based software architecture that is capable of providing robot teams that

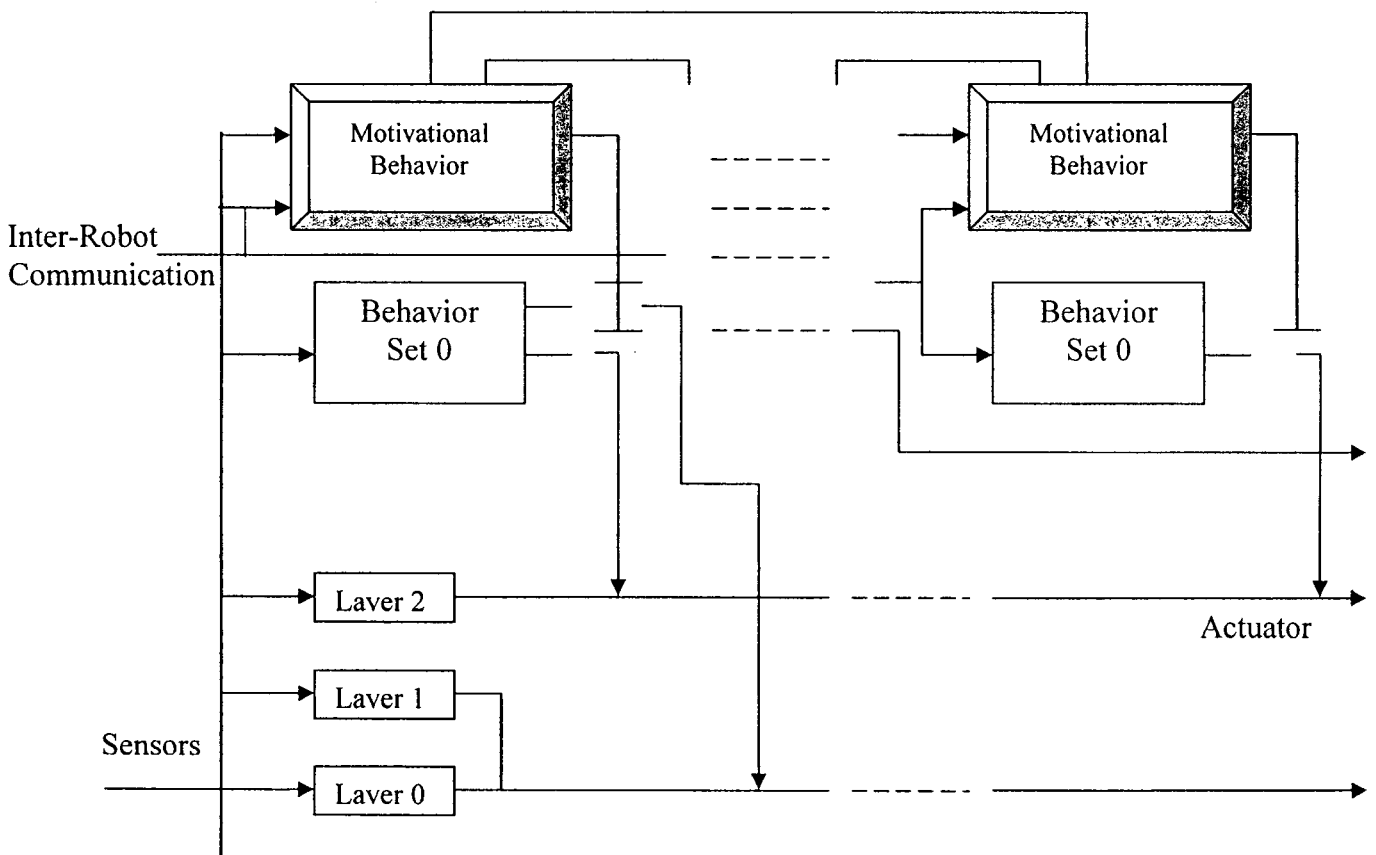


Figure 2.1: ALLIANCE architecture [Parker, 1998]

are able to deal with fault tolerance, dynamic action selection for situation in which robots fail, the mission change, the robot composition change, or the environment changes [Parker, 2002]. This is focused on up to medium size heterogeneous teams of robot.

ALLIANCE is behavior based controller use different sets of behavior for different tasks. Alliance process is executed for multi-robot cooperation. Task allocation between different robots with different structure takes place in alliance [Baghaei and Agah *et al.*, 2002] Motivational Behavior is used to control the process of selecting the appropriate action from the behavior sets. In this mechanism, two mathematical modeled motivations - impatience and acquiescence are used within each robot to achieve adaptive action selection. On the basis of current rates of impatience and acquiescence, sensor feedback and knowledge of other team member activities, the level of motivational behavior is calculated for corresponding behavior set. Once the level of abstraction is crossed the threshold, the corresponding behavior set is activated, and robot has selected an action [Parker, 2002] [Parker, 1998]. The ALLIANCE is extended to self learning system L-ALLIANCE by incorporating the monitor for each motivational behavior within each robot. The monitor provides task-oriented action selection mechanism into behavior based system [Parker, 1997].

2.5.6 Broadcast Local Eligibility (BLE) using Port Arbitration Behavior (PAB)

BLE is behavior based approach apply for task allocation with fixed priority tasks. Each robot has corresponding behavior for every task as it is capable of executing the task. BLE uses PAB technique for conflict resolution among robots. PAB uses a collection of Behavior Production Module (BPM) for programmed code to produce robot behavior. The local eligibility of the robot is estimated by BPM as robot's utility for the task. Utilities are computed on the bases of sensor data. These utilities are broadcasted to all the robots through

interface port. The robot having best eligibility produces the desire behavior and inhibits other robot behavior [Gerkey and Mataric, 2003] [Baghaei and Agah *et al.*, 2002]. [Werger & Mataric, 2000] design BLE algorithm to solve the problem of iterative task assignment in MRTA

Algorithm: BLE Task Assignment

1. If Robot_{*i*} is idle then
 Find the Robot_{*i*} for Task_{*j*} which has highest utility
 Else
 Exit.
2. Assign Task_{*j*} to Robot_{*i*} and remove them from consideration.
3. Go To Step 1.

2.5.7 Team Formation-Based Task Allocation

This technique is mainly used in robot soccer application and dynamic task allocation. A team formation decomposes the task space defining a set of roles. It includes as many roles as there are robots in the team, to fill each role by one robot. In addition, formations can specify sub-formations, which do not involve the whole team. A unit consists of a subset of roles from the formation, and intra-unit interactions among the roles. Homogeneous robot can flexibly switch roles within formations, and robots can change formations dynamically, so that pre-defined strategy evaluated at run-time. This change in robot internal state cause the change in external behavior and finally its effect on environment will change. In this way, new

task requirement acquiesced for other robots. The change in interstate of robot is communicated to other robots in order to generate a new team formation. This flexibility increases the performance of the overall team [Stone and Veloso, 1998] [Gerkey and Mataric, 2003].

3. Multi-Foraging Based on Dynamic Task Allocation

One of the greatest challenges in robotics is to create machines that are able to interact with unpredictable environments in real time. A possible solution may be to use swarms of robots [McLukin and Yamins, 2005] behaving in a self-organized manner, similar to workers in an ant colony. Efficient mechanisms of division of labor, in particular series- parallel operation and transfer of information among group members, are key components of the tremendous ecological success of ants. Here we show that the general principles regulating division of labor in ant colonies indeed allow the design of the initial colony energy. The initial colony energy is proportional to the number of robots per colony. As a result of it the good modulation of the number of individuals engaged in the two possible activities: staying in the nest (inactive) and foraging (active). Finally, for some trials robots were programmed to recruit another robot when they identified a resource-rich area, thus imitating recruitment behavior observed in many ant species. Hence, the robots have generalized information about the overall colony energy, ways to avoid interfering with one another in space and time, and, in the last experiment, the ability to transfer useful information to others. (e.g. Michael J. B. Krieger, Jean-Bernard Billeter & Laurent Keller, 2000)

3.1 The Overview

World is dynamic, which is changing frequently according to moment of time. In real life, human being has to face the unknown and dynamic environment

which implies that we are trying to replace the human being by the robot to deal with dynamic environment.

The role of dynamic task allocation in Multi-foraging domain requires the robots to divide their numbers by having some foraging for one type of objects (e.g. Green) and other for second types of objects (e.g. Red Color) and so on. More formally, the allocation of task to a group of robot in dynamic environment is called dynamic task allocation. The robot has to change its state according to the change in environment or action of the other robots in order to improve the overall system performance. The important point in dynamic task allocation in Multi-Foraging is selection of appropriate task at each moment of time to achieve the completion of the global task (collect all the objects to their corresponding home location(s)) by the team of robots. In the work we purposed an algorithm in Multi-Foraging domain on following situations:

- Varying density of objects
- Homogeneous teams and Heterogeneous teams of robot
- Objects of different color, shape and size with single and multiple locations.

The team of homogeneous and heterogeneous robots are compared with various object density, and team size with single and multiple home locations.

3.2 Algorithms for Multi-Foraging

The Team can be constructed as Homogeneous Team or Heterogeneous Team depends on the available task and requirement of the mission. In homogeneous team, all robots are of same type having same behavior-based controller. While the heterogeneous group of robots is focused on mechanical and sensor differences. The following algorithms are purposed for dynamic task allocation in Multi-foraging domain:

3.2.1 Homogeneous Teams of Robot:

Homogeneous teams of robots consists of same type of robots having same type of hardware and software, to perform similar tasks. The main aim of the robots are to arrange the objects on single home location or corresponding home location based on the type of object. In homogeneous teams of robots, all the robots having same controller involved in foraging different types of objects. For Homogeneous teams of robots, following algorithms are suggested for dynamic task allocation in multi-foraging domain with the following assumption:

Let there are total N task and M robots are involving in the mission. During mission execution, if $\text{Task}_{x,i}$ has highest priority for Robot_j then this task $\text{Task}_{x,i}$ is allocated to Robot_j where x is the different type of task based on color, or size, or shape and i and j are the task and robot number respectively.

Algorithm 3.1:

```
While  $\text{Task}_{x,i}$  is available in the environment
{
    If  $\text{Robot}_j$  is Wandering for task Then
        o Find  $\text{Task}_{x,i}$  which has the highest priority for the
           $\text{Robot}_j$ 
           $\text{Task}_{x,i} = \text{MaxPriority}(\text{Task}_{x,i}, \text{Robot}_j)$ ;
        o Allocate  $\text{Task}_{x,i}$  to  $\text{Robot}_j$ 
           $\text{Allocate}(\text{Task}_{x,i}, \text{Robot}_j)$ ;
        o Remove The allocated task from task pool
           $N = N - \{\text{Task}_{x,i}\}$ ;
    EndIf

    If New  $\text{Task}_{x,k}$  occurs in environment Then
         $N = N \cup \{\text{Task}_{x,k}\}$ ;
```

```

EndIf

If Robot  $x_j$  has Task $x_i$  Then
    o Deliver Task $x_i$  on corresponding Home or Home $x$ ;
    o Make Robot $j$  HasObject state Null.
EndIf
}
End Loop.

```

The MaxPriority(Task x_i , Robot j) Function is used to find the Task which has the highest priority in the wander robot sensing area. The Priority is calculated in two ways:

- I. According to the distance between Object and Robot when robot is away from home


```

If distance(Object $x_i$ , Robot $j$ ) < distance(Object $y_i$ , Robot $j$ )
then
    Return(Object $x_i$ )
Else
    Return(Object $y_i$ )
Endif

```
- II. According to the distance between Object and Home Location when robot is at Home or Home x .


```

If distance(Object $x_i$ , Home or Home $x$ ) < distance(Object $y_i$ , Home or Home $x$ )
then
    Return(Object $x_i$ )
Else
    Return(Object $y_i$ )
Endif

```

In homogeneous teams of robot all robot has same behavior and behavior based controller. The team of robots has internal coordination and works in

cooperation. If one robot is not able to perform one task, the task is shifted to another robot to handle the failure of one robot.

3.2.2 Heterogeneous Teams of Robot:

In Heterogeneous teams of robot consists of different types of robot having different type of hardware and software, which are specialized in particular type of task.

Let there are total N task and M robots are involving in the mission. During mission execution, if $\text{Task}_{x,i}$ has highest priority for $\text{Robot}_{x,j}$ then this task $\text{Task}_{x,i}$ is allocated to $\text{Robot}_{x,j}$ where x is the different type of task based on color, or size, or shape and i and j are the task and robot number respectively

Algorithm 3.2:

```

While  $\text{Task}_{x,i}$  is available in the environment
{
    If  $\text{Robot}_j$  is Wandering for task Then
        o Find  $\text{Task}_{x,i}$  which has the highest priority for the
           $\text{Robot}_j$ 
               $\text{Task}_{x,i} = \text{MaxPriority}(\text{Task}_{x,i}, \text{Robot}_{x,j})$ ;
        o Allocate  $\text{Task}_{x,i}$  to  $\text{Robot}_{x,j}$ 
               $\text{Allocate}(\text{Task}_{x,i}, \text{Robot}_{x,j})$ ;
        o Remove The allocated task from task pool
               $N = N - \{\text{Task}_{x,i}\}$ ;
    EndIf

    If New  $\text{Task}_{x,k}$  occurs in environment Then
         $N = N \cup \{\text{Task}_{x,k}\}$ ;
    EndIf
}

```

```

    If Robotx,j has Taskx,i Then
        o Deliver Taskx,i on corresponding Home or Homex;
        o Make Robotx,j HasObject state Null.
    EndIf
}
End Loop.

```

The MaxPriority(Task_{x,i} , Robot_{x,j}) Function is used to find the Task which has the highest priority in the wander robot sensing area. The Priority is calculated in two ways:

I. According to the distance between Object and Robot when robot is away from home

```

If distance(Objectx,i , Robotj) < distance(Objecty,i , Robotj)
then

```

```

    Return(Objectx,i)

```

```

Else

```

```

    Return(Objecty,i)

```

```

Endif

```

II. According to the distance between Object and Home Location when robot is at Home or Home_x.

```

If distance(Objectx,i , Home or Homex) < distance(Objecty,i ,
Home or Homex) then

```

```

    Return(Objectx,i)

```

```

Else

```

```

    Return(Objecty,i)

```

```

Endif

```

In this algorithm Robot_{x,j} is specialized to perform x types of tasks e.g. Robot_{Red,j} is specialized to forage the Red Objects on particular home location and Robot_{Green,j} is specialized to forage the Green Objects. There can be two types

of home location in Multi-foraging. In first type, there is single home location and all the robots have to collect all the objects of different type on this location only. In second type, there is multiple home location e.g Red-Home, Green-Home etc. The red and green objects are collected on Red-Home and Green-Home location respectively.

An expansion of heterogeneous teams of robot that makes system more robust and robot works in more coordinated way. In this expansion, if Robot_{x,j} finish all Task_{x,i} then its moves to accomplish the Task_{y,i} tasks. Each robot involves in foraging until the mission is not over to increase the over all performance of the system and to cope the changing environment.

3.3 Robot Behavior Finite State Automata(FSA)

The FSA is a representation of continuous states used to activate most relevant continuous process operating in behavior-based robotics. FSA is designed to sequence from one state to another. The behavior control for each robot is encapsulated in sensor motor. FSA consists of related states which execute the related behavior for robots working in dynamic environment. It acts as behavior manager.

Formally, the FSA machine M is defined as quadruple (Q, δ, q_0, S)

where:

- Q is set of allowable states

$$Q = \{q_0, q_1, q_2, q_3, \dots\};$$

- δ is transition function used to map the current executing state to same state or another continuous states.

$$\delta: q_0 \rightarrow q_1$$

- q_0 is Initial state from where we'll start FSA.
- S is set of accepting states and S subset of Q .

An example of FSA that is represented by quadruple:

$$M(\{q_0, q_1, q_2, q_3, q_4, q_5\}, \delta, q_0, \{q_2, q_3, q_4\});$$

The FSA for the robot performing task in Multi-foraging domain is representing in figure 3.1. The FSA consists of following states:

$Q = \{\text{Start, Wander For Object, Pick up Object, Avoid Obstacle, Wander for Home, Move toward Home, Deliver Object, Stop}\}$

$q_0 = \{\text{Start}\}$

δ is transformation function which is used to convert one state to another based on Activation Condition like ObjectDetect, ObstacleDetect, HoldingObject, HomeDetect, NearHome etc. FSA states are explained as bellow:

- ***Start***

It is an initial state. That determines the starting of the mission.

- ***Wander for Object***

After start state, robot moves to wander state and start wandering in random direction searching for the object of particular type scattered in the environment. If ObjectDetect=1 then it moves on *Pick up Object* state. If ObstacleDetected=1 then Avoid Obstacle state is activated.

- ***Pick up Object***

When ObjectDetect=1, robot activate Pick up Object state. Robot grasps the object and moves in wandering state while holding the objects.

- ***Avoid Obstacle***

While robot detects an obstacle, it activates *Avoid Obstacle* state. This state is incorporate with wandering behavior of the robot.

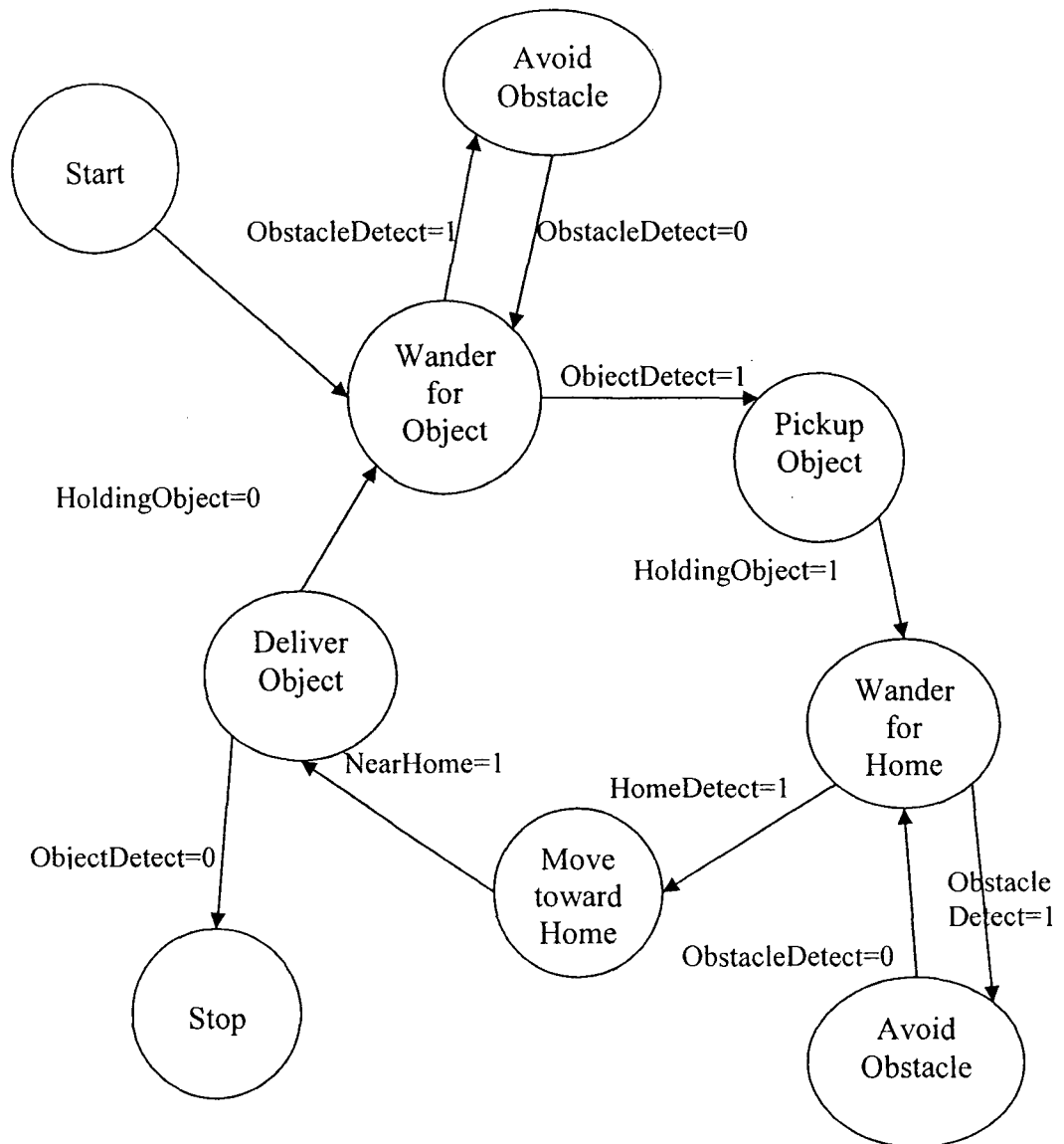


Figure 3.1: Finite State Automata for the robot for foraging task.

- ***Wander for Home***

When robot pick up an object (HoldingObject=1) then robot start searching for the appropriate Home location according to the type of object. The obstacle avoidance is still there.

- ***Move toward Home***

When robot detects Home location (DetectHome=1) then it move toward that home to deliver the object.

- ***Deliver Object***

Robot drop the object near home location and again move to next state depends on the availability of objects in the environment. If the DetectObject=0 then robot moves to stop state else it moves again to Wander for Object state.

- ***Stop***

In this state, robot has no more task and it stop working and moves to stop state.

4. Implementation and Results

The simulation system is utilized to perform experiments with various control strategies for the robot team and team organizations, evaluating the comparative performance of the strategies and organizations. The objective of the robot team, once deployed in an environment with multiple tasks, is to accomplish as many tasks as possible. The simulated robots are capable of navigation through the environment, and can communicate using simple messages. The simulator maintains the world, provides each robot with sensory information, and carries out the actions of the robots. The simulator keeps track of the tasks completed by robots and the elapsed time, in order to evaluate the performance of the robot teams. The robot teams can be composed of homogenous robots or heterogeneous robot. In homogeneous robot, identical control strategies are used to generate the behavior of each robot in the team. But in heterogeneous robot, particular type of control strategies are used depends on the type of task.

This chapter presents the implementation of the purpose algorithms using MissionLab-7.0 simulator. In order to experimentally demonstrate the dynamic task allocation in multi-foraging domain an artificial environment is used as a physically realistic environment. My simulation experiments were performed using multi-robot simulator MissionLab-7.0.

4.1 The Simulator

The simulator is provided with a variety of parameters to specify the scope of the experiment and then displays the robots as they are working in real

environment. In each time step, a robot uses its sensors to detect objects, and the sensory data is then mapped to the rule sets (Behavior-Action pairs), resulting in the associated action. MissionLab-7.0 is a 2-D and 3-D GUI based multi-robot simulator. It consists of powerful set of software tools for developing and testing behaviors for single robots and a group of robots. Code generated by MissionLab7.0 can directly control commercial robots. ATRV-Jr / Urban Robot (iRobot), AmigoBot / Pioneer, MRV2, Hammer, and Nomad are among those robots MissionLab-7.0 has supported successfully. A primary strength of MissionLab-7.0 is its support of both simulated and real robots. A developer can experiment with behaviors in simulation and then run those same configurations on mobile robots.

For more detail please refer to *Appendix A*.

4.1.1 Simulation Parameters

Before execution of the mission, there are some parameters which we have initialized at the initial stage of experimentation. Our complete mission is basically consists of three main component, robot, puck and mission area. Every component has some parameter. The parameters for our mission are specified in Table 4.1.

Parameter	Name or Value	Parameter	Name or Value
No. of Robots	2-8	Mission Area	31 × 30 (meter)
Type of Team	Homogeneous and Heterogeneous	Start Point	(2.5,3.0)
Robot Radius	0.70 meter	Distance to the Area edge	5 meter
Name of Robot	MRV2	Mission Execution Time	Variable
No. of Puck	20 to 54		
Type of Puck	2-3		
Puck Radius	0.3 meter		
Number of Home	1 or 3		
Home Radius	1.5 meter		

Table 4.1: Simulation Parameters

4.1.2 Team and Environment Configuration

The group can be composed as team of Homogeneous or Heterogeneous robots. Several experiments are conducted on different number of robots and various puck densities. In Multi-foraging domain there can be single and more than one home on which robot has to collect the Puck of desired type. So according to team composition, I divide configuration in two groups.

Homogeneous Group: In this composition, the team is group of homogeneous robots as given in table 4.2.

Team Composition	Team Size	Types of Object (Color)				Number of Home
		Red	Green	Blue	Total	
Homogeneous	2	12	8	0	20	1
Homogeneous	3	12	8	7	27	1
Homogeneous	3	12	8	7	27	3
Homogeneous	6	24	16	14	54	3

Table 4.2: Detail of the task and homogeneous group

Heterogeneous Group: In some situation homogeneous robot is not able performed better. To overcome these situations we need different specialized robot called heterogeneous robot. In the simulation on heterogeneous robots the following configuration group is used (Table 4.3).

Team Composition	Team Size	Types of Object (Color)				Number of Home
		Red	Green	Blue	Total	
Heterogeneous	2	12	8	0	20	1
Heterogeneous	3	12	8	7	27	1
Heterogeneous	3	12	8	7	27	3
Heterogeneous	6	24	16	14	54	3

Table 4.3: Detail of the task and heterogeneous group

There is multiple home location and multiple types of Puck defined as:

Home = {Home_Red, Home_Green, Home_Blue}

Puck = {Red_Object, Green_Object, Blue_Object}

4.2 Implementation

Each experiment is carried out using MissionLab-7.0 and is treated as a mission. The mission is defined by Configuration Description Language (CDL). The algorithm can be written in CDL or CNL (Configuration Description Language). The CDL includes a Graphical configuration editor (CfgEdit) that is used to specify the instantiation and coordination of primitives and not their implementation. Therefore, each of the primitives must have a CDL definition which specifies the programming interface.

4.2.1 FSA and Activation Condition for Robot

For every robot, there is a different behavior controller that represent as Finite State Automata. FSA based mission specification consist of one or more than state with a trigger. Once the condition in the trigger got true, the robot will transition to the next state to which the trigger is pointed.

- I. **Homogeneous robots:** The Figure 4.1 presented the operating behavior and specified the FSA required implementing the task allocation in multi-foraging domain having single home. The Figure 4.2 is the FSA for three home location. Table 4.4 represents the conditions Activation for the robot having behavior in Figure 4.1 and Table 4.5 for Figure 4.2.

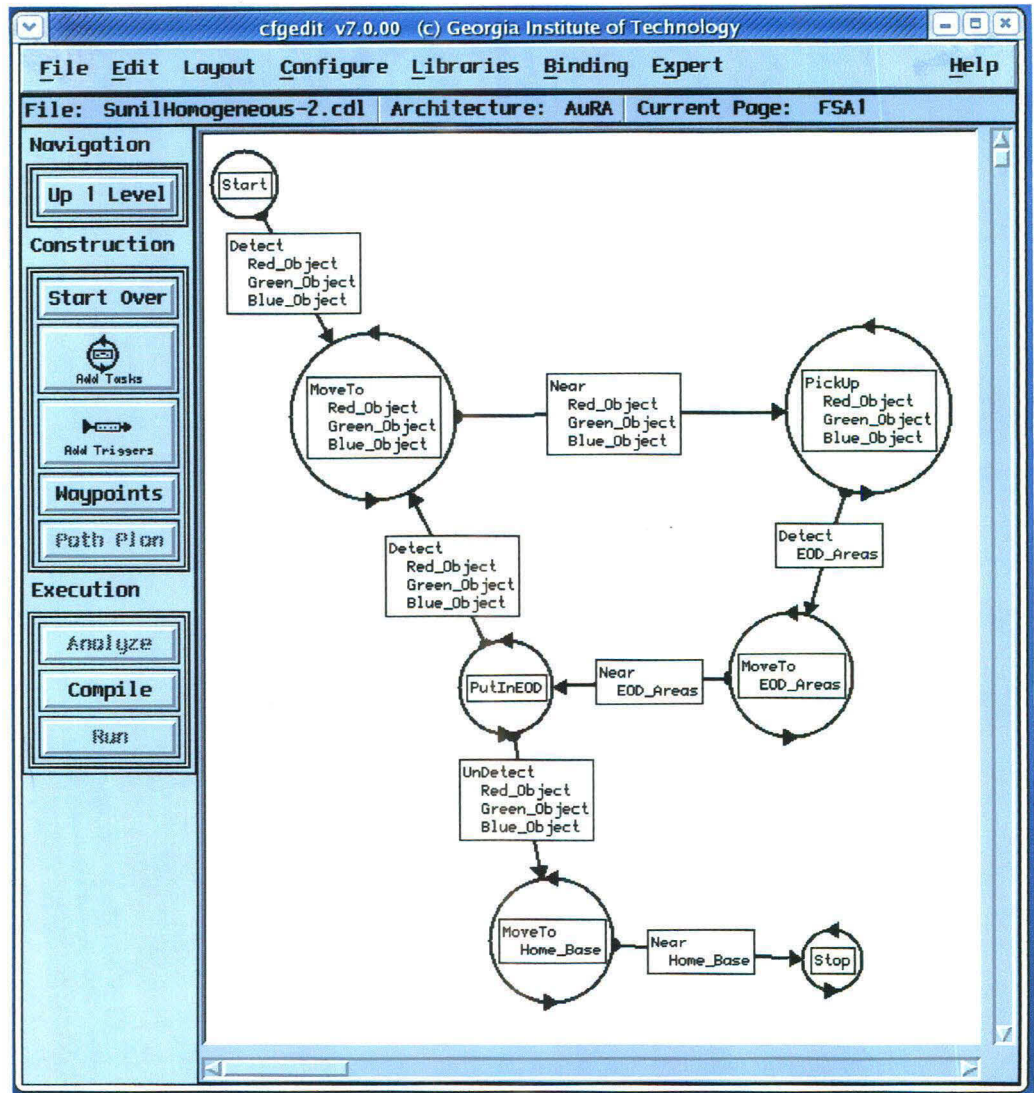


Figure 4.1 FSA for Homogeneous Robot with single Home

location.

Detect Object _{<i>i</i>}	NearObject _{<i>i</i>}	Holding Object _{<i>i</i>}	Detect Home	NearHome	
0	0	0	×	×	WanderForObject
1	0	0	×	×	MoveToObject_{<i>i</i>}
1	1	0	×	×	PickUp_{<i>i</i>}
0	0	1	0	0	WanderForHome
×	×	1	1	0	MoveToHome
×	×	1	1	1	PutIn_{<i>i</i>}

$i \in \{\text{Red, Green}\}$

× represent that condition is not applicable.

Table 4.4: Behavior Activation Condition for Single Home

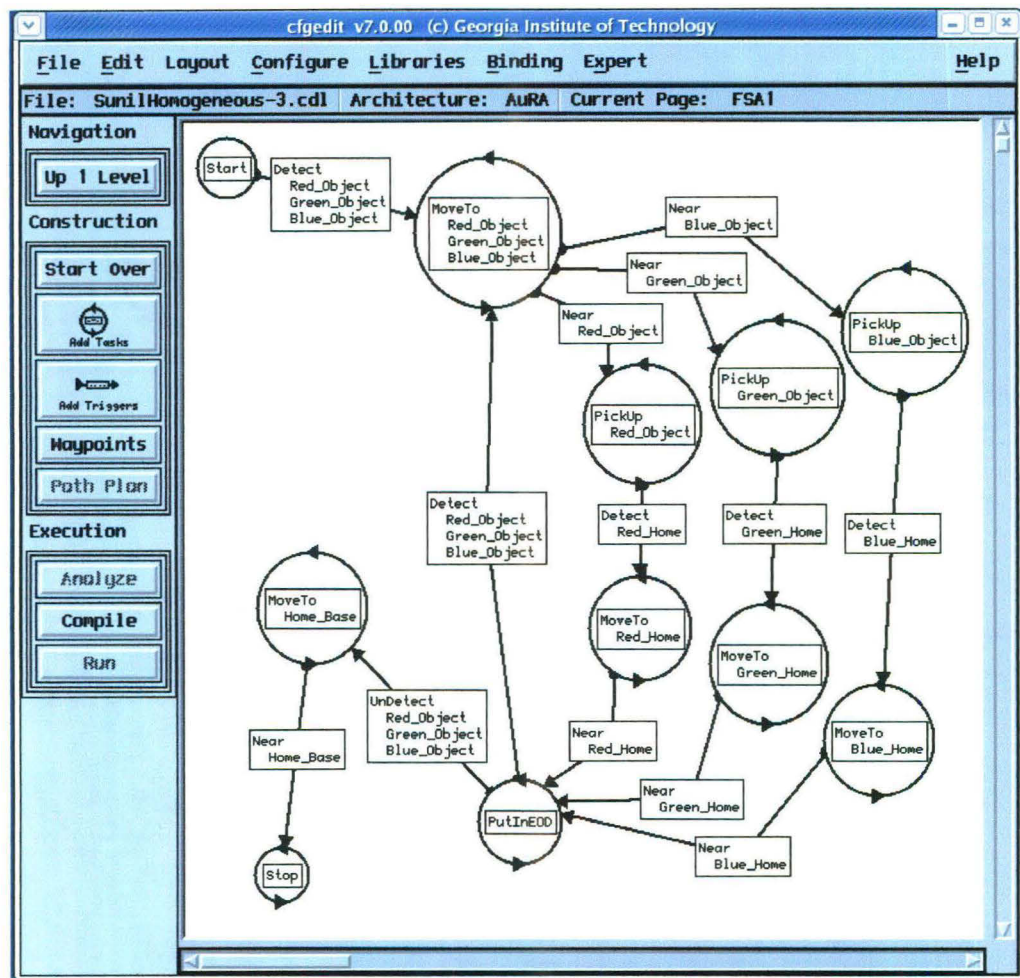


Figure 4.2 FSA for Homogeneous Robot with three Home locations.

Detect Object _{<i>i</i>}	NearObject _{<i>i</i>}	Holding Object _{<i>i</i>}	Detect Home	NearHome	
0	0	0	×	×	WanderForObject
1	0	0	×	×	MoveToObject_{<i>i</i>}
1	1	0	×	×	PickUp_{<i>i</i>}
0	0	1	0	0	WanderForHome
×	×	1	1	0	MoveToHome_{<i>j</i>}
×	×	1	1	1	PutIn_{<i>ij</i>}

$$i \in \{\text{Red, Green}\} \quad j \in \{\text{Red_Home, Green_Home, Blue_Home}\}$$

× represent that condition is not applicable.

Table 4.5: Behavior Activation Condition for Multiple Homes

II. Heterogeneous robot: In heterogeneous team of robots, every robot is specialized for a particular type of task hence we use to design different FSA for every robot. The robot $Robot_{Red}$ is used to detect and collect the Red_Object, $Robot_{Green}$ and $Robot_{Blue}$ used for Green_Object and Blue_Object respectively. The Figure 4.3 represents the FSA for $Robot_{Red}$, the Figure 4.4 and Figure 4.5 used for $Robot_{Green}$ and $Robot_{Blue}$ respectively. Table 4.4 represents the conditions Activation for the single Home location and Table 4.5 for three Home Location, for the robot represent in Figure 4.3, Figure 4.4, and Figure 4.5.

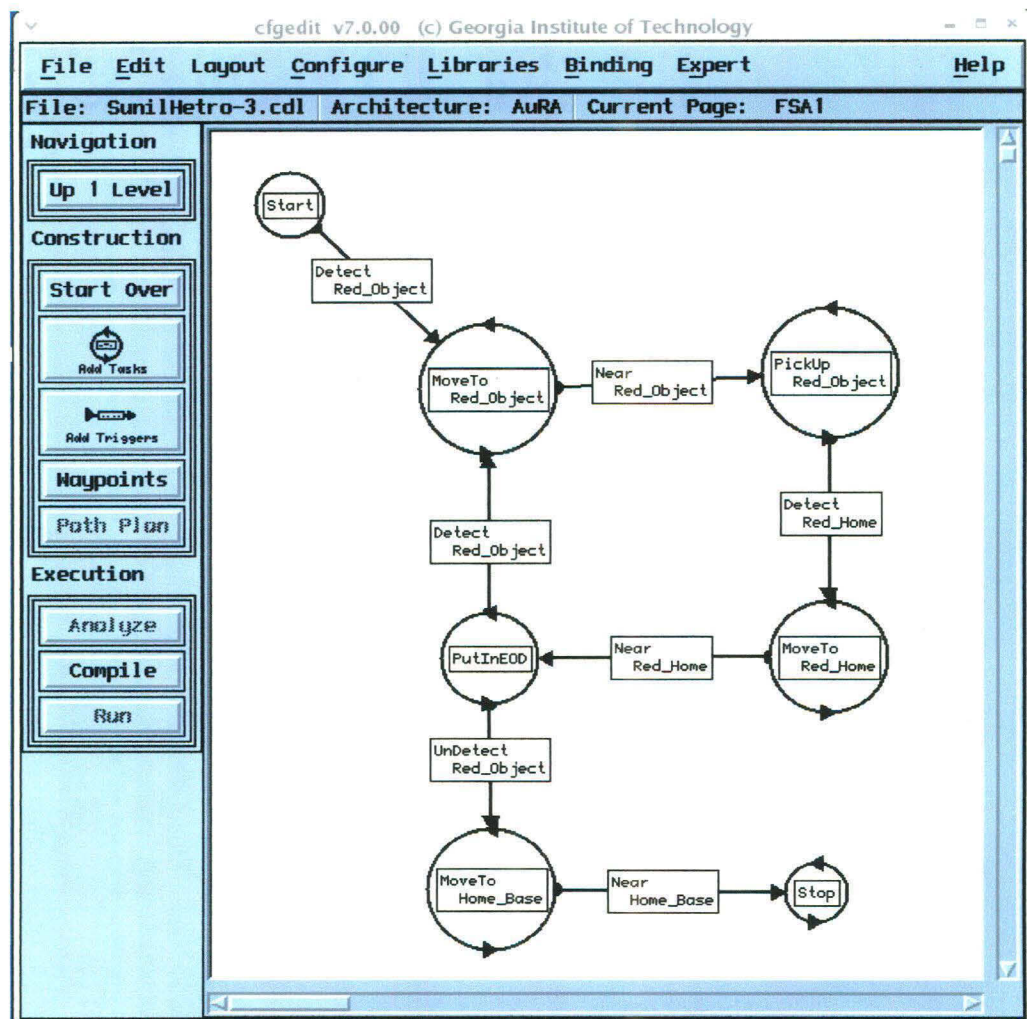


Figure 4.3: FSA for the $Robot_{Red}$ Specialized to Collect the Red_Object to Home_Red

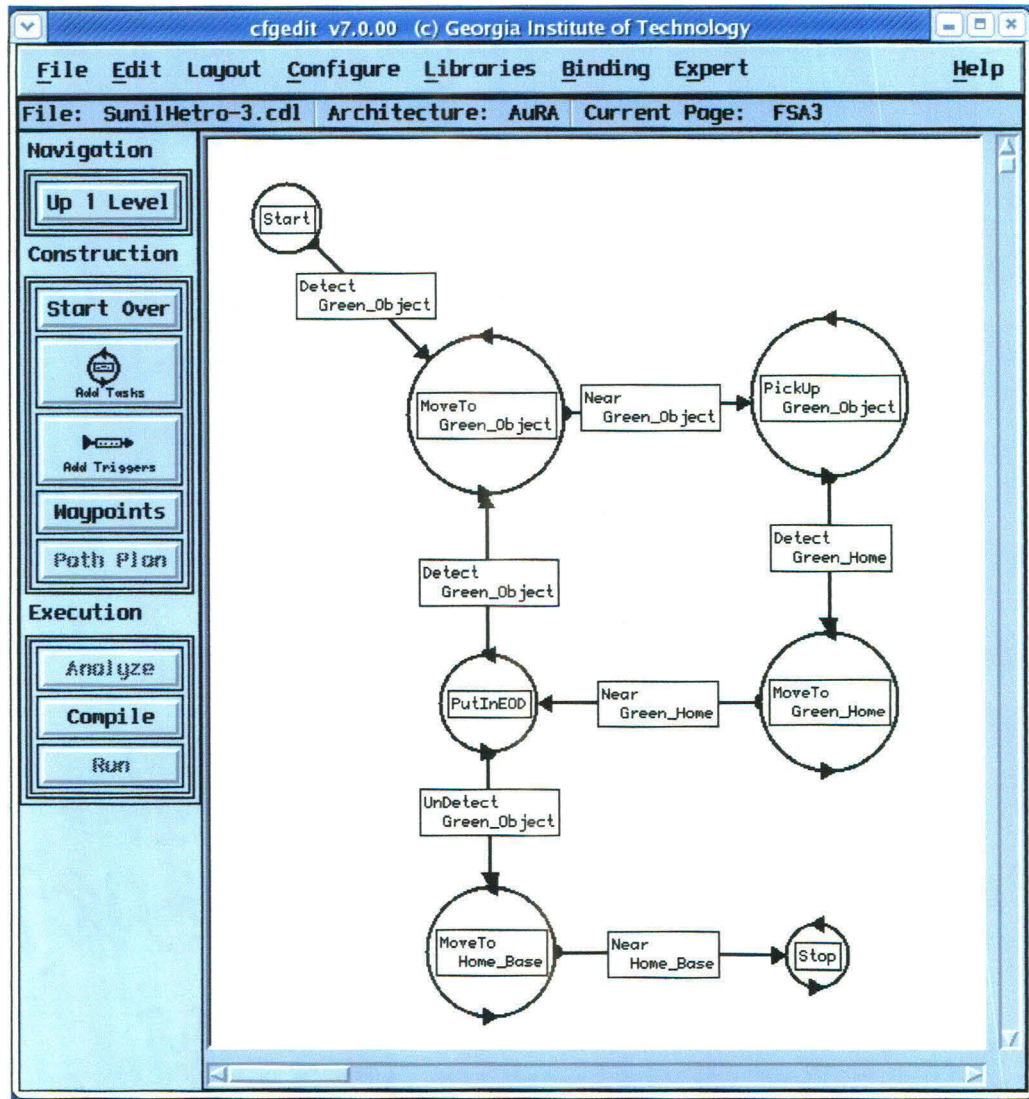


Figure 4.4: FSA for the Robot_{Green} Specialized to Collect the Green_Object to Home_Green

The FSA shown Figure 4.3 to Figure 4.5 are used for different robot. Each FSA consists of set of states and corresponding triggers. For example given in Figure 4.4, the initial state (behavior) of the robot is start state. When robot detect a green object in the environment, the detect Green_Object trigger fired and robot transit to next state that is MoveTo Green_object and so on. At end, if robot not detect any object, it MoveTo Home_Base and stop the foraging. The whole is procedure is game of states and triggers.

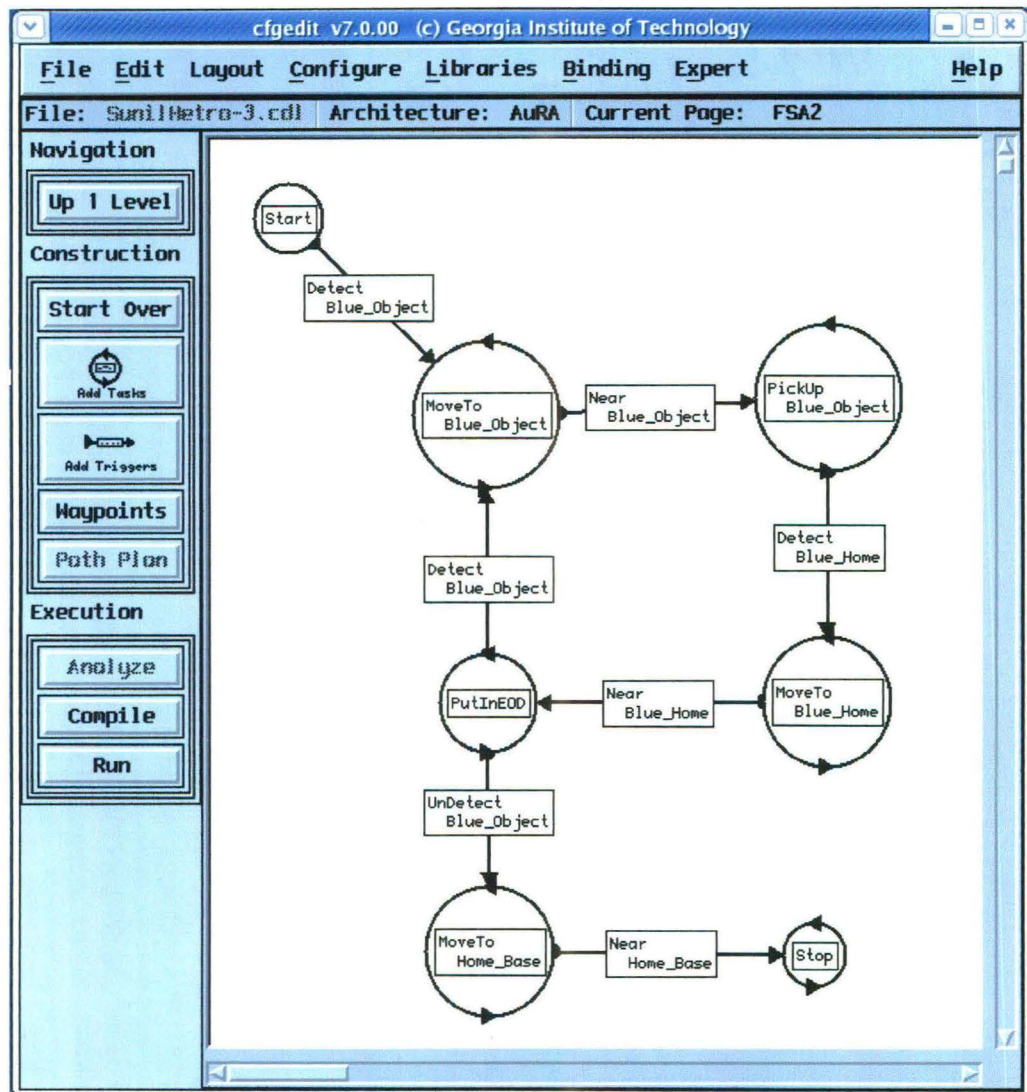


Figure 4.5: FSA for the Robot_{Blue} Specialized to Collect the Blue_Object to Home_Blue

4.2.2 Compilation and Execution

After designing the FSA for robot of particular type, the team of Homogeneous or Heterogeneous is constructed depends on the mission requirement as shown in Figure 4.6.

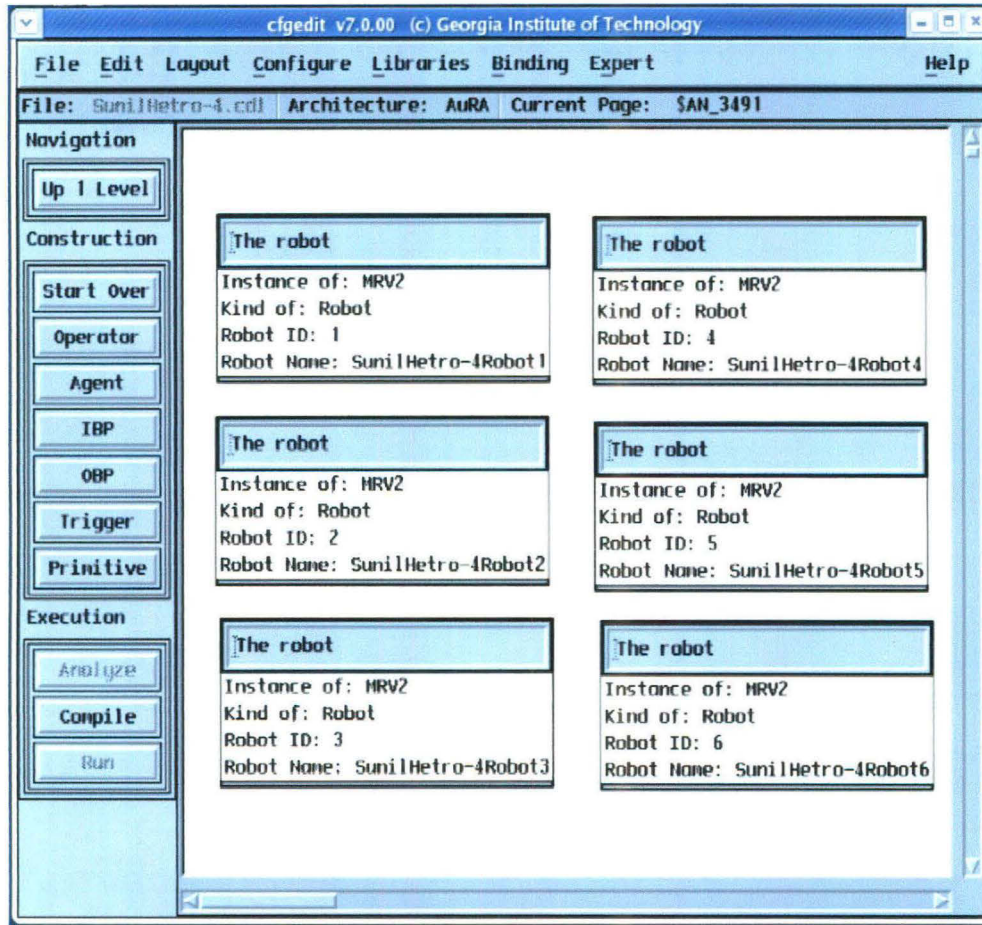


Figure 4.6: Team of Six MRV2 Robot

Now we can create a robot executable by Compiling the Configuration Description Language (CDL) file as represented by the Figure 4.7. After compilation the mission is ready to use.

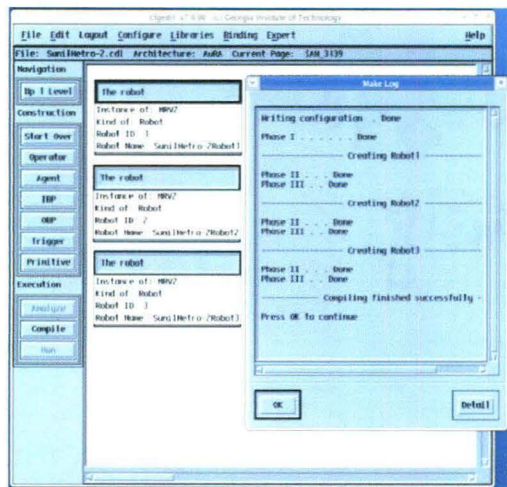


Figure 4.7: Compilation of robot

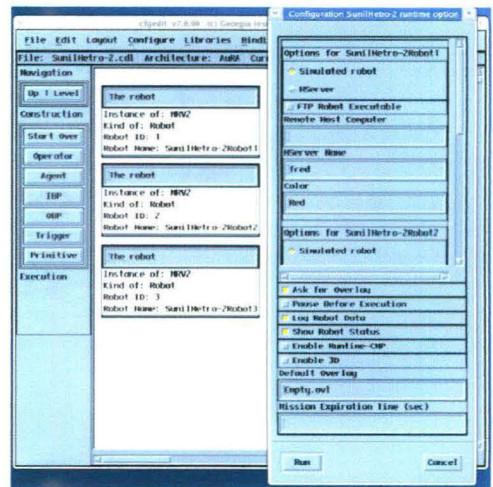






Figure 4.8: Runtime Option

We have to initialize some parameter before execution according to the requirement of mission. There are following Runtime Options like Simulated robot or Hserver, overlays file, Show robot state, 2-D or 3-D visualization of the execution, and Mission expiration time that is used to terminate the mission in unusual situation or infinite execution. The mission runtime option is given in Figure 4.8. During the initial step of running state, simulator launches the robot in given environment where object are scattered randomly. The environment is created by Overlays file, that consists of mission are, starting point, obstacle, description of objects like radius, position and color etc., description of Home position, type of home etc. We can add the obstacle and new objects during the run time. The Figure 4.9 to 4.11 shows the environment and state of robots performing multi-foraging. In following figure the shape  represents Robot,  and shapes   represent object of red, green and blue color respectively.

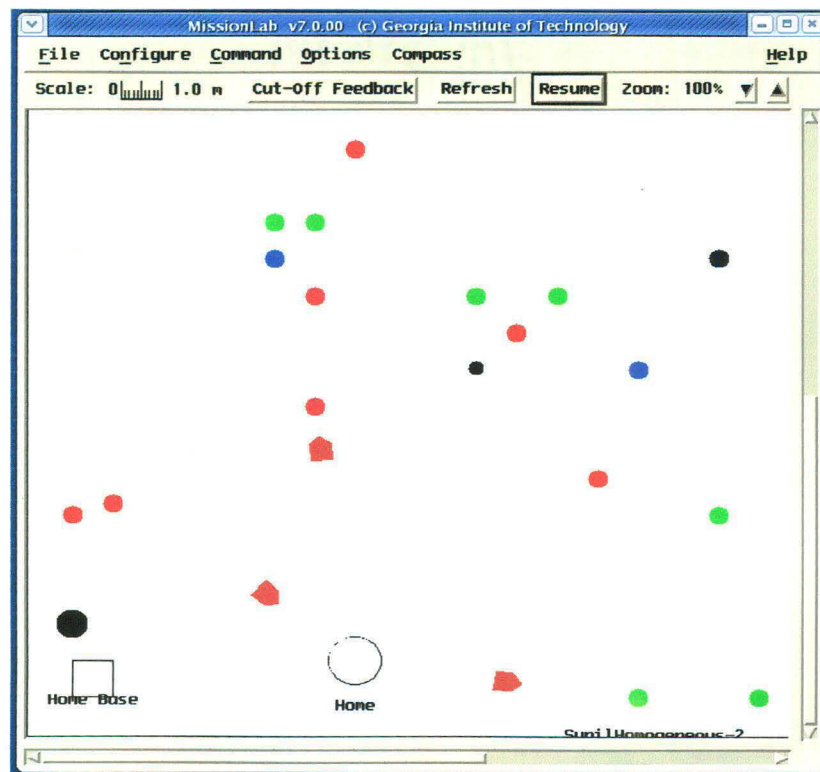


Figure 4.9: Execution of three Homogeneous Robots with single Home location.

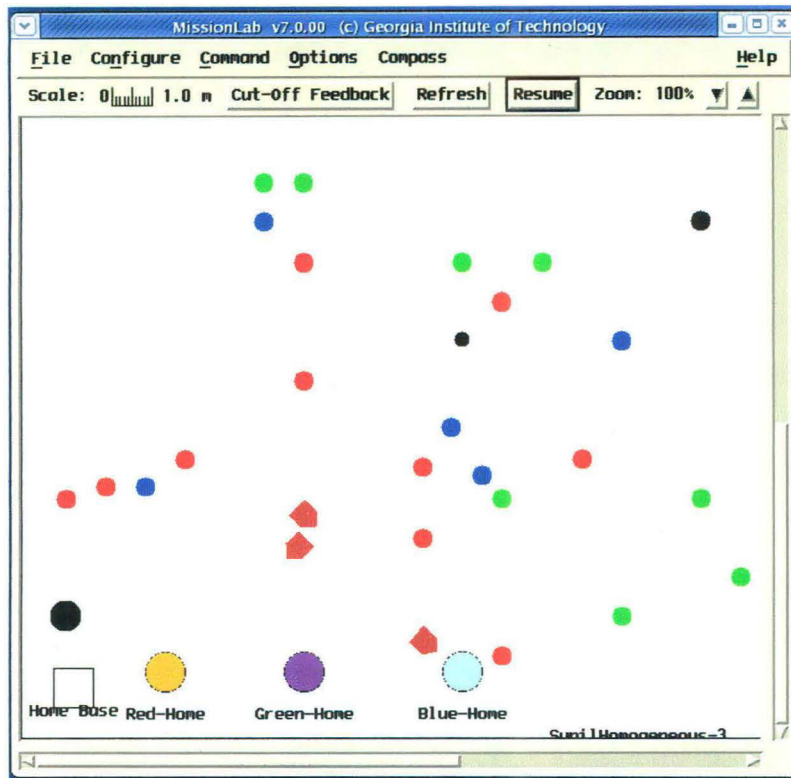


Figure 4.10: Execution of three Homogeneous Robots with three Home locations and different size objects.

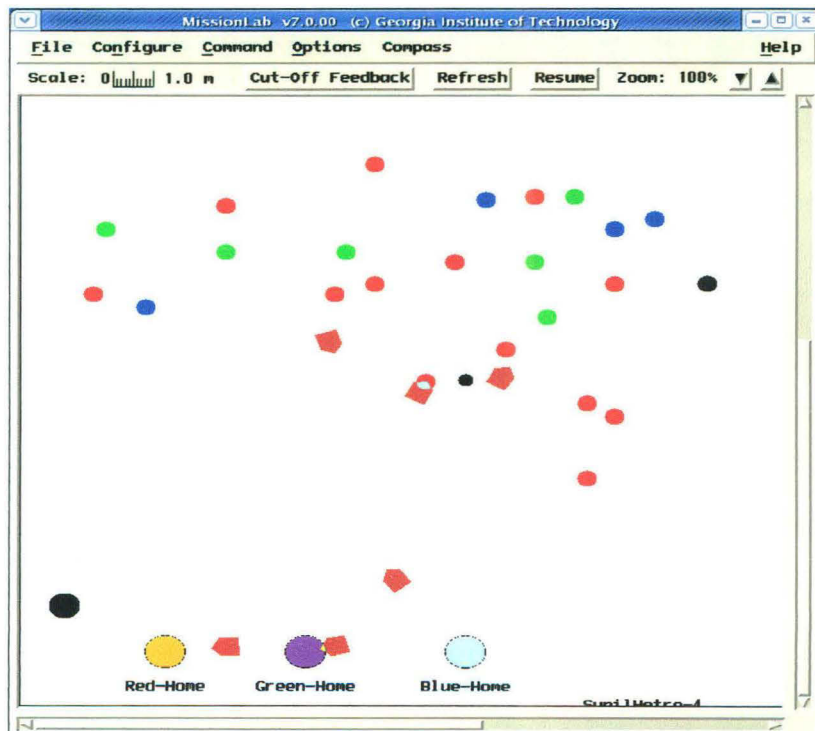


Figure 4.11: Execution of six Homogeneous Robots with three Home locations.

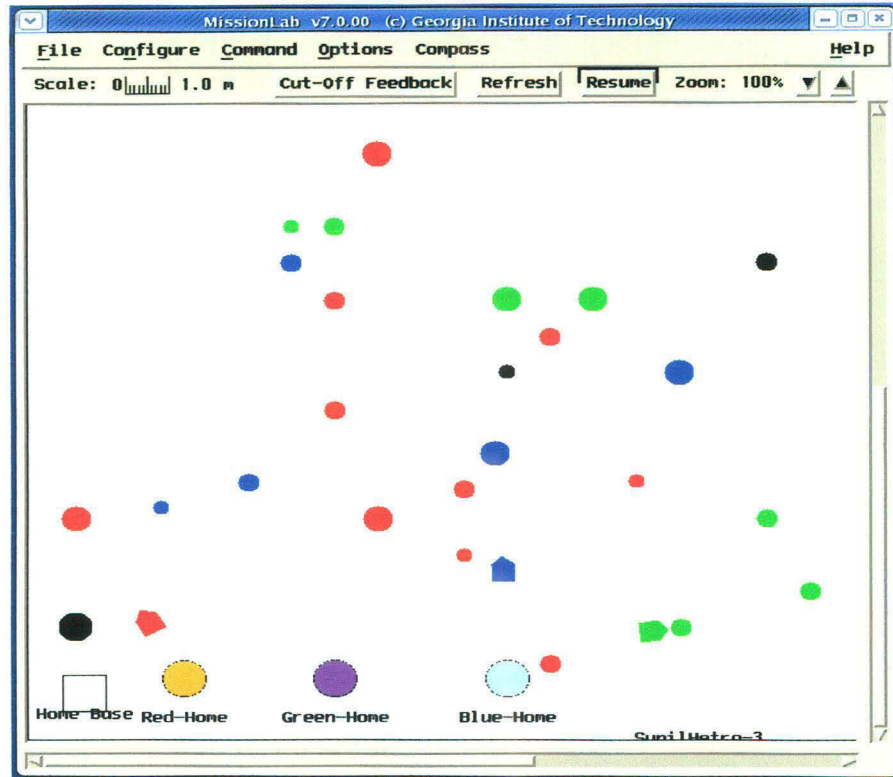





Figure 4.12: Execution of three Heterogeneous Robots with three Home locations and objects of different sizes.

In Figure 4.12, the we have following shape:

-  Represents Robot_{Red}
-  Represents Robot_{Green}
-  Represents Robot_{Blue}

In homogeneous group, robots start wandering for the pucks or objects. It moves towards the object that is under the range of sensor and it is closer than other objects. In case of single home location robots consider the entire objects as same type and they just pick the objects and drop at home(s) location(s). But in case of multiple home location robots detect the particular type objects and drop them on corresponding home location according to their types for example if robot detect red object, it pick and move toward and drop at Red_Home location.

This process will continue until there is more objects in the environment or all the objects are not on their home locations.

In heterogeneous group, robots are specialized for particular type of tasks. Robot_{Red} is expert to forage the red objects and Robot_{Green} is for green objects. The Robot_{Red} robots wander only for red pucks or objects and Robot_{Green} for green objects. For multiple home locations after holding the desired type object, robot start wandering to search the appropriate home for that object. The Robot_{Red} robot picks only red object and deliver it on corresponding Red_Home location. After dropping the object the robots move back to the environment.

4.3 Simulation Results and Analysis

Three important variables in the experiments were the number of robots in each group, the types of objects, and the number of home location.

4.3.1 Comparisons of Homogeneous and Heterogeneous Group

The performance of the dynamic task allocation in multi-foraging is examined experimentally. Several experiments are conducted to test the performance of framework of robot teams with different number of objects scattered in the environment in terms of total time to accomplish the overall mission for multi-foraging domain.

For the first experiment there are two groups of robots one is homogeneous group and the other is heterogeneous group consisting of two robots in each team. There are 20 objects of two colors, Red and Green, are scattered in 31×30 meter environment (as shown in Table 4.1 and 4.2). The robots search for objects and collect on the single predefined home location.

To measure the performance of the system, total foraging time is measured for each robot of both teams as shown in Table 4. 6.

Foraging Time (Seconds)			
Type of Group	Robot 1	Robot 2	Total
Homogeneous	68.132	61.205	129.337
Heterogeneous	69.55	68.473	138.023

Table 4.6: Foraging Time for team of size two for single Home

According to graph shown in Figure 4.13, we find that homogeneous team performs better than heterogeneous as team size is very small and single home location speeds up the execution of the mission execution.

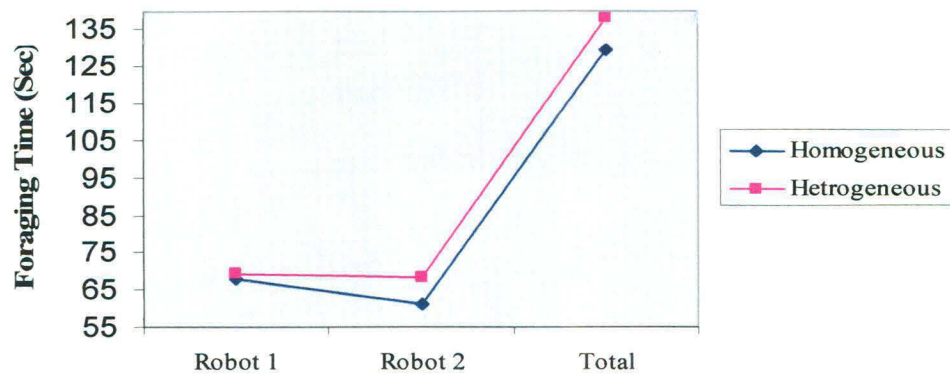


Figure 4.13: Comparison of Heterogeneous and Homogeneous Groups size of two

In the second experiment, the group size as well as number of objects and type of objects are increased. Three types (Red, Green, and Blue) of 28 objects with different size are scattered in 31×30 meter environment (as shown in Table 4.1 and 4.2).

To measure the performance of the system, total foraging time is measured for each robot of both teams as shown in Table 4. 7

Foraging Time (Seconds)				
Type of Group	Robot 1	Robot 2	Robot 3	Total
Homogeneous	57.29	53.52	49.79	160.60
Heterogeneous	40.16	65.64	58.70	164.50

Table 4.7: Foraging Time for team of size three with single Home

According to graph shown in Figure 4.14, we find that performance of the group is just like first experiment.

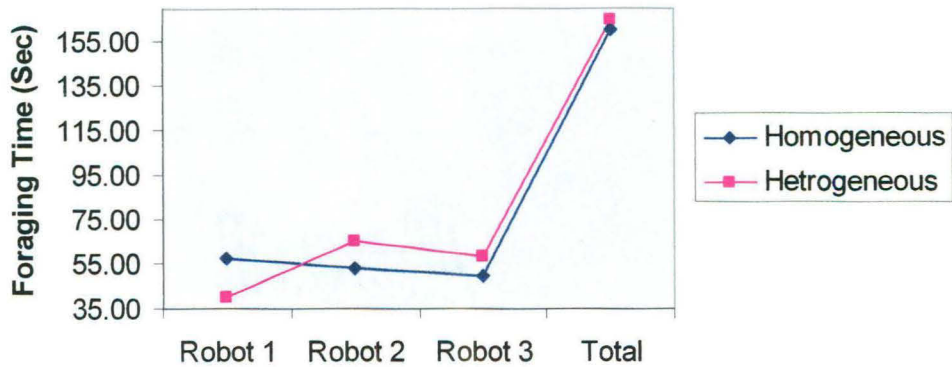


Figure 4.14: Comparison of Heterogeneous and Homogeneous Groups size of two.

The third experiment is totally based on multi-foraging domain that consists of multiple types of home locations beside multiple types of objects. Now we have three home locations, Red_Home for red objects, Green_Home for green objects, and Blue_Home for blue objects. Three types (Red, Green, and Blue) of 28 objects of different size are scattered in 31×30 meter environment (as shown in Table 4.1 and 4.2). The robots search particular type of objects and drop them on their corresponding home location.

Table 4.8 shows the time consumed by each robot to complete the mission.

Foraging Time (Seconds)				
	Robot 1	Robot 2	Robot 3	Total
Homogeneous	67.772	67.375	73.614	208.761
Heterogeneous	77.749	41.454	62.124	181.327

Table 4.8: Foraging Time for team of size three with three Home locations

By looking at the Figure 4.14 we can say easily that in Multi-Foraging domain, heterogeneous team performs better as compare to homogeneous.

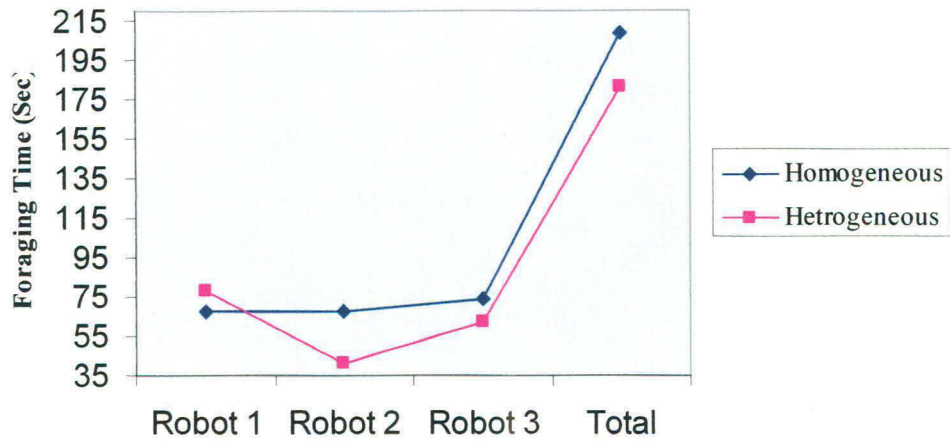


Figure 4.15: Comparison of Heterogeneous and Homogeneous Teams of size three with three Home locations

The experiment four involves more number of objects and robots than previous experiment. The group has six robots and environment has 54 objects of three types are scattered in same 31×30 meter environment (as shown in Table 4.1 and 4.2). The Foraging time of mission reading is given in Table 4.7.

Type of Group	Foraging Time (Seconds)						Total
	Robot 1	Robot 2	Robot 3	Robot 4	Robot 5	Robot 6	
Homogeneous	79.86	84.65	80.12	51.06	96.03	77.20	468.93
Heterogeneous	55.99	76.81	57.55	50.10	72.78	51.70	364.93

Table 4.9: Foraging Time for team of size six with three Home locations.

We can see clearly in Figure 4.15 that heterogeneous group of robots perform better than homogeneous group of robots as the team size increases. Homogeneous team tasks more time due to internal competition among the group members.

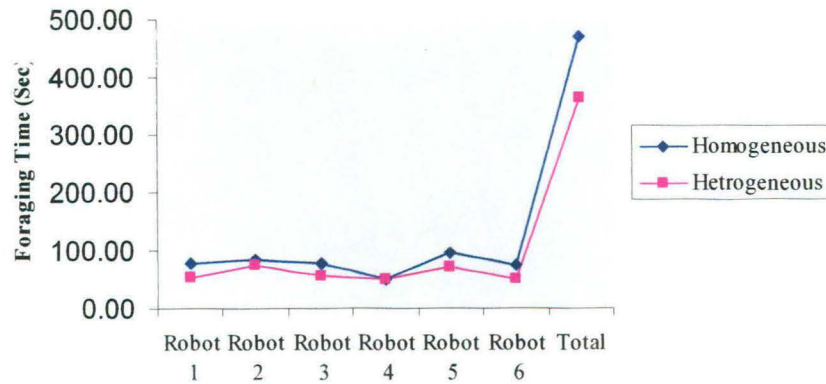


Figure 4.16: Comparison of Heterogeneous and Homogeneous Teams of size six with three homes

4.3.2 Interference in Homogeneous Group

Interference is caused when more than one robot tries to grasp the one object at the same time. This problem is more in homogeneous robots. Larger the size of the group, greater is the degree of interference between the robots. If the size of the environment is fixed and the size of the robot group is increasing, after an optimal team size the foraging efficiency of the system starts decreasing due to interference. Many missions are executed to demonstrate the interference in a group of robots.

In this experiment to observe the effect of the size of the group of robots on performance, the task allocation is applied on a group of robots of size varying from two to eight in a multi-foraging domain.

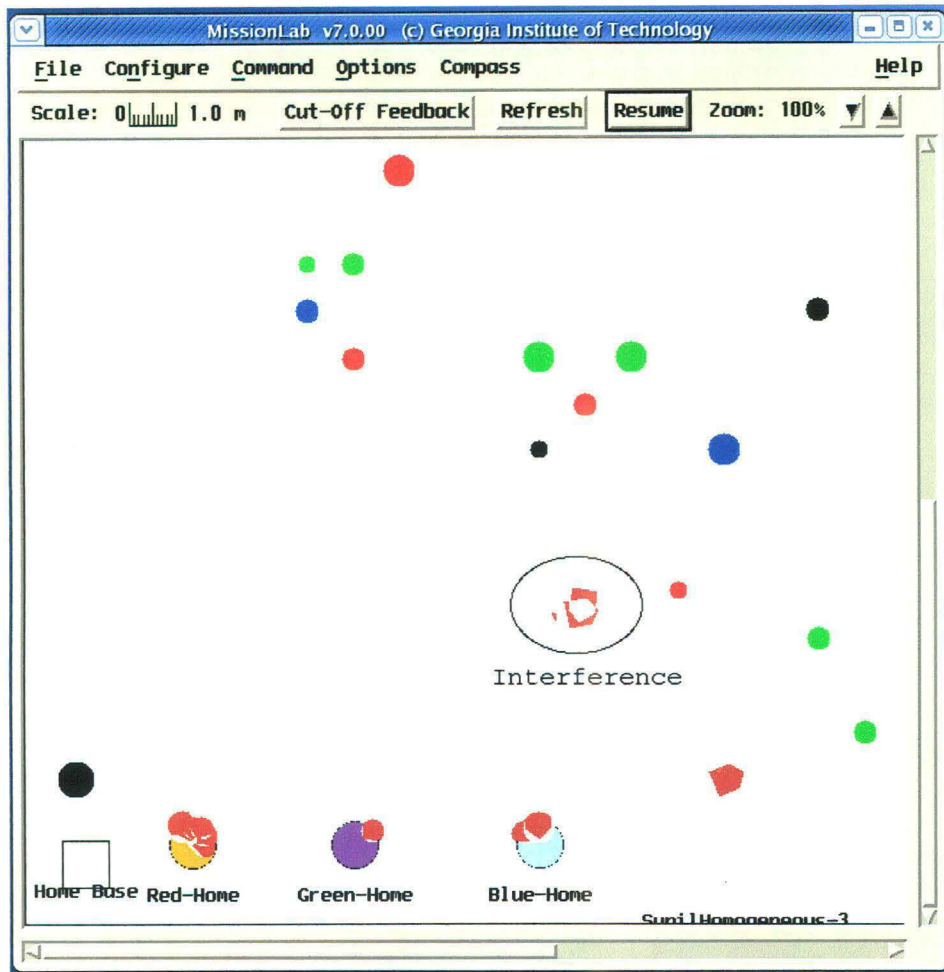


Figure 4.17: Interference during the execution of foraging task

The numbers of objects are 54 of three types as given in Table 4.1. Figure 4.16 shows the interference between two robots at the time of mission execution. Table 4.8 gives the different mission completion time for multi-foraging with different size of groups.

Homogeneous Team	
Team Size	Total Time
3	197.85
4	185.37
5	183.24
6	174.42
7	189.23
8	197.49

Table 4.10: Foraging Time (sec) for Homogeneous Team

In Figure 4.17, we can see that foraging time is decreasing from group size three to six. After six the time is starts increasing. The foraging time for group size seven is more than group size six and group size eight is greater than group size seven.

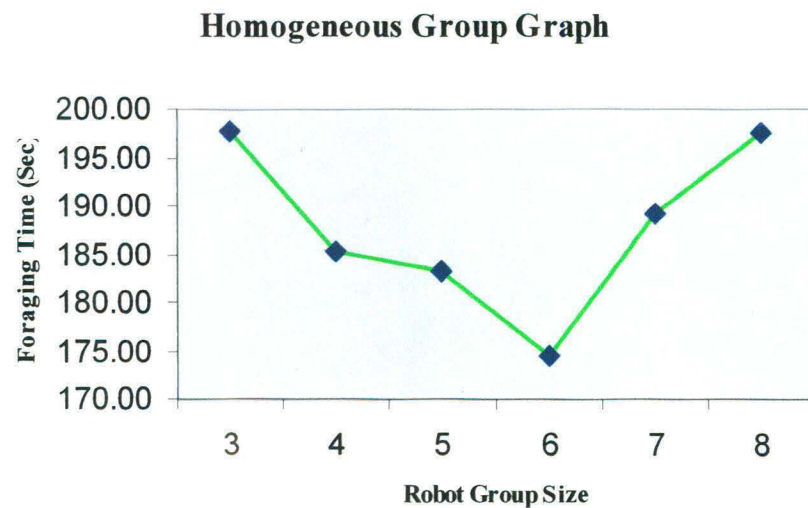


Figure 4.18: Foraging Time (sec) for Homogeneous group

In this experiment it is observed that group size **six** is an optimal size for the above environment for homogeneous robots.

5. Conclusion

In this dissertation a Dynamic Task Allocation mechanism is proposed for Multi-Foraging domain using groups of robots in dynamic environment. The multi-foraging scenario is analyzed for homogeneous and heterogeneous group of robots. This architecture is fully distributed at the level of individual robot and the group of robots. At the individual robot level, appropriate behavior is executed based on the behavior activation condition to perform a particular task. At the group level, each robot selects task individually based on the priority of the task without having any centralized control system.

A full implementation of proposed scheme is presented using multi-robot simulator MissionLab-7.0 developed by Georgia Tech Mobile Robot Laboratory. It is demonstrated by simulation how groups of robots can be used to produce dynamic allocation in the context of multi-foraging.

Several experiments are conducted with different team sizes, number of objects and object types both for homogeneous and heterogeneous robots. Simulation results are quite encouraging and establish the effectiveness of proposed scheme.

The current implementation employs up to eight robots, but the approach can easily be scaled to an arbitrary number of robots. The problem that we anticipate with large number of robots is the increasing amount of interference between robots that diminishes more to the performance of homogeneous team as compare to heterogeneous team of

robots. A heterogeneous team of robot is more comfortable with different types of tasks and performs better in large group as it is less prone to interference. The other problem is with the size of objects. The proposed algorithm is working well on objects of different sizes with the maximum object size limited to the capacity of single robot. If the object size is very large that can not be picked up by one robot then two or more robots would be needed. Therefore, further extension would be required to deal with the objects of larger size.

In future work we can extend this strategy to evaluate and analyze the priority of tasks based on other states and communication among robots with more complex scenarios so that by using the proposed algorithm, the system can be made more robust and intelligent.

Further work would be required to test and evaluate the system in real-world scenarios like mine finding and collecting in military operation, trash collection on the different locations, and other rescue operations etc.

A. MissionLab Overview

MissionLab is a powerful set of software tools for developing and testing behaviors for single robots and a group of robots. Code generated by MissionLab can directly control commercial robots. ATRV-Jr / Urban Robot (iRobot), AmigoBot / Pioneer AT / Pioneer 2DX (ActivMedia, Inc.), and Nomad-150 / 200 (Nomadic Technologies, Inc.) are among those robots MissionLab has supported successfully. A primary strength of MissionLab is its support of both simulated and real robots. A developer can experiment with behaviors in simulation and then run those same configurations on mobile robots (Figure A.1). MissionLab has a distributed architecture. Thus, the main user's console can run on one computer while multiple robot control executables are distributed across a network, potentially on-board the actual robots they control.

The core of the MissionLab tool-set is composed of six primary components:

- A.I **mlab**: mlab is a console-like program from which a user monitors the progress of experimental runs of the robot executables. Locations of the robots and detected obstacles are examples of various data mlab can monitor. When mlab is used for simulation (as opposed to controlling mobile robots), it serves as a sensor and actuator simulator from the point of view of the robot executable. On mobile robots, the actual sensors are used instead.
- A.II **CfgEdit**: The Configuration Editor, or CfgEdit, is a graphical tool for building robot behaviors. The designer can build complex control structures with the point and click of a mouse. CfgEdit generates source code which, when compiled, can directly control a simulated or real robot.

- A.III **CDL:** The cdl code generator translates the CDL (Configuration Description Language), which is generated by CfgEdit, into CNL (Configuration Network Language) code. In general, users will not need to be concerned with CNL. However, because programming in CNL is very similar to programming in the C language, advanced users may develop their own primitive behaviors and store them as a library and/or write their own control programs without using CfgEdit.
- A.IV **CNL:** The cnl compiler compiles CNL code generated by the cdl code generator, and produces C++ code. Once this C++ code is compiled with the GNU C Compiler (gcc), the compiled program (or robot executable) may now directly control a robot. The cnl compiler is automatically invoked by CfgEdit when needed.
- A.V **HServer:** HServer (Hardware Server) directly controls all the robot hardware, either via TCP/IP or a serial link, and provides a standard interface for all the robots and sensors. The CfgEdit generated code uses this standard interface to control the real robots. HServer also provides direct control, configuration, and status of the robots and sensors.
- A.VI **CBRServer:** CBRServer (CaseBased Reasoning Server) generates a mission plan based on specs provided by the user by retrieving and assembling components of previously stored successful mission plans.

In addition to CDL and CNL described above, there are two more original languages that were specifically developed for the MissionLab system:

- **CMDL:** The Command Description Language (CMDL) may optionally be used for describing simple sequential robot missions. A CMDL file, containing both background and command information, will be read by mlab at runtime and offer a mechanism for providing high-level input to robot behaviors developed in CNL. When robot executables are directly created by CfgEdit, users may not need to use CMDL.
- **ODL:** The Overlay Description Language (ODL) provides descriptions of the environment, which mlab can graphically translate to a map or a floor plan of an experimental area. For example, a robot's starting point, obstacles, boundaries, are among those features that ODL can describe.

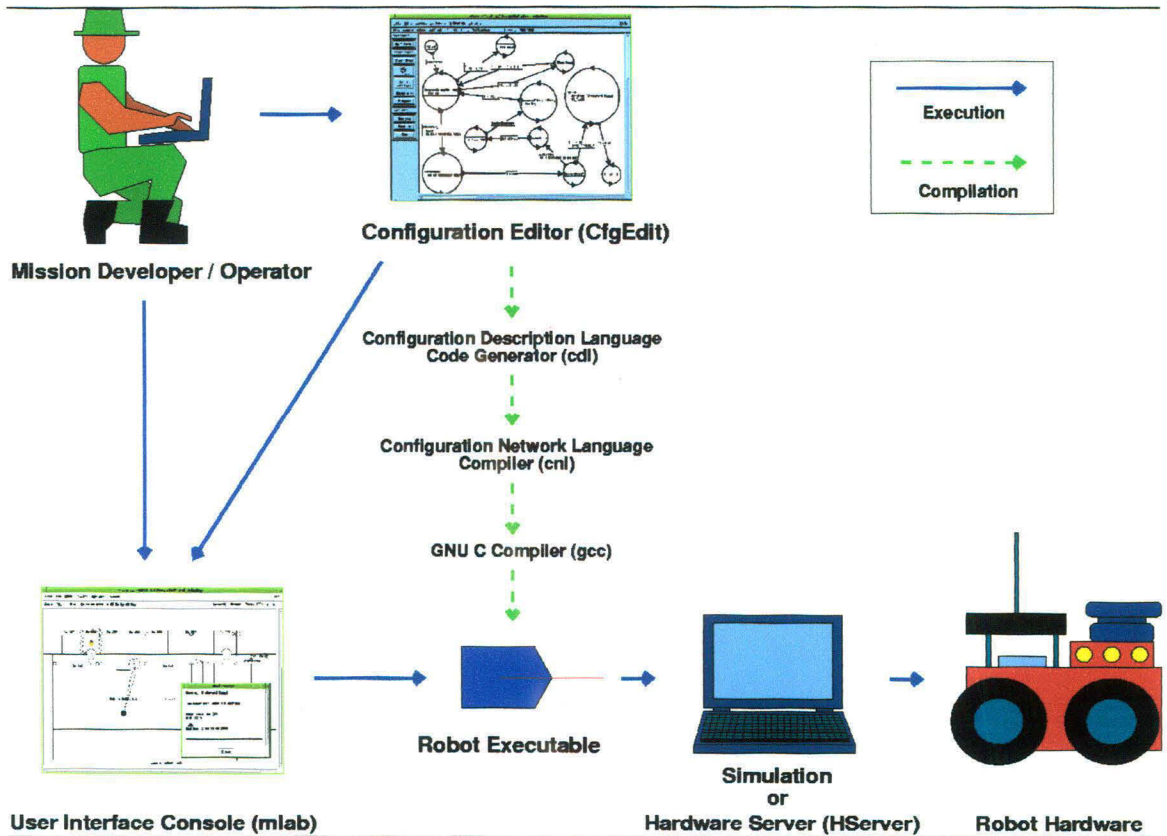


Figure A.1: MissionLab Components: the mission developer can create a mission for a robot or a group of robots with the Configuration Editor (CfgEdit). Once the mission is created, it can be compiled as a robot executable, which will give commands to the robot hardware or its embedded low-level software via Hardware Server (HServer). The robot executable is executed by the User Interface Console (mlab), and mlab can be invoked by both CfgEdit and the operator. mlab can also run a simulation before the actual real robot run.

For More detail Please refer to website:

<http://www-static.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab/>

REFERENCES

- Alessandro Farinelli, Luca Iocchi, Daniele Nardi (2004), "*Multi-Robot System: A Classification focused on Coordination.*" IEEE Transactions on Systems Man And Cybernetics Part B Cybernetics, Volume 34, Number 5, Pages 2015-2028.
- Beckers R., Holland, O. E., and Deneubourg (1994), J. L., "*From Local Actions to Global Tasks: Stigmergy and Collective Robotics*". In proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems ArtificialLifeIV , Cambridge, MIT Press, MA, USA, Pages 181-189.
- Brian P. Gerkey (2003), "*ON MULTI-ROBOT TASK ALLOCATION*", Center for Robotics and Embedded Systems, University of Southern California, Los Angeles, *CRES Technical Report CRES-03-012*.
- Brian P. Gerkey and Maja J Mataric (2003), "*Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architecture*". In Proceeding of IEEE International Conference on Robotics and Automation (ICRA 2003) Taipei, Taiwan (Also Technical Report CRES-02-005), Pages 3862-3867,
- Brian P. Gerkey, Maja J Mataric (2001), "*Principled communication for dynamic multi-robot task allocation*", In *Experimental Robotics VII, LNCIS 271*, D. Rus and S. Singh, editors, Springer-Verlag Berlin Heidelberg, Pages 353-362.
- Brian P. Gerkey, Maja J Mataric (2004), "*A formal analysis and taxonomy of task allocation in multi-robot systems*", International Journal of Robotics Research 23(9): Pages 939-954.
- Dimitri P. Bertsekas (1989), "*Auction algorithms for network flow problems: A tutorial introduction*", Computational Optimization and Applications, Volume 1, Pages 7-66.
- Ding Yingying, He Yan, Jiang Jingping (2003), "*Multi-Robot Cooperation Method Based On The Ant Algorithm*", Swarm Intelligence Symposium, SIS '03. In Proceedings of IEEE, Pages 14-18.
- Eiji Uchibe, Tatsunori Kato, Minora Asada, Koh Hosoda (2001), "*Dynamic Task Assignment in Multiagent/Multitask Environment based on Module Conflict*

- Resolution*”, Robotics and Automation, In Proceedings 2001 ICRA. IEEE International Conference, Volume 4, Pages 3987-3992.
- Esben H. Østergaard, Maja J Mataric, and Gaurav S. Sukhatme (2002), “*Multi-robot Task Allocation in the Light of Uncertainty*”, In *Proceedings, IEEE International Conference on Robotics and Automation (ICRA-2002)*, Wasington DC, Pages 3002-3007.
- Goldberg D. and Maja J Mataric (2000), “*Robust Behavior Based Control for Distributed Multi-robot Collection Task*”, USC Institute for Robotics and Intelligent System, Technical Report IRIS-00-387.
- James McLukin and Daniel Yamins (2005), “*Dynamic Task Assignment in Robot Swarms*”, Proceedings of Robotics: Science and Systems.
- Khashayar R. Baghaei, Arvin Agah (2002), “*Task Allocation Methodologies for Multi-Robot Systems*”, The University of Kansas, Lawrence, Kansas – 66045, Technical Report ITTC-FY2003-TR-20272-01.
- Kristina Lerman, Chris Jones, Aram Galstyan and Maja J. Mataric (2006), “*Analysis of Dynamic Task Allocation in Multi-Robot Systems*”, Int. Journal of Robotics Research, 25(3), Pages 225-242.
- L.E. Parker (1997), “*L-ALLIANCE: Task-oriented Multi-Robot learning in behavior based system*”, Journal of Advance Robotics. Volume 11, No. 4, Pages 305-322.
- L.E. Parker (1998), “*ALLIANCE: An Architecture for Fault-Tolerant for Multi-Robot Cooperation*”, IEEE Transaction on Robotics and Automation, 14(2): Pages 220-240.
- L. E. Parker, Claude Touzet and Fernando Fernandez (2002), “*Techniques for Learning in Multi-Robot Teams*” A K Peters, Pages 191-236.
- Lerman K; Galstyan A (2002), “*Mathematical Model of Foraging in a Group of Robots: Effect of Interference*”. Source: Autonomous Robots, Volume 13, Pages 127–141.
- Luca Iocchi and Deniele Nardi, Maurizio Piaggio and Antonio Sgorbissa (2003), “*Distributed Coordination in Heterogeneous Multi-Robot System*”, Autonomous Robots 15, Pages 155-168.
- Maja J Mataric (2001), “*Learning in Behavior Based Multi-Robot Systems Policies, Models and Other Agents*”, Cognitive Systems Research, special issue on Multi-Disciplinary Studies of Multi-Agent Learning, Ron Sun, ed., 2(1), Pages 81-93.

- Michael J. B. Krieger, Jean-Bernard Billeter and Laurent Keller (2000), “*Ant-like task allocation and recruitment in cooperative robots*”, Nature 406 (31), Pages 992-995.
- Mike Campose, Eric Bonabeau, Guy Theraulaz and Jean-Louis Deneubourg (2001), “*Dynamic Scheduling and Division of Labor in Social Insects*”, Adaptive Behavior, 8(2), Pages 83-92.
- Natsuki Miyata, Jun Ota, Yasumichi Aiyama, and Tamio Arai (1999), “*Real Time Task Assignment for Multi-Robots- Application to Cooperative Transportation Task*”, In proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems, Pages 1167-1174.
- Nitz, E., Arkin, R. C., and Balch, T. (1993), “*Communication of Behavioral State in Multi-agent Retrieval Tasks*”, In proceedings of the 1993 IEEE International Conference on Robotics and Automation: Volume 3, Pages 588-594.
- Peter Stone and Manuela Veloso (1998), “*Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork*”, In Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98), Artificial Intelligence (99), Volume 110, Pages 241-273.
- Sanem Sariel, Tucker Batch, Nadia Erdogen (2006), “*Robust Multi-Robot Cooperation through Dynamic Task Allocation and Precaution Routines*”, The International Conference on Informatics in Control, Automation and Robotics (ICINCO), Pages 196-201
- Shervin Nouyan (2002), “*Agent-Based Approach to Dynamic Task Allocation*”, M. Dorigo et al. (Eds.) : ANTS 2002, LNCS 2463, Springer-Verlag Berlin Heidelberg, Pages 28-39.
- Tucker Balch (1999a), “*The impact of diversity on performance in multi-robot foraging*”. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, Seattle, WA, Pages 92-99.
- Tucker Balch (1999b), “*Reward and Diversity in Multirobot Foraging*”, IJCAI-99, In Proceedings of the “Agents Learning about, from and with other Agents” Workshop.