The Pennsylvania State University

The Graduate School

Electrical Engineering Department

DEVELOPMENT OF A MATHEMATICAL HUMAN–SENSOR MODEL UTILIZING MULTIAGENT BAYESIAN LEARNING FOR TEAM COLLABORATION IN EXTREME EMERGENCY OR DISASTER EVENTS

A Thesis in

Electrical Engineering

by

Hemant Kumar

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2010

The thesis of Hemant Kumar was reviewed and approved* by the following:

David Hall
Professor
College of Information Sciences and Technology
Thesis Advisor

Tracy Mullen
Associate Professor
College of Information Sciences and Technology

Kenneth Jenkins
Professor and Head of Department
Electrical Engineering

*Signatures are on file in the Graduate School

ABSTRACT

In this thesis a probabilistic mathematical model human sensor software agent has been illustrated followed by interaction with a team of heterogeneous multiagent team of human and normal mobile or immobile sensors deployed in the field for tracking a target. A Hierarchical Agent based Architecture based on Distributed Bayesian Network employing Distributed Data Fusion to mobilize resources (vehicles, security personnel, arms and ammunitions) towards the main region of terrorist actions has been proposed. A description concerning the working of the above agent system is provided along with mathematics involved in fusing heterogeneous data from both hard sensors (camera sensor, vehicular based mobile sensors) and humans modeled as information sources. A Human Perception Probabilistic model (HPM) is being used to provide a common framework for heterogeneous data streams coming from both types of sensors. Information from cyber sources like Twitter and other social networking sites like Facebook, etc has not been considered, though such fusion will greatly enhance the performance of the rescue operations mobilization. Details of the techniques pertaining to identification of particular features of the anti-elements are not the scope of this research but the features and gestures which can help in identifying them has been considered. This thesis details hierarchical heterogeneous human and sensor model, emphasizing on the multi-agent learning involved in team trying to work in a coordinated way in a disaster response operation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my thesis advisor Dr. David Hall and to Dr. Tracy Mullen, for their valuable suggestions, guidance and constant encouragement. I thank my parents who constantly encouraged me throughout my masters at the Academy. I thank my brothers, sisters and friends for their constant encouragement and support.

Chapter 1

Introduction

1.1 Introduction to Multiagent Systems

A multiagent system is a system of interacting multiagents agents responding to their immediate environment depending on their perception. A multiagent system can be used to model large decentralized systems. These multiagent systems are currently used to model application areas as diverse as large social media architectures, eBay auctions and target tracking applications. Consequently scalable multi-agent systems technology is becoming increasingly important, and multi-agent research is a lively and growing area facing many challenges. In particular, the inherent dynamism in many of these problems calls not for offline computations of solutions to problems, but rather timely online responses to new unknown sciences.

The proposed architecture here is used for tracking targets in an urban environment and identifying a major catastrophic event such as a terrorist attack on an urban city based on observations from sensor data acquisition system deployed in the field and human and reports sent by human agents prevalent in the field. The human agents have been modeled as software proxy agents which characterize the real field agents based on their individual fitness and skills.

In more detail, the agents working in such unknown scenarios will frequently be uncertain, both about the current environment and about the behavior of other agents. Specifically when the agents are not able to see all aspects of their current situation, the scenario is described as partially observable. In such a setting the agents must carry out a discovery phase to learn about the scenario before they can focus on their individual goals and collaborate among themselves for

taking robust and most optimum actions. Agent discovery phase happens with exchange of messages between various individual agents either by wireless, wireline or a local central agency mediating between them.

In a multiagent system, an agent is always acting in the context of other agents, and so it must adapt its plan according to the expectation of others. This needs to take into account coordination which is a key issue in multiagent system. In particular, in uncertain and open systems the protocols for coordination must function against a background where agents are not fully aware of the situation, the resources available to them or the presence of goals of other agents. Integrating the behavior of individual agents with the heterogeneous team is done on the basis of weight factors attached with the quality of observations reported by the agents. The coordination clubbed with uncertain behavior of agents makes it a tough paradigm for integrating a new type of human sensor agents in the hierarchical and heterogeneous multiagent architecture as proposed here.

1.2 Overview of Existing Multiagent Systems

In an agent system, an intelligent agent is functioning in a dynamic environment. This environment provides stimulation to the agents' senses to which the agent responds by acting on the environment. If the system is a multiagent system, then many agents coexist in the same environment, and the actions of one agent can cause perceptible changes in another's environment. Such multiagent systems are becoming increasingly prevalent as a result of a number of significant trends in modern technology [21] .

Some of the main characteristics of multiagent systems which are utilized in decentralized data fusion architectures are:

- **Ubiquity:** As computing chips become smaller and smaller, it is possible to add computational power and intelligence to many kind of devices in almost any location. Systems made of networks of these ubiquitous devices offer much greater possibilities than individual devices.
- **Decentralization:** With the advent of World Wide Web and other computer networks such as grid computing and peer-to-peer networks systems that distribute data and tasks among networks of machines are increasingly common.
- **Openness and dynamism:** Open systems are those in which agents may enter or leave at any time, while in dynamic systems the environment properties may change at any time. Many real world systems are both open and dynamic and there has been a corresponding trend in computing towards providing interactive systems which are able to respond to changing environment.
- **Uncertainty:** Uncertainty plays a large role in systems which respond to environmental or sensor inputs. Moreover, a trend towards increasingly large and complex systems means that frequently systems are effectively uncertain even if they are technically deterministic. In case of human agents, eliminating uncertainty present inherently in their observations is a real challenge.

The combination of these features describes a broad class of complex, dynamic, large scale systems which may be implemented or modeled as multi-agent systems. Along with the characteristics mentioned above multi-agent systems may be heterogeneous in nature containing nodes with a variety of capabilities and goals representing different nodes in the system. For example such a heterogeneous sensor network might contain pressure, temperature and camera sensors along with human agents deployed in the field presenting the central data fusion engine with continuous information of their observational data streams. The objective of every active

centralized agent in such a network which governs a group of heterogeneous nodes is to make an informed inference or decision based on observation reported which is bound to be uncertain in nature. The central fusion node has to make optimal decisions under uncertainty and facilitate the agent coordination to achieve the objective in predicting the correct event and making right moves in case a particular target needs to be tracked.

Before describing the hierarchical heterogeneous multiagent architecture which consists of human proxy agents and other sensors agents, a scenario is introduced below which can encompass all the characteristics required for multiagent research like disaster emergency response operations or terrorist attack inciting panic and mayhem.

The importance of such a scenario stems from the fact that we increasingly encounter such disasters for instance, Asian Tsunami in 2004 Terrorist attacks on WTC 2001, Hurricane Katrina in 2005, and most recently the Haiti Earthquake and other earthquakes happening quite frequently in 2010. Reaction to all of these scenarious calls for an effective coordination between heterogeneous team consisting of humans and sensor agents.

In this work, a terrorist attack on an urban city has been simulated, and we focus on how a hierarchical heterogeneous agent team works to identify the various events, confirms the main event and finally launching an overall rescue and search operation.

1.3 Disaster Scenario Description

The disaster scenario described here is very typical but it contains all the nuances and characteristics which require the implementation of a general multiagent system applicable in every disaster emergency situation. The implementation of such systems can be fully automated with no human element involved or it can have humans working as sensors providing data streams as well as regulating the various other sensor nodes. The focus of this work is to

demonstrate the impact of optimal coordination and decision making ability of humans involved in disaster response operations both as a passive supplier of information and an active agent governing other sensor nodes.

The utility of such an operational multiagent system becomes apparent considering the ubiquity of variety of smart phones with powerful features which can be used to transmit multimedia data comprising of text messaging, photos and videos for statistical analysis and fusing it with other data streams from hard sensors like pressure, temperature, camera and speed sensors. In this work, the primary focus is on modeling the human agents as software proxies and then introducing them in a multiagent model and demonstrating the dynamic interaction of these agents with other agents in the system aiding in proper coordination to achieve the objectives at hand rather than focusing on techniques of fusing the data streams from humans, i.e., soft sensor agents and hard sensor agents. The scenario is explained as follows:

A metropolitan city assumed to be under attack by a group of terrorists and numerous multiple events are occurring in various parts of town. The numerous events may or may not be related to the attack. The coordinating agent at the top of hierarchy in a sub-region of the town is getting numerous feeds of information from camera sensors, temperature sensors, mobile robotic agents and human observers . The coordinating agent must make a decision on the confirmation of that sub-event and relay to a higher level agent governing it. A typical scenario involves occurrences of small fires, smoke emanating from the fires, newly damaged buildings, and roads due to explosion, sound of gun firings, bomb explosions etc. and release of poisonous gases. The authenticity of events happening needs to be also verified before a concrete decision is made of the sub-event. These events can also be confirmed from cyber agents who report sightings of original images from the scene of action. For instance, if there is a fresh reports saying that a particular building has been damaged, then the current image can be matched up with the current

stored image to confirm the damage. The confirmation of major event is based on Bayesian learning by the multiagents. For instance, if there is a series of observations e_1, e_2 , e_3…….e_n and so then, a major event can be confirmed if weighted summation of all these events crosses a particular threshold.

The goal of the hierarchical heterogeneous multi-agents is to confirm the overall event, i.e., a major terrorist attack, as well as mobilize resources like ambulances, fire-fighting professionals, trucks with equipment, as well as while mobilizing and warning other teams about the blocked roads, and dangers in the other region so as to facilitate speedy recovery from the damage, protect as many lives as possible by carrying the injured to hospitals and dousing building fires , clearing the rubble and debris of collapsed to accelerate the rescue operations. So based on this, a reward function has been formulated whose main objective is to minimize the damage and maximize the saving of lives and at the same time expediting the rescue operation by better coordination among decentralized teams. The decentralization of teams needs to be considered as in an event of disaster or attack. For example, it may happen that the network gets blocked due to panic among local populace and so abruptly the network traffic goes high. For handling this, in the appendix section of this thesis, a mechanism to integrate 3G cellular infrastructure which gets slowed down due to network congestion with 802.11 based wireless LAN link has been demonstrated.

Now taking this complete disaster scenario as an illustrative domain, the properties mentioned above for a scalable multiagent systems [21] will be described.

Decentralized: As explained above in such an emergency rescue operation multiple teams with heterogeneous agents will be operating in parallel. There is no centralized commanding agency. Instead, there are multiple sub-regional agents giving instruction to these teams based on individual sensor capabilities and feedback provided by various agents. So, there is

decentralization in the operating nature of multiagent system which needs to operate in such a paradigm.

Dynamic: The conditions in such an environment in the wake of a disaster or recovery operation keeps on changing with further collapsing of buildings, more reporting of fires, leading to blockage in the functional routes, changing weather conditions and many more. All this needs the central coordination agency which is having the overall view of the geographical area to coordinate the movement of teams through safe routes and diverted to where maximum casualty has happened.

Open: Such a large scale disaster rescue and search operation will be functional over a large geographical region, and so there will be entry and exit of human agents. Some may simply vanish because of casualty and more volunteers may join the team from outside.

Uncertain: In a disaster scenario, it is highly unlikely that one agent will have a complete view of the situation. Moreover, information which reaches the agents may be error-prone, increasing the uncertainty. At a different level of granularity, environmental conditions such as weather conditions and working conditions are uncertain at every moment of the operation.

Heterogeneous: As explained earlier, the teams working in such a disaster rescue operation will be consisting of agents, hard soft or cyber agents, which have different set of capabilities and goals which may be common or in conflict. At the minimum, there will be rescue teams with distinct tasks: ambulances, police, helicopters teams, ground observers who just report events and there will be people affected by the disaster.

Bandwidth: Due to vast amount of messages and information being exchanges among the various agents at different levels, this will lead to blockage of the communication networks in the field. This requires the various agents to exchange information on different frequencies in such a Sensor Ad-hoc network. One of the sections in this thesis addressees the integration of 802.11

Wireless LAN based orthogonal frequency division multiplexed sensor ad-hoc network integration with the existing 3G CDMA EVDO based infrastructure as mentioned earlier.

Large: Such a disaster recovery operation will be over a large region involving hundreds or thousands of team agents, organizations operating over a large area. Considering this the solution should be scalable and flexible enough to include or exclude agents as they enter the area of interest or leave it.

1.4 Overview of Hierarchical Hybrid Multi-Agent Architecture:

The various types of agents viz. human agents, sensor agents and software agents are distributed over four layers as shown in the figure below in Figure 1.1. Although this working of this architecture can be used observation and confirmation of terrorist activities over a region, it can be used for observation of any distributed activity and correlation among them using Bayesian logic to arrive at a particular conclusion.

The urban region has been divided into square blocks. Each square block comprises a number of human (red) agents and sensor agents (yellow). Human agents are mobile in nature, and hard sensors can be fixed as well as mobile. Every square region has a coordinator node (fixed human sensor node) which acts as central agency for that square region and handles the configuration of Bayes Net in that region under governance of first tier software agent (FTSA).

FTSAs are the first point of information sink in the software from the real time environment and so they process the heterogeneous information streams from both human sensor coupled with uncertainty and hard sensors which almost give accurate information. The purpose of creating such software agents is to reduce the computation load from humans nodes HPM. This reduces

the workload on them besides regulate the mobilization of mobile sensors in different regions. For instance if FTSAs have a majority of reports emanating from a particular area then they have to give instructions to mobile sensors to that region. This type of allocation of resources happens only after taking into consideration the trade off involved between the benefits obtained by mobilization of resources and by value/cost of information associated with it. In addition to this FTSAs also resolve the resource conflict if any between two human sensors querying and commanding a single sensor to take some action. In case there is no coordinator node in any particular region, then it assigns the neighboring coordinator who is in the least activity state to take control of nodes present in that region.

Every time a new mobile agent enters a new region it signups with the coordinator which then in turn informs FTSA. The FTSA creates an instance of HPM for the human registering itself with the network. This HPM will be used to process any raw observations submitted by the human based sensor source. In this way the sub network of agents in every square region is configurable. Every coordinator node tries to make a probabilistic guess about the state of event in its region and informs the FTSA.

Above the layer of FTSA is the second tier of software agents (STSA). STSAs take information from two or more FTSAs and fuse them together again using Bayesian logic to arrive at some conclusion and identify the events happening at the ground level. They serve to coordinate a group of FTSAs and regulate the coordination between them allocating them regions based on the workload and intensity of operation. For instance, for a region having a high level of activity the distribution of FTSAs has to be uniformly made so as to take the load off a single FTSA operating for that region initially.

Finally all the STSAs give their inputs to the top level root node which based on Bayesian logic again arrives at a decision regarding the overall ground situation. Besides this it coordinates the working of the STSAs and turns them off/on.

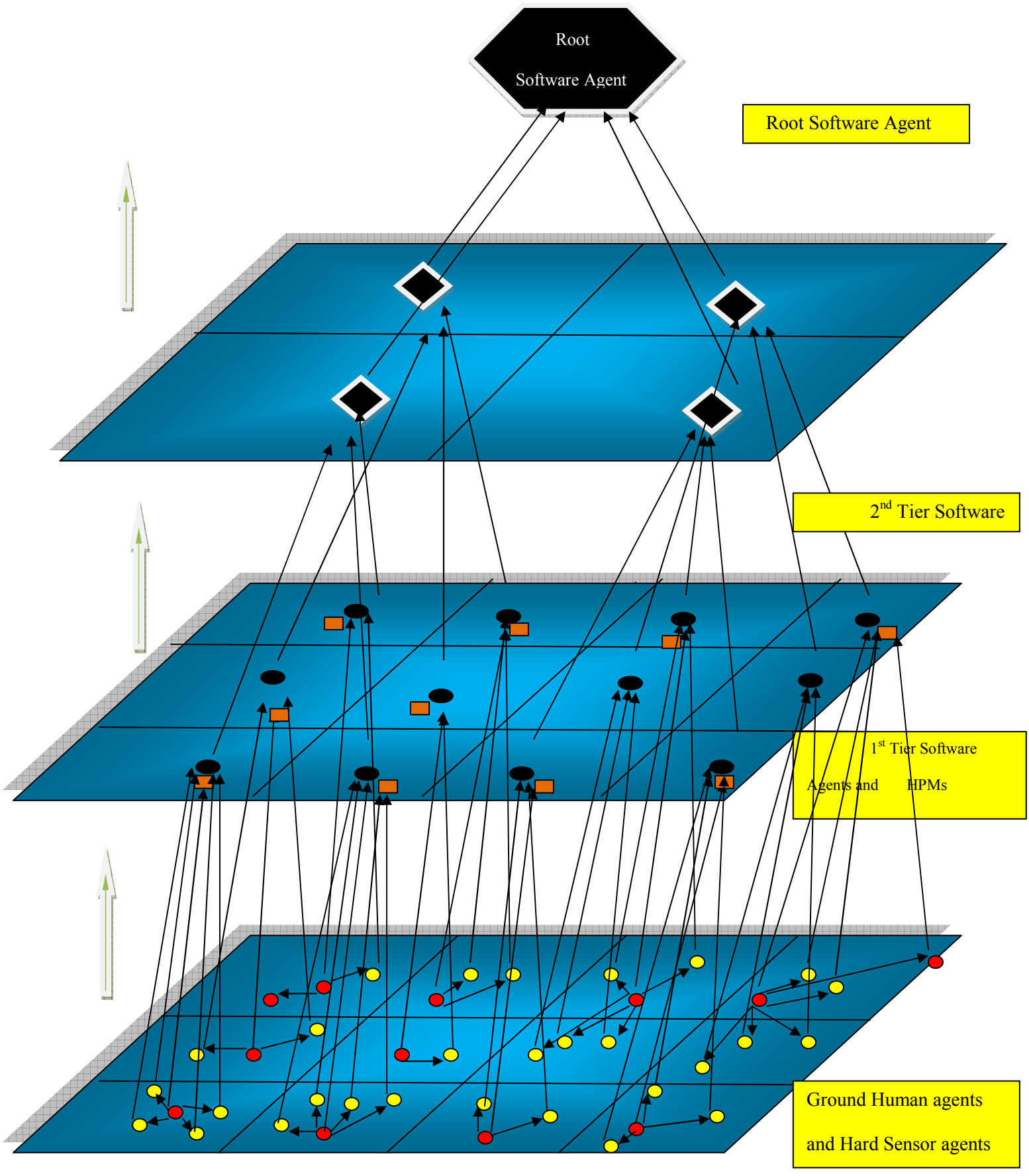
Basically the STSAs help in formation of local beliefs about the phenomena which is happening at the ground level, and all these local beliefs are then fused together to form a global belief in a step by step fashion until all this information reaches moves upwards to the top level root node.

Humans operate both as information sources as well as sink in the above model. They also act as an authority for controlling the mobile hard sensors at the ground level. So it becomes necessary to develop a probabilistic model for them. This model forms a part of the FTSAs for accepting information from them and fusing them with information obtained from other hard sensors on the other interface.

The Information flow model in this architecture can be of four types:

- Human Push: Humans voluntarily providing information about events going in their immediate surroundings. Information flows from Human to FTSAs.
- Human Pull: Humans querying and tasking the sensors. Information flows from sensor to Humans.
- FTSAs Push: FTSAs commanding humans and mobile sensors for mobilizing them to the point of action. These actions are taken by FTSAs voluntarily.
- FTSAs Pull: Humans querying FTSAs and so the information flows from FTSAs to human sensors.

Figure 1.1 Hybrid Hierarchical Multi-agent model



1.5 Thesis Background and Literature Survey

This thesis is concerned with combining the perceptual abilities of human operators in the form of observers, rescuers, fire personals and all types of human agents present at the site of disaster rescue operation with the deterministic data streams generated from hard sensors in the form of pressure, temperature and other electronic gadgets. The combination of these multiple streams of data is then utilized to make an informed decision to drive the agents towards achieving their goals and making them aware of other agent action so as to maximize the output of rescue efforts. First of all, an effort was made to study the tools and techniques used in developing such a probabilistic mathematical model for human agents in software [1][2][3]. For this in the first phase of research work, I studied the basic techniques like Bayesian Network Model, Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) modeling, Inference Diagram approach and Casual Bayesian Network design. In the next phase, a study of the work on integrating humans in sensor networks was done. Some of the relevant work which describes Human in the loop experiments are ([7] [8] [14]) . Despite the fact that perception of the environment is a critical element in performing any task, cooperative multiagent learning [4] is essential component to be incorporated in framework for effective team coordination for rescue operation. In this research work, specifically Bayesian Agent learning has been utilized along with MDP and POMDP to model the disaster scenario and then probabilistic analysis has been done to formulate the multi-agent learning.

For exploring on uncertainty involved in human observation and for modeling the human perception model described in chapter 3, [10] [11] [12] are relevant. Although, this literature dwells upon probabilistic reasoning by humans, a concrete interface for modeling software agents which can represent humans on ground has been not been described. This thesis incorporates a reasoning engine and introduces a human perception model which when trained using actual human data can be used to predict the best action in collaboration with other heterogeneous

agents. Besides this, a study was made on research already done in the field of human assisted team collaboration for target tracking in urban warfare scenario [17]. The ubiquitous use of PDA phones, handheld image capture and texting devices along with remote controlled data stream generating devices used for wireless surveillance system have been used for collaborative multiagent action in some of the research work [13], [14], [18] .

Chapter 2

Human Perception Models (HPM)

2.1 - A Probabilistic Human Mathematical Model encompassing Uncertainty measure.

The concept of treating humans as information sources or decision making agents as part of probabilistic reasoning and decision making engine in any complex multi agent system implementation is a novel contribution the field of traditional Joint Director of Laboratories (JDL) data fusion model [22].

The potential benefit of this concept has recently gained attention in the context of crisis management where people in the field may be able to contribute valuable information after natural or man-made disasters have occurred. Another application area where operators may add information content to a decision making or reasoning system in crisis management in command and control module when based on observation reported by another users, and human observers apart from data streams reported by sensor units, an alert human can take the most optimum and informed decision for next course of action.

Before delving into the modeling of human modeled as sensor, the next paragraph discusses the differences between humans behaving as sensor and usual sensors assembly motes deployed in wireless or wireline sensor networks which motivates the implementation of probabilistic model for humans.

Abstraction Hard sensor assembly units perform well in low-level descriptions such as geometric properties. Measuring the range and bearing to a point using a laser scanner is an example. In contrast a human observer is valuable in contributing more abstract features reporting. For

instance, an object recognization is trivial for a human sensor, but for the same objective complex visual algorithms has to be deployed.

Accuracy The hard sensors are typically very accurate in the property they measure whereas humans function on a more qualitative level. For instance, a laser scanner can measure the distance to an object with an accuracy in the order of centimeters, but humans will report it more abstract terms like shorter, longer, 10 meters approx. or between 1 and 2 meters, etc.

Time Scale Human sensors can report different reports depending upon their individual capacity to retain the information and process the observations they make to arrive on a decision whereas sensor units can make periodic observations at a very high rate in synchronization with the hardware processor.

Uncertainty Both real sensors and human operators are uncertain in their measurements in varying degree of extent. The difference lies in the fact that degree of uncertainty varies from one human observer to another, in case of hard sensors the degree remains constant for sensor with same build and type.

Variability In case of a human observer, variability in accuracy and uncertainty associated with the information reported changes with time of the day even if the observer remains the same. In case of hard sensor though the difference in observations occupy small scale over a given period of time.

The differences listed above, illustrated the fact that hard sensors and humans have complementary perceptual abilities which offers an opportunity for effective information fusion.

The necessity to characterize humans as sensors arises out of the fact that, in mostly all the disaster rescue efforts, military defense operations and any team exercise, humans are at the thick of action either acting as supervisory mode, worker node or merely acting as an observer. The observers are generally not exploited in these situations but doing so can greatly enhance the

speed of these operations. With the telecommunication and ad-hoc network infrastructure in place and every human carrying multimedia enable mobile phones, they can act as a rich source of information by making observations though their senses and reporting it back to the fusing node and carrying their usual supervisory role but in a controlled manner. The humans make imprecise observations, and so an uncertainty is attached with these observations which need to be taken in to consideration before fusing this information from other sources. For developing software based model of Humans, which can process the raw observations from humans and process it to create an equivalent output which can be merged with hard sensor reading, the various characteristic features of Humans have been identified in the next section.

Approaches to model humans as information and decision making Software agents

Humans report abstract information which may be accurate with a very low degree of certainty, and such observations can be categorized into two modes. First, one is the "likelihood mode" in which the human observer himself adds the uncertainty while reporting his readings and second one is "raw observation" mode in which the human just reports the observation reading and then the decision making system based on the properties of the particular human sensor will add uncertainty value to the reported observation. The second model is what forms the core the "*Human perception model*" described in the next section.

2.2 Characterization of Humans:

The various characteristic features which affect the ability of humans to make correct observations are dependent on his intelligence level, knowledge and experience pertaining to the situation in which observations are to be made, his physical fitness and robustness, access to external gadgets (night vision goggles, camera, laser source, etc.) he is carrying on his body,

workload at any point of time during the course of operation, time of the day, hours in operation or residual energy if no replenishment in the form of intake of energy is done.

The overall potential of a human sensor is modeled as a function of weight attached to all these parameters. Based on the potential value of the various human sensors they can be segregated in to expert, mediocre and novice levels. The potential of the human nodes varies as a function of time based on some of the factors mentioned above. The reserve price of human sensors is a variable parameter and decides the trade-off involved while allocating soft sensors in various locations.

Some of the factors, like residual energy, have a monotonic decreasing function associated with them which continually reduces the value from the start of the operation from initially reported values. Similarly the other parameter also needs to be modeled for monitoring their effect on the overall potential of the human sensor.

Before the operation starts all the human based sensor sources need to register themselves with the FTSAs. By signup, it is meant that only after verifying the background of the person volunteering to be a sensor can FTSA start accepting information from it. This is necessary because during the course of operation if some anti agent begins to send false information then the system becomes corrupted leading to total collapse of the system.

2.3 Establishing Static model of the human sensors participating in the operation:

Multiple parameters affect how the observational capability of humans vary over a period of time as well as vary from person-to-person. So a single HPM cannot fit in as most optimum model for

all humans in the network, but still a basic core can be designed with a configurable parameter which can be modified to model a particular individual. The initial value of these configurable parameters is established using already stored information concerning the Humans and some offline experiments conducted on them to establish their IQ, physical fitness and vision. The following figure depicts HPM core instances and the associated parameters which affect the working of the probabilistic model.

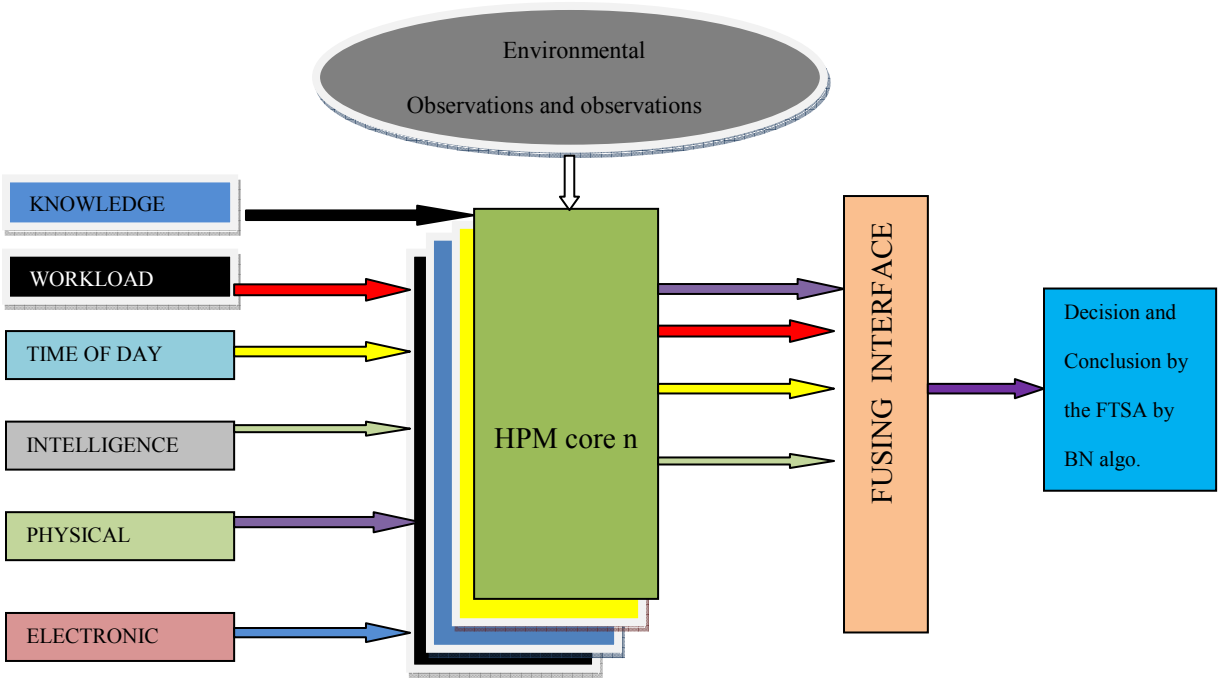


Figure 2.1 : A stack of HPMs "core" instances created by FTSAs

The parameters affecting the HPM can be divided into two categories. The first ones are static parameters like Knowledge, IQ level, Physical Fitness which once established at the starting of the operation are not changed. A second set of parameters comprises of dynamic parameters like workload and time of day, number of functioning gadgets which needs to be updated at regular

intervals to determine the potential of a human sensor and put him in one of the three brackets for classification.

2.4 Mathematical Model of HPM:

A human perception node can be defined as a multi-interface object capable of cognitive decision making and propagating its belief arrived at after fusing the observations made by him from his immediate environment based on the "potential factor" assigned to him. The potential factor, as mentioned earlier, is weighted sum of various parameters affecting human observation making capability and it keep on varying. The various interfaces of a human relevant to their operation as sensors can be classified as:

2.4.1 Sensory and Perception Interface (SPI):

This is the interface by which humans voluntarily make observations all the time and keep on storing it in memory and making inference on them, till the point of time when they are invoked or commanded to elicit that observation. This includes the sensory capability of the humans to hear, visualize, percept and smell and fuse all this to derive conclusion. The information flow on this interface is input to the HPM. The information reported by human agents on the ground is fed through this interface to HPM modeled in software.

2.4.2 Data Fusion Interface (DFI):

This is output interface in HPM which is used to inform the FTSA about the decision arrived after fusing all the information aggregated. This produces the likelihood of the observations made by a human over a period of time based on a window of past observations and uncertainty attached with particular human sensor at that point of time based on the value of potential. Basically

uncertainty will result in some variance factor association with the observation reported by Humans. This uncertainty can be minimized by feedback from the FTSAs or neighboring nodes.

2.4.3 Command Control and Communication (CCC) Interface:

This is the interface which serves as input/output for HPM. It accepts observations from other soft sensor and hard sensors which makes it aware of other events going in the region, local/global belief from the FTSAs so as to aid and enhance its decision making ability. Information flows both as input and output to this HPM at this interface. Input Information can be command from the FTSA and output can be command from HPM to any sub-ordinate sensor under control of that particular HPM.The HPM Core can be modeled as shown in the diagram (The arrows show the direction of information flow at each interface in a HPM core):

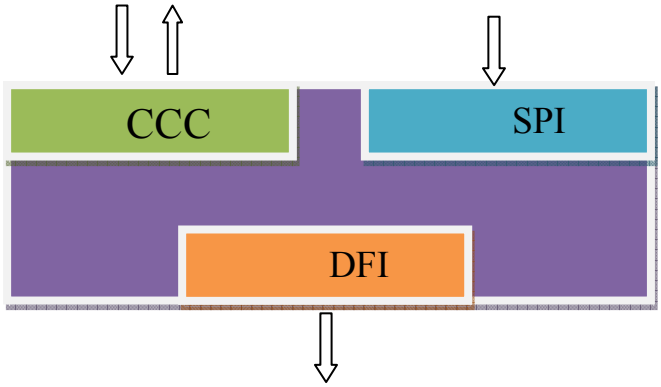


Figure 2.2 A block diagram model of Human Perception model showing the data flow interfaces

The HPM needs to have as many adaptive sensory tasks running as the types of observation reported by humans and so it has a Distance range estimator, Color Identification estimator (Visual), Smell Identifier Estimator, Temperature Estimator and General Class Label Identifier (Object type). The information processed by these estimators can be in absolute value attached with some range, or it can be completely fuzzified form. In case of fuzzy information we need to

convert it to some quantifiable value so as to fuse it with information from hard sensors. Also all performance of all estimators depends upon the potential of individual human sensor. All these are estimator types are described as follows:

2.4.4 Distance Range Estimator (DRE):

The input to this estimator will be both as fuzzy as well as absolute value with some range associated with it. For example the information can be in terms of "target 15 meters in front of me" or it can be "approx. 15 meters in front of me or" or it can be "target very close to me". All this should be converted to absolute values by first locating the GPS location of the human observer and then adding the relative distance reported by him. The relative distance is extracted from the above reported information by the Estimator and is fused with the past observations using Bayesian filter mechanism stated in equation 1 below, which is the Standard Bayes Conditional Probability rule for updating beliefs in response to evidence.

$$P(H \mid e) = \{P(e \mid H).\ P(H)\} / \{P(e)\} \qquad (1)$$

According to the above equation mentioned in [30], the belief is assigned a hypothesis H upon receiving evidence e {denoted by P (H | e), also called posterior} by multiplying our previous belief P (H) {priori probability} by the likelihood P (e | H) that e will materialize if H is correct.

The above equation can be applied in a modified form to estimate the new range based on past observations of human sensors and new observation made. This is termed as Recursive Bayesian Updation.

Let $\mathbf{e_{n-1}}$ be sequence of observations $e_1, e_2, e_2, \ldots, en_{-1}$ and let e_n be the new observation

made in the nth time slot. Based on this then we can compute changed new belief:

$$P(H \mid \mathbf{e_{n-1}}, e_n) = \{P(e_n \mid \mathbf{e_{n-1}}, H)\}.\ P(H \mid \mathbf{e_{n-1}}\} / \{P(e_n / \mathbf{e_{n-1}}\} \quad (2)$$

According to the above equation 2, the new belief is obtained by multiplying the old belief $P(H \mid \mathbf{e_{n-1)}}$ with likelihood function $P(e_n \mid \mathbf{e_{n-1}}, H)$ {implies the probability of new evidence given the hypothesis and old observations}.

We can combine observations up to nth time slot into one variable $\mathbf{e_n}$ and so equation 2 can be reframed as:

$$P(H \mid \mathbf{e_n}) = \alpha\ \{P(e_n \mid H)\}.\ P(H \mid \mathbf{e_{n-1}}\} \quad (3)$$

Where α is a normalizing constant.

Further we have to take into consideration the varying potential of the Humans which has substantial effect on observation making capability of the humans. The dynamic function to arrive at a particular potential value of P is dependent on the individual factors discussed earlier affecting humans. Assuming that we are constantly updating the potential P after regular time slots, the equation 3 can be modified to include dependence on P as:

$$P(H \mid \mathbf{e_n}, \mathbf{P_n}) = \alpha\ \{P(e_n \mid H)\}.\ P(H \mid \mathbf{e_{n-1}}, \mathbf{P_{n-1}}) \quad (4)$$

As is evident from the above equation this is recursive in nature. The first term on the RHS represents the likelihood, i.e., if H is the correct value of the range then what is the probability that e_n will be the observation made by sensor and the second term represents the

belief till last time slot.

A particular probability distribution can be assigned to uncertainty attached with the human sensor observation. This pdf will have some parameters (like mean, variance for Gaussian pdf) grouped into β associated with it which needs to be updated with time.

Finally after a particular number of time slots a mean estimate of all the observations are made to approximately determine the position of target. So this operation can

localize the target in a definite region. Equation 4 after including β will be:

$$P(H \mid \mathbf{e_n}, \mathbf{P_n}) = \alpha \{P(e_n \mid H ; \beta)\}. P(H \mid \mathbf{e_{n-1}}, \mathbf{P_{n-1}}) \quad (5)$$

2.4.5 Practical Implementation of DRE:

To initiate range estimation by Humans for any target during the course of operation, first we need to find out the priori pdf associated with uncertainty associated with any human observer. This basically has to be done to train the DRE module in HPM instance regarding prior bias corresponding to that particular human observer.

So statically for various targets at various distance range, the human observer is made to estimate the location of target. The distance range will vary from 0-5, 5-10, 10-15, and so on. Corresponding to this data set we will have mean and variance associated with every range. In this type of bias estimation, we have already prior knowledge of the target's location.

In actual condition, when a human reports information for the first time, then DRE will have no prior information about the location of the target, so either it has to rely on any other hard sensor observing the same object or trust the observation reported and then calculate the estimated location using the following equation:

$x_n = H_n + N(\mu,\Delta)$ assuming the pdf associated with DRE is Gaussian in nature.

μ,Δ ; the parameters associated with uncertainty distribution have to be updated in every slot for increasing the accuracy level of the observation. This corresponds to the update of parameter $\beta(\mu,\Delta)$ as discussed in the previous section whose update depends upon past window of observations, other sensor reported estimated value of true range of target, P, and the current potential of human observer. A mathematical formulation needs to be derived to update μ,Δ in a systematic manner as a function of all these parameters for every human observer involved in the field reporting information in every time slot whenever information is reported.

2.4.6 Color Identification estimator (CIE)

The main objective of modeling a CIE is to fuse the fuzzy information provided by Humans and absolute value provided by the hard sensors. Besides it should be able to remove the uncertainty associated with information reported by human observations. Every color component of the spectrum is assigned a range of values. Every shade of a particular color will have some mathematical function. For instance, if we consider red color, then its shade can vary from light red to very dark red and so this range is assigned a function growing linearly from 10 to 20. All the sensor on observing a particular target will report its color. The value reported can be plain text "red" or an absolute value 15. For text based report they are converted to absolute value with the help of a Look Up table and then the mean of all values reported is taken and reverse mapped to the particular color component and its shade in the loop table. Now the question arises how uncertainty involved with human observation is removed from this estimator? The uncertainty arises due to vision properties of the human observers and their potential P at any instant during the course of operation .

Before the operation is started “color identification test” on all the human observations has to be made with respect to all the component of colors and also a probability value is assigned for every human observer as shown below :

Color shade	Faintof Red	Light Red	Middleshade	Almost Red	Dark Red
Human1	.9	.8	.4	.8	.6

Table 2.1 Color Identification Probability values for a typical human observer

Then at run time the probability in the above table is multiplied with probability associated with a correct observation making capability of the human observer depending upon potential P at that instant which gives us P_{final}(color_observation).

So to determine the color of the object expected value of the observations reported is calculated by the following formula:

$$| \text{Expected\_Color\_value} | = \sum_{i=1}^{N} P_{final}(\text{color\_observation}) \text{ color\_value}$$

N is the number of total number of hard sensors and soft sensors who report the corresponding to the observation reported by them. This expected value is then reverse mapped into the LookUp table to find the corresponding color of the object.

Lookup table aids in this operation. Other Estimators like Smell detectors, temperature sensor can also be designed in a similar fashion involving human perception but has not been dealt with in this article.

2.4.7 Class label identifier (CLI) is very high level identifier and basically aides in direct recognization of objects in case of visual tracking. For instance without this capability of humans,

a lot of sensors have to make numerous observations and then some pattern recognization algorithm is being executed to ascertain the identification of the target object. On the other hand, if a human observer reports the class label to which the object belongs with considerable ease.

Although they will not be able to provide details of the target. For example, if there is a car as the target, then it can be classified directly into "vehicle" class.

2.5 Soft Agent Models(FTSAs and STSAs)

The first tier software agent (FTSA) as mentioned earlier acts as the main fusion nodes for the heterogeneous data streams coming from various hard and soft sensors. The various main functionalities of FTSA can be classified into following major categories:

- Regional Event Identification

FTSA nodes run Casual Bayesian network* based algorithm (in the context of disaster response scenario) to identify the target and event type happening in the region for which they are monitoring the data inputs from various sensors. Based on the correlation between various sensor observations reported, and the type of observations made, a decision is made confirming the happening of the event or certainty about the target feature and its presence or absence. For instance, if all heterogeneous sensors are observing a particular target (assuming the case of a single target which needs to be tracked down) and after DRE in the HPM has acted upon and every other hard sensor has also reported down the value to FTSA, a final calculation is done by the to find the weighted average on all observations as :

$$|\text{Expected\_value\_observation}| = \sum P(\text{correct\_observation\_by\_sensor})\ \text{observation\_value}$$

Where P(correct_observation_by_sensor) corresponds to the potential of the either human

observer or hard sensor to make a correct observation. The observations can be the distance of the target relative to the sensor making the observation, and so a GPS position needs to be added with respect to some reference position to get the absolute observation value. Observation can also be value pertaining to the color component. Also, humans can report the class label associated with the object. So, by this fusion process, a final value is reached and the result can be " "red car" at 5

meters observed by sensor moving at 40kms/hr (speed sensor can report this) moving in some particular direction".

- Authenticate and Authorize (AA)

FTSA has also an AA module which exchanges security keys with the human observers every they try to enter a new region. The potential value of that observer is fetched from the central database of all sensors who have registered for working and reporting back field observations from the place of action. This serves to avoid breach of security so that the system is secure.

- HPM Instantiation and Data Aggregation

  FTSA instantiates a software based module HPM (explained earlier) with its parameters customized to a particular human observer. The number of such HPM instances is equal to the number of observers present in the region under control of FTSA. The purpose of creating such HPM agent is to generate most likely information after removing the uncertainty attached to the raw observations reported by the humans. Besides it directly takes raw observations from the hard sensors and refines them to fuse them with human observations.

- Resource Allocation

FTSA works in collaboration with STSA to allocated resources. If some FTSA reports a lot

of activity happening in its controlled region to the STSA and it cannot handle any further then it is the task of STSA to mobilize the resources from another area which is lying idle for a period of time. For this it needs to communicate this to the FTSA which finally issues commands to human observers to move to another region. Also sometimes a single FTSA will handle multiple regions simultaneously in case of the failure of some FTSA as commanded by STSA.

- Command and Control

FTSA issues commands to human observers from time to time to mobilize them and to control the state of hard sensors present under its control. It also assigns priorities to various sensors and helps resolve conflict if any between two human observers for accessing hard sensors at the same time. For example, if two human observers are trying to issue commands to a hard sensor, e.g., camera to point in opposite directions then, it is the FTSA which will resolve this conflict and depending upon priority attached to a particular human observer, he will be given first control.

The FTSA can also take information from cyber space such as viz. Twitter, and other social Networking sites from registered users and from another government based security websites regarding the activities happening at the ground level and fuse these information with information reported by Hard and soft sensors. In this thesis the method to extract such information and querying the sources for it has not been discussed to avoid too much complexity.

The Second Tier Software Agent (STSA) also possess the basic functionalities mentioned above, the only difference is they act upon FTSAs. Based on inputs from all FTSA they make a decision which percolates down back to the human observer through FTSA again and besides this they also help in allocation and deallocation of FTSAs from one region to another one. The

STSA maintains a database for each FTSA with the level of activity associated with them which helps them to determine whether any FTSA is idle or in high activity state so as to do optimum allocation of tasks. A FTSA with low activity can be merged with high activity FTSA at run time to maintain optimum performance of the system.

The major utility of the STSA comes when predictions related to the movement of the target and its mobility directions. Thus, it has a full map of the area and also has GPS locations of all the sensors present in the region. So whenever FTSA informs it regarding position coordinates and direction in which the target is proceeding, STSA can inform in advance and send red alerts to the FTSA of another neighboring regions which in turn then alerts the ground agents.

Chapter 3

Tools and Techniques for Modeling

3.1 Probabilistic Representations and Belief Propagation Inference Algorithms

This chapter deals with the methods for conventional techniques for representing sensor network nodes forming large distributed graphs and how the information propagates among these nodes to finally facilitate the decision making ability for various nodes in the network. It introduces the background to the work contained in this thesis, explaining the way in which the multi-agent approach to partially observable systems is developed from single-agent decision theory and justifying the decisions made at each step in building on the state of the art.

Probabilistic representations have gained wide acceptance due to their suitability to address the main research problems of perception, decision making, and planning. Probability theory allows to explicitly accommodate the uncertainties of the real world. Uncertainties stem from both noisy measurements and inaccurate world models.

Probability theory provides a unifying mathematical framework for reasoning and decision making under uncertainty. Probabilistic algorithms for perception, control, and planning problems are briefly discussed below.

3.1.1 Autonomous Agents

In this thesis, we assume that an agent is an entity which is situated in some environment and reacts to that environment, in order to try and achieve some objective or goal (this definition is based on [21], chapter 1. Such agents will be able to reason logically about their actions and the effects of the actions on the current state of the world, relating this to their goal. At times, inconsistencies and conflicts in agent goals and beliefs may crop up; the agent's reasoning mechanisms must have some means of resolving these. I believe that probabilistic

provide a realistic way to do this for two reasons. First, such techniques are effective for reasoning in uncertain scenarios where an agent may want to reason using its belief in a particular property [24]. Second, probabilistic representations are typically more compact than their logic-based counterparts for both the input data and the agent models [25],[26].

Given such a reasoning mechanism, this mechanism must consider both the signal the agent receives from its environment, and the effects of its own actions. When reasoning, the agent may maintain an explicit world model or it may leave the model implicit as it reasons about plans. In this setting, explicit models have more potential for reasoning about states and behaviours, as they store more information explicitly. However, maintaining explicit models may be computationally and memory intensive [27].

Despite this, I believe that the aforementioned benefits of explicit models justify their use where practical. Now, if, as in many disaster arenas, the world is large and detailed, agents may only be able to create such explicit models for small parts of the world, due to memory constraints. In such worlds, it is, therefore, appropriate either to use a model which can store information at different levels of detail, or to reason in a simplified abstract world.

Finally, as well as reasoning about their environment, agents in a multi-agent system will interact with each other. Techniques are described to show how to extend models for reasoning under uncertainty to incorporate explicit considerations of the other agents.

3.1.2 Markov Decision Process (MDP)

The most straightforward of this class of dynamic problems is the single agent observable Markov decision process (MDP). The MDP forms the theoretical foundation for reinforcement learning problems—when the environmental dynamics are unknown—and partially observable problems, both of which we will be discussed.

In an MDP the agent perceives the state of the world **s** through its sensory inputs, and decides on its immediate action **a** based on this state. Following the agent's action, the world *transitions* into a new state **s′,** and the agent may receive some *reward* **r**. A key feature of such problems is *delayed reward*: states which have no or negative reward, but which ultimately lead to higher rewards [28]. This is a common feature of many real life problems in crisis like an earthquake, ambulances may use up valuable fuel for no immediate gain, and rescuers risk being hurt , before the final goal of a rescue is achieved. Determining the reward achieved from a particular action may be straightforward— a rescue worker retrieving valuables from a building may receive a fixed reward for each valuable he retrieves, or an ambulance team may receive a fixed reward for each person loaded into an ambulance alive.

However, in some kinds of problem deciding a reward function may be more challenging. For example, following an earthquake, buildings may be burning while humans are buried and trapped. A reward function for a team simultaneously rescuing humans and extinguishing buildings may try and put relative values on the buildings and the human lives, supplying some reward for unburnt buildings and some for live humans. An alternative reward function might try and assign higher value to some humans—for example, the prime minister, or people who can be immediately useful to the rescue.

These intuitions are pinned down by defining a finite set of states S, a finite set of actions A, and a finite set of rewards R. The environment dynamics are then defined by [28]:

A transition probability function, $\mathbf{T_f} = P(\mathbf{s'|s, a})$. This defines the probability of reaching state **s′** from state **s** given that the action performed was **a**.

A reward probability function $\mathbf{R_f} = P(\mathbf{r|s, a, s'})$. This defines the reward achieved by taking action a from state **s**, resulting in state **s′.**

The agent makes decisions according to a policy ∏ where ∏ (**s, a**) defines the probability the agent will take action **a** from state **s.**

In this model, the probability of transitioning to a particular state, or achieving particular reward, does not depend on any of the history of states and actions. This *Markov property* is the fundamental feature of Markov decision theory [28]. Given this context, the goal for the agent is to maximize some function based on the reward obtained. This may be [28]:

• over a fixed time horizon

• during an *episode* in which the agent continues to act until some termination condition is reached

• the average reward over an indefinite time period, or

• the total reward over some time period

In the last case, more recent rewards may be valued more highly than earlier rewards—in particular, this encourages adaptation to *non stationary* environments, in which the environmental models are changing over time. In disaster scenarios, we may consider either the total reward accrued (perhaps in number of lives saved) when some termination condition is reached (the scene is cleared up), or we may consider how efficiently agents can act to accrue reward over a fixed time.

In the experimental simulation of the disaster scenario, the objective is to see how fast the agents are able to accrue high rewards. Specifically in choosing any action at time $t = t_T$ the agent's aim is to optimize the expected discounted future rewards defined by

$$R_T = \sum \gamma r_t \quad (t \text{ ranges from } T \text{ to } \alpha)$$

Where γ is a problem specific parameter which defines the systems view of importance to the action taken by the agent in depending upon the timeframe. It balances the importance we place

on future states with our need to accumulate reward now.

Typically a Y value of .8, making look ahead negligible after around ten steps into the future- in a fragile disaster scenario, this is expected to be sufficient for planning purpose as agents have to adjust their plans to a changing situation within a few steps in any case.

3.1.3 Partially Observable Markov Decision Process (POMDP)

Generally, MDP forms the basis of complex localized multi agent environment, but in case of Large-scale distributed network, it is common for an agent to make local observations without observing the complete state directly (although in a Multi agent environment, local observations may be augmented with communication information). When the underlying process of moving from one global state to another global state is still Markov, the scenario is described as partially observable Markov decision process or POMDP.

Specifically, if it is assumed that there exists a fixed known model, $O_{f=} P(\mathbf{s} \mid \mathbf{o})$ where **o** is the current set of observations and **s** is a possible state. Although the sequence of states is defined by MDP, the sequence of observations is not. Like, if the current observation is $\mathbf{O_2}$ then the probability that the next observation will be $\mathbf{O_5}$ differs depending on whether the previous observation was $\mathbf{O_3}$ or $\mathbf{O_1}$.

To solve a POMDP, as will be happening in case of a distributed crisis or disaster management scenarios, a secondary MPD - *belief* MDP can be derived with a single continuous variable let say **b** which is for every possible value of state **s** which is in turn multidimensional in nature.

The value of **b(s)** indicates the agent's belief that the underlying state is **s**, given that agent's prior knowledge and history of observations and actions. The system proceeds from state defined by $\mathbf{b_1}$ to $\mathbf{b_2}$ at each step using Bayes rule:

P(x| observations) = P(observations) P(x)

Apart from this *state abstraction* technique is used in which, many combinations of input variables are mapped to a set of states on the higher level. Such state abstraction can also be used to encode intuitive states which cannot happen, like fires will never occur in water body . Besides due to humans acting as observers, and their specialty to recognize coarse features easily rather than minute finer details such state abstraction will easily facilitate recognization of a particular feature.

In the above technique, the agents must have some notion of similarity between states. With abstraction, similar states are treated the same and given the same action. With generalization, as well as similarity, the learner must have an idea regarding whether the new unknown state has a higher or lower value than the known state. In a new emerging disaster operation, the various states are not known *a-priori* so, as and when the situation evolves the agents have to learn based on the past events they have faced and training they have obtained before starting out with the operation.

In this context, a *statistical clustering* based technique is used [22], in which a type of probabilistic clustering is utilized to achieve this kind of abstraction. Each state can be identified by a set of binary features (at minimum ,states can be defined by numbers and the binary features the bits in state's number). A cluster is then described as probabilities over the states in the cluster :

$C = <p_1, p_2, p_3, p_4, p_5 \ldots, p_n>$

Where p_i is the probability that the **bit i** is set , given that the state is in that cluster. The probability of a new states being assigned to a new cluster can thus be calculated as every setting or unsetting /setting of bit which signifies identification or realization of new event or feature is independent in nature. The probabilistic nature of statistical clustering and state abstraction fits well with learning using Bayes net this can be used in modeling the disaster response scenario

described in the Introduction in next chapter.

The algorithm for modeling the statistical clustering along with various features and identification criteria apart from state assignment will be explained in next chapter

3.1.4 Bayesian Networks and Extensions

Perception and decision-making problems mentioned above can be easily represented by dynamic Bayesian Networks, or if decisions and utilities are represented explicitly, Influence Diagrams. In a single representation, they are capable of representing

• Variables related to perception and decision making

• Both correlations and the lack of correlations between random variables

• Observed and unobserved random variables

• Discrete and continuous random variables

• Static and Dynamic processes

• Multiple abstraction levels

• Different time scales

Besides this flexibility for modeling purposes, there are many prominent algorithms for efficient inference and learning for these representations [25] [27] [28] [29]. The properties and usage of BNs, DBNs, and IDs are presented next.

3.1.5 Bayesian Networks

A probabilistic representation should encode the relationships between random variables qualitatively (model structure) and quantitatively (model parameters), and allow efficient inference and learning. A class of graphical models fulfilling these requirements is *Bayesian Networks* (BNs). Like other graphical model representations, BNs encode a joint probability distribution of a set of random variables in a compact form by exploiting conditional independence assumptions.

Each random variable is called a *chance node* and denoted by a capital letter, *e.g. X*. The realization of a random variable is denoted by small letters, *e.g. x*. If the random variable is a vector as opposed to a scalar, it is printed in bold, *e.g.* **x**. Chance nodes can be either discrete or continuous. All equations in this section assume discrete nodes which can be rewritten for continuous nodes by replacing sums with integrals.

3.1.6 Bayes Net Representation

A BN is a directed acyclic graph (DAG) consisting of chance nodes and edges which connect the nodes . If there is an edge from node *X*1 to *X*2, *X*1 is called a *parent* of *X*2 and *X*2 is called a *child* of *X*1. Each node *Xi* has a conditional probability distribution (CPD) $p(\mathbf{x}i|parents(\mathbf{x}i))$ encoding the effect of the parents on that node. An edge from parent to child can be interpreted as a cause-effect relationship.

The set of nodes and edges form the topology of the BN which specifies the conditional independence assumptions. Two equivalent statements can be used to define conditional independence in BNs:

(1) a node is conditionally independent of its non-descendants given its parents

(2) a node is conditionally independent of all other nodes in the network given its parents, children, and children's parents (the so-called *Markov blanket*).

To determine the global independence relationships of three sets of nodes (is *X*1 conditionally independent of *X*2 given *X*3?) the *d-separation* criterion can be used which is beyond the scope of this short introduction.

The full joint distribution of a set of random variables *X*0, ...,*Xn* as part of a BN can be written as

$$p(\mathbf{x}_0, \ldots, \mathbf{x}_n) = \prod p(\mathbf{x}_i | \, parents(\mathbf{x}_i))$$

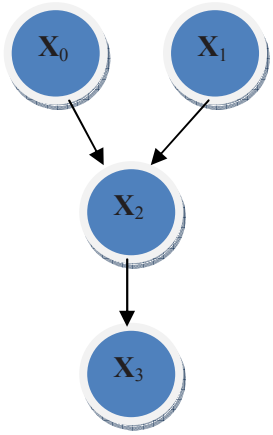


Figure 3.1 Bayesian Net example

An example of a BN is shown in Figure 3.1 above. According to this, the joint distribution for this example can be written as

$p(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = p(\mathbf{x}_0)p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_0, \mathbf{x}_1)p(\mathbf{x}_3|\mathbf{x}_2)$

3.2 Inference from Bayes Net

Since BNs encode the full joint probability distribution of a domain, any query can be answered. A typical query asks for a probability distribution of a set of *query* nodes X given some *evidence* nodes Z which is referred to as *probabilistic inference*. In mathematical terms, a query is written as $p(\mathbf{x}|\mathbf{z})$ where $\mathbf{z}$ emphasizes the instantiation of Z with a value. Nodes in the BN which are neither query nor evidence nodes are denoted by Y. The answer to a query can be computed by using the joint distribution:

$p(\mathbf{x}|\mathbf{z}) = p(\mathbf{x}, \mathbf{z}) / p(\mathbf{z}) = \sum_y p(\mathbf{x}, \mathbf{y}, \mathbf{z}) / \sum_x p(\mathbf{x}, \mathbf{z})$

Expressions $\sum_y p(\mathbf{x}, \mathbf{y}, \mathbf{z}$ and $\sum_x p(\mathbf{x}, \mathbf{z})$ are called *marginalisations* of Y and X, respectively.

A special case of the general problem is the BN shown in Fig. 3.2 below which is used to introduce a

number of important terms. The BN contains two sets of nodes *X* and *Z*. Initial knowledge about **x** is represented as a *prior* probability distribution $p(\mathbf{x})$. The evidence entered on *Z* is called *observation* **z**. For this particular problem, Bayes' theorem can be applied:

$p(\mathbf{x}|\mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x})\ p(\mathbf{z})$

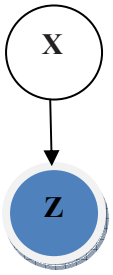


Figure 3.2 Simple Bayes Net

The figure 3.2 above illustrated a simple BN to demonstrate Bayes' theorem and define important terms. The shaded node is observed

The answer to the query $p(\mathbf{x}|\mathbf{z})$ is called the *posterior* encoding the belief of **x** after incorporation of observation **z**. Observation $\mathbf{z} = \mathbf{z}$ is substituted into the CPD $p(\mathbf{z}|\mathbf{x})$ to yield a *likelihood function* $p(\mathbf{z}|\mathbf{x})$. The function represents a distribution over the values of the true state **x** and can be understood as a slice through the $\mathbf{x} - \mathbf{z}$ space. A likelihood function is simply referred to as a *likelihood* throughout the remainder of the thesis. Finally, $p(\mathbf{z})$ acts as a normalization constant.

For more complex BNs with many nodes, the conditional independence assumptions encoded in the graph topology can be exploited to design efficient inference algorithms.

The strongest reason for the use of Bayesian Networks is that "they are a direct representation of the world, not of the reasoning process" [30]. The directed edges in the Bayesian Network graph represents the real casual relationships between parent nodes and their children and is not just indicative of the flow of information during the inference process. Bayesian Networks also allow

bidirectional inference – given a cause what are the effects, or given an effect what are the likely causes,

3.2.1 Belief Propagation Inference Algorithms

Complex Bayesian graphical models consisting of thousands of cause effect relationships needs to converge on one inference so as to help in decision making for the system utilizing the Bayesian logic and decision making engine. The complexity of solving a graphical model is not only determined by the number of nodes in the model, but also by the structure of the graph. Graphs which are singly-connected are relatively straight forward and algorithms provide exact solutions that run in polynomial time. However, if loops exist in the graphical model then the complexity increases dramatically. A loop is an undirected path in a graph that both starts and ends at the same node. If a loop exists in the graph then the nodes in graph are not singly connected.

Loops in the Bayesian networks are undirected cycles in the underlying network (i.e., neglecting the direction of the link arrows). A network with loops is no longer singly-connected and local message-passing techniques such as belief propagation may not converge. For loopy graphs message passing algorithms risk accounting for the same evidence multiple times since the evidence from a single source may arrive at specific node through multiple paths.

Chapter 4

Implementation of Bayesian model of Partially Observable Markov Decision Process

4.1 Definitions

First of all, some basic definition will be explained before modeling the general Bayesian POMDP which will be customized in next chapter to a specific problem of disaster response and crisis management in the wake of earthquake or emergency created due to unexpected attack by terrorist to demonstrate the utility of online Bayesian learning algorithms developed in this chapter.

Throughout the description of this general model, an assumption is made that, there is an underlying state which changes in response to the joint action of the agents and with constant effect of dynamic factors emerging in the environment. Another assumption is that agents are not able to make complete inference about the state of things existing in particular area of the vast geographical region. They make some observations **o** from which they make inferences about the state. These inferences may include their own decision making capabilities in case of humans, or simply reporting their observations like hard sensors to FTSAs which then make decision or identification of the event based on reported observations.

Formally the model includes these parameters:

- $I : \{I_1, \ldots I_k\}$, a set of k agents
- $S : \{s_1, \ldots s_n\}$, a set of n_s states. A state will generally be described by a set of state variables.
- $L \in S = \{L_1, \ldots L_k\}$, a location variable for each agent. These determine the viewpoint

from which agents make local observations.

- $A = \{a_1, \ldots a_n\}$, the set of individual actions. $A = A_k$ is the set of joint actions . Thus, we differentiate between a single action a and a joint action **a** by using bold for the latter, to emphasize that it is a vector. We will also use a_{-i} to refer to the vector **a** with the element corresponding to i removed, and a * a′ to refer to a with a′ integrated.

- $O : \{o_0, \ldots o_n\}$, a set of n possible observations. Each agent's observations will be taken from this set.
- $T_f : Tf(s_{t+1}, s, a_t) = P(s_{t+1}|s_t, a_t)$, the transition function from the state at time t to the state at time t + 1, where $s_{t+1}, s_t \in S$ and a $\in$ A. We use the subscript **f** here and below to distinguish the functions from sets.

- O_f : An n_0 -dimensional function where $O_f(s_t, o_t)_i = P(o_t|i, s_t)$, the observation function for agent i, where $o_t \in O$ and $s_t \in S$.
- $R : \{r=1, \ldots r_n\}$, $n_r \leq n_s$, a set of possible rewards which an agent may receive
- $R_f : S_x A_X S \rightarrow R$, a reward function: each agent will have its own reward function. Typically, the reward will be associated with the immediate state, but for some problems it may be associated with the transition between states (for example, if actions have a cost).

When taken together, T_f , R_f and O_f describe the dynamics of the environment. We will use $\Phi = (T_f, R_f, O_f)$ to refer to these dynamics as a whole. An individual agent, A, may also have:

- A (deterministic) policy Π: $(P_x H_x O) \rightarrow A$ where h $\in$ H defines all relevant historical information (observation sequences including communications from other agents), p $\in$ P any prior or domain knowledge, $o_t \in O$ is the current observation and a $\in$ A is a single

agent action. Typically, (p, h) will be compressed to contain the sufficient statistics for a belief state (a probability distribution over states and unknown parameters).

- Beliefs over unknown parameters: for some variable X taking values $x_1, x_2, \ldots, b(x_i)$ is the probability that $X = x_i$, given the agent's prior information and subsequent observations.
- Models of the other agents' behavior: $P(\Pi_i | p, h)$ where Π_i has the same form as Π above and (p, h) refer to the prior and historical information of the agent A. To be clear, we assume that the other agents have deterministic policies, and our agent maintains *beliefs* over these deterministic policies. The agent can be human agent or first level or second tier software agent getting feeds of information from ground level agents of all types.

Taking these definitions, the next section will describe a formal model of learning in multi-agent systems, extending the Markov decision process to a continuous "Bayesian MDP" and explaining how this Bayesian MDP can be used to encapsulate partially observable multi-agent systems (POMDP) in the next section.

4.2 Bayesian MDP

Consider all the possible variables and parameters of a partially observable multiagent MDP: the world state s, the agents' actions a, the environmental models $\Phi = (Tf , Rf , Of)$, and the agents' strategies Π. We can create a multi-dimensional "grand state", which contains all of these things, $g = \{s, a, \Phi, \Pi\}$ and describe a "grand MDP" which proceeds through such states given the single agent action act:

In this MDP, provided the environment and agent behaviours are static, Φ and Π are unchanged between states, while the transitions for the state are determined by Φ —the dynamics for the

underlying MDP—and **a**, and the transitions for the joint action are described by Π —the agent behavior models—and **s**. From this MDP, we can describe a POMDP in exactly the same way as any POMDP is described from an MDP with partially observable states. In order to define the POMDP from the MDP we must specify the observations. The set of observations for agent **i** in this POMDP includes:

- O_S: observations about the state
- O_{aj} : observations about agent actions
- R_i: the individual agent's rewards
- O_{rj} : (in a non-cooperative system) observations about the rewards of others
- $O_s \rightarrow Ai * O_a \rightarrow O_s'$: observations about the transitions.

From this POMDP, a belief MDP can be defined wherein the "belief states" of the belief MDP contain probability distributions over the variables in the actual state.

The belief state is described as $b\{s, \{a_j\}, \Phi, \{b_j\}, \{\Pi_j\}\}$ where :

- s is the current immediate state of the agent.
- $\{a_j\}$ are the immediate actions of the agents in immediate surroundings of the current agent.
- $\Phi = T_f, R_f, O_f$ are the environmental variables.
- $\{b_j\}$ are the beliefs of the other agents about the world state.
- $\{\Pi_j\}$ are the behavior of the other agents based on their beliefs.

These definitions and terminology described above can be used to define an MDP Belief Algorithm for an agent i in a partially observable multi agent scenario as follows:

MDP Algorithm

- The Agent initialises its belief state $b_0 = b\,(M)$ where $M = b\{s, \{a_j\}, \Phi, \{b_j\}, \{\Pi_j\}$ either uniformly or based on the domain knowledge.
- At each time step , the agent i
    1. Makes its observations
    2. Updates its beliefs over M using Bayes rule resulting in a new belief state b_t
    3. Calculates the action a_i which optimizes $Q\,(\,b_t\,,\,a_i\,)$ where

$$Q\,(\,b_t\,,\,a_i\,) = \sum_M P\,(\,M \mid b_t\,)\,Q\,(\,s,\,a_i \mid M\,)$$

    4. Executes this action and makes new observations.

For updating the belief at step 2, complex calculations are required as summing over all the unknowns in the environment needs to be done. For implementing that, junction tree algorithm for calculating within graphical models can be used. Before moving further, the next section will put further light on belief networks.

4. 2. 1. Belief Networks

A Bayesian network, such as shown below in the figure, exhibits dependency relations between variables, with directed arrows indicating the direction of causality: an arrow from a node A to a node B indicates "A directly affects B". Figure 4.1 shows a simple model for an earthquake: Earthquakes are assumed to occur independently (with some probability), so Earthquake forms a root node in the network. The likelihood of a fire is dependent on whether or not an earthquake has occurred; This is represented by drawing an arrow "*Earthquake affects Fire*". The left hand

table, called a conditional probability table or CPT, shows the conditional probability of a fire given that an earthquake did occur (Earthquake=1) or did not occur (Earthquake=0). Similarly, we have an arrow "*Earthquake affects Buildingcollapse*" (the corresponding CPT is omitted). Finally, both fires and building collapse may result in victims and so we have arrows from each to victims. The table 4.2 shows the conditional probabilities for finding victims given all possible combinations of values for (Fire, Earthquake): the "parents" of the Victims node. Such a diagram encapsulates the dependences between variables and thus the independence relations between them. Now, in general, in a probabilistic system, the probability of a particular variable assignment can be factorized using the product rule as follows:

$$P(x_1, \ldots, x_n) = \prod_{i=1} P(x_i | x_{i+1}, \ldots, x_n)$$

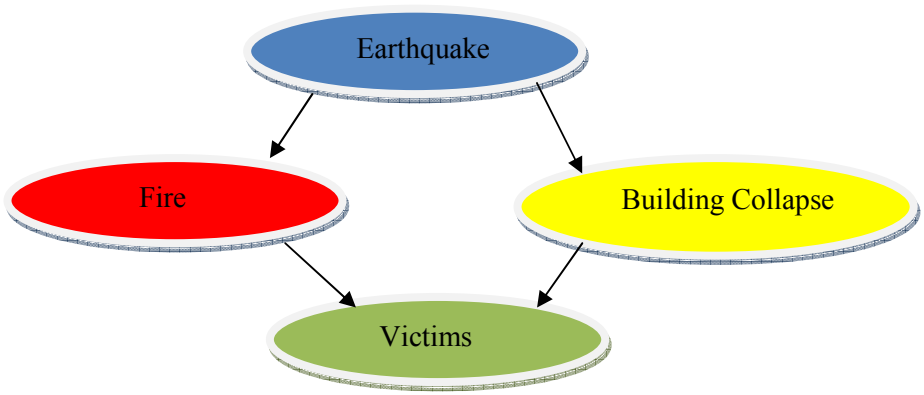


Figure 4.1 Bayesian Belief Network Analysis Example

Earthquake

Fire	0	1
0	. 8	.6
1	.2	.4

Table 4.1 Probability chart for Earthquake-Fire

Fire, Earthquake

Victims	0,0	0,1	1,0	1,1
0	.9	.6	.15	.85
1	.1	.4	.30	.15

Table 4.2 Probability chart for Fire, Earthquake and Victims

Where the conditional independence relationships in the system are known, this factorization can be simplified given the fact that P(A|B,C) = P(A|C) if A and B are conditionally independent given C. Now, the variables can be ordered in any way before applying the product rule, and some orderings permit more such simplifications than others. To see this, consider a particular variable assignment; Fire = f, Earthquake = e, Building collapse = b, Victims = v over the variables in figure above

If we order the variables e, f, b, v, then the product rule gives us:

P (f, e, b, v) = P(e|f, b, v)P(f |b, v)P(b |v)P(v) which permits only one simplification:

P(f, e, b, v) = P(e|f, b)P(f | b, v)P(b |v)P(v)

However, if we order the variables (v, e, b, f), then the product rule gives us:

P(v, e, b, f) = P(v|e, b, f)P(e|b, f)P(b|f)P(f) which has two simplifications:

P(v, e, b, f) = P(v | b, f)P(e|f)P(b | f)P(f)

Furthermore, there is no general way to find an optimal ordering [23]. However, the Bayesian network diagram supplies a way of ordering efficiently. This is to order the variables so that every variable has a lower number than all its parents, and then each node need be conditioned only on its parents (see for example[23]), chapter 6 for more details).

Now, in any problem, we will have some variables which are observed, and others which are *hidden*. We care about the likelihood of some of the unknown variables, but not all. For example, if an earthquake occurs, a disaster response agency will have ambulances to dispatch to victims, and fire engines to dispatch to fires, but no services to dispatch to collapsed buildings: collapsed buildings are only of interest because of their effect on the likelihood of victims. Thus, if the

buildings are not observed there is no need to compute the likelihood of their collapse. Thus, we marginalize out the unknown variables which do not interest us (summing over them), and normalise over the observed variables. In the belief network visualisation, as with the MDP progress diagrams in the previous section, we will distinguish between the observed variables and the unknown variables by shading unknown variables gray and leaving observed variables unshaded. Bringing together these techniques gives rise to the following standard algorithm for calculating the probability of a hidden variable v in a system:

1. Identify all the variables and parameters in the system
2. Draw up a Bayesian network diagram with a node for each variable or parameter, indicating the "directly affects" relationship between nodes.
3. Shade the nodes which correspond to hidden variables or unknown parameters.
4. For the node corresponding to the variable of interest, write down its probability, conditioning on the observed (unshaded) nodes {obs1, obs2, ...} and summing over all the other hidden (shaded) nodes {v1, v2, ...}:

   $$P(v|obs_1, obs_2, ...) \alpha \sum_{j=1,2...} P(v_1, v_2, . . . , v, obs_1, obs_2, ...)$$

5. Using the Bayes network as a guide, factorise and simplify P(v1, v2, . . . , v, obs1, obs2, ...), removing any constant factors (i.e., factors independent of v) as these can be normalised over later. To achieve this factorization, first, note that the term within the sum contains every node in the system. Begin at the leaves of the network and for each leaf l, write down the term P(l | parents(l)) where parents(l) refers to those nodes which directly affect l. Then, for each node p in parents(l), write down the term P(p| parents(p)). Continue in this way until you have included a term for every node in the system. This gives the factorization. To simplify the equation, move any terms which contain no summed-over nodes outside the sum. Any terms which contain only observed variables may be removed as constant factors.

Now, we propose to apply this technique to our MDP system. To do this, we draw up one agent's belief network. This network will contain variables for the current state g, the agent's action, and the future state. In practice, rather than use a single variable g to represent the grand state, we can separate out each of the variables in the grand state, thus making explicit the dependencies between variables and allowing us to exploit any conditional independences. Figure 4.2 shows the belief networks which correspond to a fully observable multi-agent MDP with known

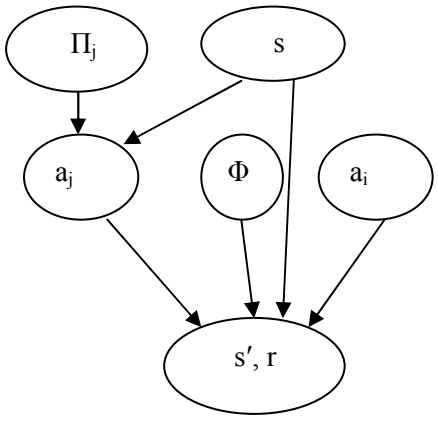


Fig 4.2 (a) Fully observable multi-agent MDP

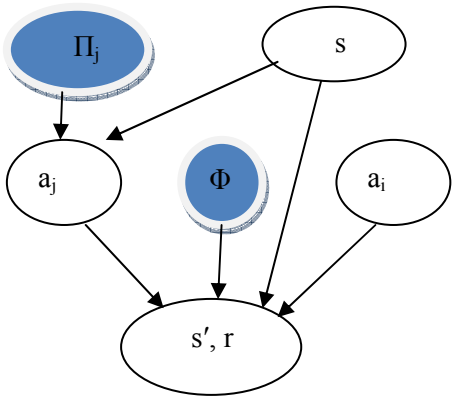


Fig 4.2 (b) Multi-agent MDP with unknown Variables.

Equations describing the likelihood of two unknown nodes in the figure above are as:

$$P(\Phi|obs) \ \alpha \ P(\Phi)P(s', r|s, \Phi, a_i * a_j)$$

$$P(\Pi_j|obs) \ \alpha \ P(a_j|s, \Pi_j)P(\Pi_j)$$

environmental and agent models (4.2 (a)) and with unknown models (4.2 (b)): the latter is the model of [34]. In both these networks, the single leaf is the (state, reward) observations for the new state s′. These depend on my action a_i, the actions of others a_j (which together form a joint action $a_i * a_j$), the previous state s, and the environment

dynamics Φ. My own action is assumed to be independent because its CPT is under my control, but the actions of others depend on the previous state s and on their behaviour models Π_j . The only difference between the two models in figure 4.2(a) and 4.2(b) is what is observed, indicated by the shading. Given the network in 3.4(b), with its two hidden nodes, we can apply the algorithm described above during the update step of algorithm 1 to calculate separately the probability distribution for the unknown node Φ and the probability distribution for the unknown node Φ.

The factorized equations derived by the above algorithm are shown in the below the figure above.

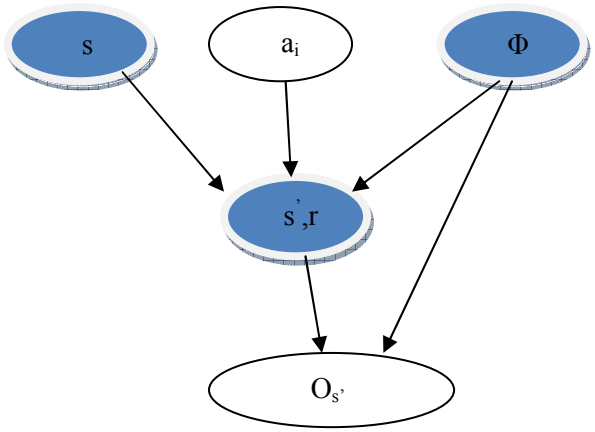


Figure 4.3 Single Agent Partially Observable MDP

Single agent POMDP with update equations as shown in the figure 4.2.

$P(s', r| obs) \alpha \sum_{s,\Phi} P(o's| \Phi, s')P(s', r| s, a_i, \Phi) P(s) P(\Phi)$

$P(\Phi | obs) \alpha P(\Phi) \sum_{s',s} P(o' s| \Phi, s', r)P(s', r| s, a_i, \Phi)P(s)$

Similarly, figure 4.3 shows the POMDP with unknown models described in [30], [31].

Unlike the previous MDPs, this is a single-agent model and so does not have nodes for the actions or behaviour models of other agents. However, we have a new node o′ representing the observations which arise from the current state s′ and corresponding reward, which are now shaded as a hidden variable. We do not have a node representing the previous step's observation because once we have calculated P(s) we consider the observation redundant. Below the figure 4.4 are shown the associated updates. In this POMDP, we do not recalculate the probability for the hidden previous state s, which, being a root node, would come out to $P_{t+1}(s)$ α $P_t(s)$, but only for the current state s′, as well as the environmental model Φ.

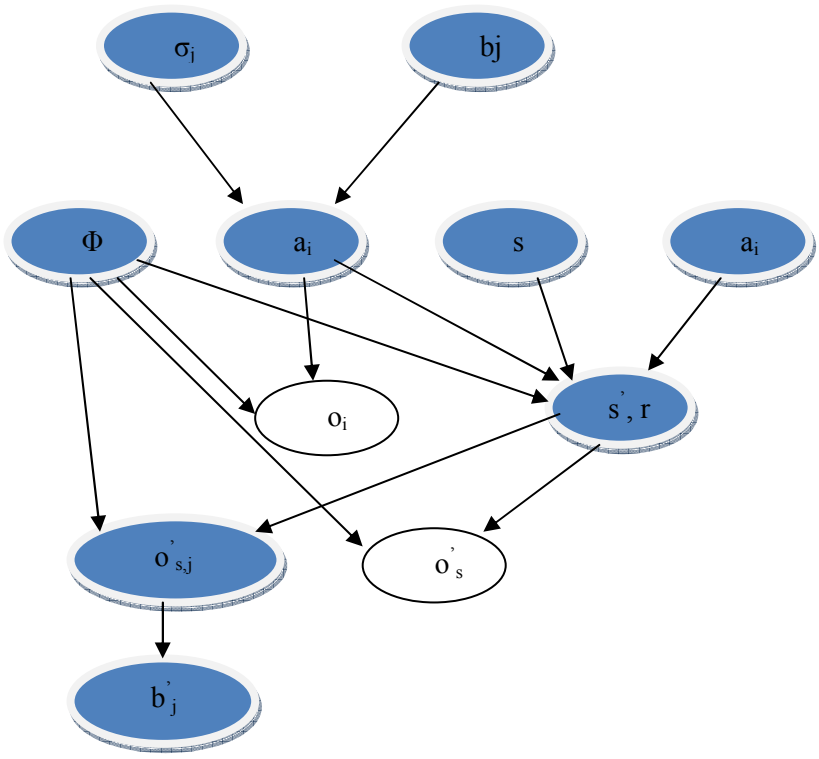


Figure 4.4 Completely Partially observed Multi-agent Network

Finally, figure 4.4 shows the “grand MDP” system described in the previous section, which combines the single agent POMDP of figure 3.5 and the multiagent learning MDP of figure 3.4(b). To help draw together the ideas discussed, we explain this figure in detail. As before, we represent the others by the single agent j. In practice there may be several distinct “agent j”s, each requiring its own set of nodes.

• At the top, the “starting” nodes are the strategies and beliefs of the other agents, the Aj and bj. In general, bj refers to an agent’s beliefs about the current state. These strategies and beliefs are hidden to us.

• Given beliefs about the state, and a strategy, the other agents decide on their actions, the aj. We also decide an action, ai. We may be able to make some observation about the others’ actions, oj.

• Given the true state s (hidden to us), the actions aj and ai, and the environmental dynamics _ (also hidden to us), a state transition takes place resulting in a new state s′ and emitting a reward r.

• The new state is also hidden to us, but we can make some observations about the state, os′.

• Similarly, the other agents will make their own observations about the new state, os′,j. From these observations they will update their belief states to give new belief states b′j. Thus, the core of the figure 3.4(a) contains almost the same structure as figure 3.4(b), but with the aj and s′, r nodes now hidden. we can also see again that when there are hidden variables which cannot be normalised or marginalised away, as in the case of partially observable states, the update step of 1 will have to perform multidimensional summations, making timely updates a challenge in large state spaces. Similarly, the computation of the best response (the Q-values) involves summing over all the hidden nodes and thus the complexity increases with the number of unknowns.

Chapter 5

Implementation Details of Disaster Response Model with Multi- Human agents Ground Operation

In order to simulate the Human Perception model for human agents as proxy software agents and integrate them with the MDP algorithms and online Bayesian learning algorithms described in the previous chapter, along with the hierarchical hybrid agent model described earlier, has been utilized in a disaster response emergency scenario. The scenario has been modified to keep the complexities at minimal and just illustrate the basic working of model with minimum number of all types of agents and some basic environmental conditions which can emerge in that scenario but this model can be scaled up for practical purpose.

Specifically we used the scenario described earlier in which there has been a major terrorist attack on a urban city. In its aftermath, a lot of rescue and disaster response efforts have been going to save the victims, carrying them to hospitals, making availability of ambulances in time, reporting of emerging conditions, identification of threat somewhere as well as eliminate targets responsible for the attack if possible has been simplified to just illustrate the probabilistic logic using Bayesian logic for multiagent learning.

In following section, the simplification made to the observation space in the environment has been described at the same time integrating it as a fully fledged multi-agent POMDP described in earlier chapter.

With this, the next section describes the basic scenario with some simplifications to reduce the complexities.

5.1 Basic Disaster Model Development

In more details, suppose that we have a rescue problem in the aftermath of the attack wherein the region of interest (ROI) has been divided into n * m grid world. K agents of all types can move left, right or up, down (constrained in their movement at the edge of the grid) or dig at their own place through the rubble mass. Instead of focusing on too many instances and sub-scenarios developing in such a case the work just focuses on rescue operation undertaken taken by the agents. In the grid world, there are buried victims, described by two parameters R and D . D ('deadness') is a measure of the proximity of the victim to death . When it reaches a maximum level (which is determined by the human agent, may be medical professional or based on intuition in which case the human agent is likely to make a mistake) the victim is declared dead and subsequently ignored for the purpose of the rescue operation. R ('rescue needed') is a measure of the depth at which victim is believed to be buried. Agents digging can reduce R. If R reaches 0 before the victim dies then the victim is assumed safe. The urgency of the victim therefore increases with increased D and with increased R , unless R is sufficiently large compared to D that, the victim can be considered as completely lost case.

The figure below shows an example grid with three agents making some movements at current time t. The grid is described by 16 <D, R > pairs. The value of D and R are updated based on the observation and their reports sent by the agents to the FTSAs and STSAs which then broadcast the information to the ground agents moving in the area. Currently three human agents are roaming around in the grid area viz. A_0 , A_1 and A_2 . The agent's actions at time t are:

A_0 : Dig A_1: Move Left A_2 : Move Right

After the actions have been carried out, the grid can be updated to show a new state , adjusting the agents locations and <D,R> pairs associated with the grid block.

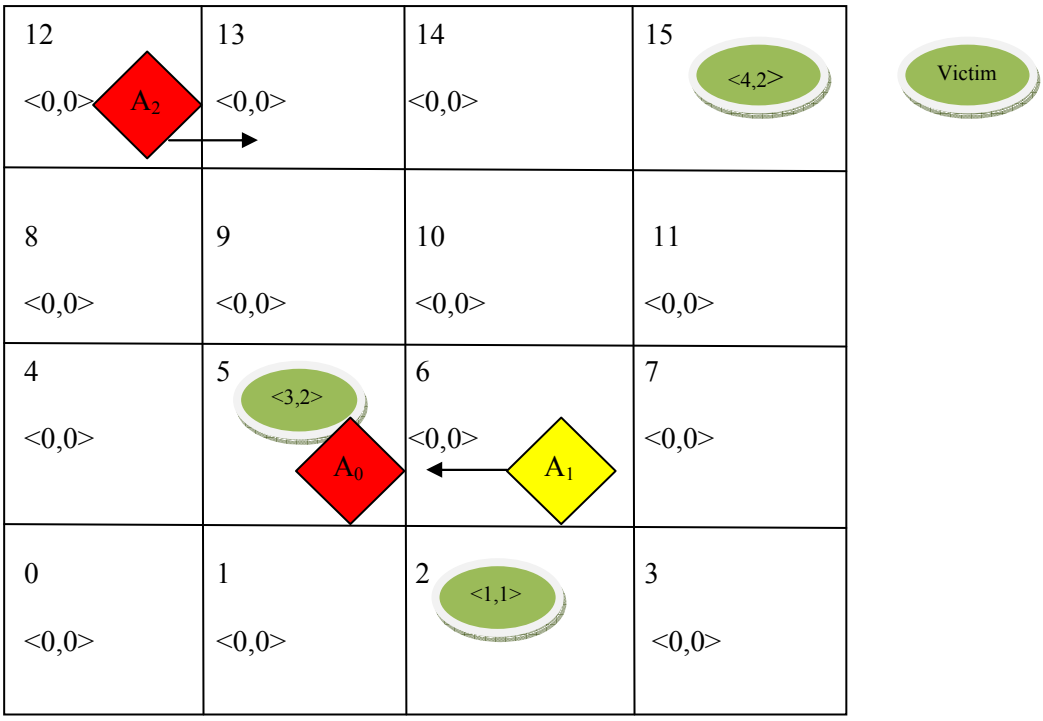


Figure 5.1: One Step of the rescue problem on a 4*4 grid with three agents.

Some of the entities involved in the basic scenario explained above are:

Agents: We assume that the number of agents, k, is fixed throughout each problem (although it can be modified) and known to each agent. In figure 5.1, the set of agents is:

$\{A_0, A_1 \text{ and } A_2\}$

States: A state of the world is defined by using a pair of variables < D, R > for each of the grid squares, characterizing the D and R values in the squares (a simplifying assumption is that there can be at most one victim in the square) and a variable for each agent identifying the current square.

For D and R, l_d and l_r discrete values in turn have been used respectively so for each square region there are $l_d l_r$ distinct states and for each agent there are mn possible states. This suggests that state can be described by a total of mn+k characters where first mn characters have $l_d l_r$ possible values and last k characters have nm possible values making it a grand total of $(l_d l_r)^{mn} * (mn)^k$ possible states. Thus the number of states is exponential in the size of the grid and in the number of agents. In the example in figure shown, there are $16^{15} * 16^3 = 7.55578637 * 10^{22}$ possible states and the current state is :

[0 :< 0, 0 >, 1 :< 0, 0 >, 2 :< 0, 0 >, 3 :< 0, 0 >, 4 :< 0, 0 >, 5 :< 3, 2 > . . .. A0 : 5, A1 : 6, A2 : 12]

j

Locations: The location variable for each agent is its current square. Thus for the example shown the subset of the location is L = {A0 : 5, A1 : 6 , A2 :12 }

Actions: Agents may take Move actions (left, right, up or down), or Dig actions in their current square. This result in five possible actions per agent (we do not admit "null" actions):

A = {Dig, Move left, Move right, Move up, Move down}

Consequently, there are k^5 joint actions: 125 joint actions in the example. Assuming the agent ordering [a0, a1, a2], the immediate joint action is a = [Dig, Move left, Move right].

Observations: An agent observes some subset of the state variables, so there is one observation variable for each state variable. The values taken on by observation variables are those of the corresponding state variable, plus "null", when no observation has been made. Consequently, there are $((l_d+ 1)(l_r+1))^{(mn)} *(mn+1)^k$ possible observations, $25^{16} * 3^{17} = 1.14389695\times1026$ in the 4x4 example shown above.

Transition function: We can consider each of the independent state variables in turn.

• **Agent location** Each agent's location depends only on its own action, and only on its previous location:

$P(L_{i,t+1} = x| \mathbf{a}, s_t) = P(L_{i,t+1} = x| a_i \in \mathbf{a}, L_{i,t})$.

In this problem, Move actions are deterministic, and move the agent one square in the requested direction. If this is impossible because the agent is at the edge of the grid, the action has no effect. Dig actions leave the location unchanged.

• **Deadness in a square with a victim** Each square j transitions (D_j ,R_j) independently of other squares, so it is sufficient to define the transition function for one square. We use a global probability, p_d, to specify the probability of D increasing: this is a constant probability independent of the action: $P(D_{j,t+1} = x + 1|D_{j,t} = x) = p_d$.

• **Depth in a square with a victim** The R level reduces only if there is a Dig action. We assume that if there are $n_d(j)$ digs in square j in the joint action, they are carried out one after another. n_d is a vector function of the state and the joint action (the action specifies which, if any, of the agents are digging, and the state specifies which square these are agents are in). With each of the $n_d(j)$ digs, the square depth (R_j) is reduced by 1 with probability p_r, with a minimum R_j value of 0.

$P(R_{j,t+1} = x - 1|R_{j,t} = x, n_d(j) = 1) = p_r$

$P(R_{j,t+1} = x - x'| R_{j,t} = x, n_d(j) = r) = (1 - p_r) * P(R_{j,t+1} = x - x'| R_{j,t} = x, n_d(j) = r - 1)$

$+ p_r * P(R_{j,t+1} = x - x' + 1|R_{j,t} = x, n_d(j) = r - 1)$ (where $n_d(j) > 1$)

• **Deadness and depth when an agent dies or is rescued** After the joint action has been applied in a square j, we carry out a "tidying up" operation on the (D_j ,R_j) settings. If the square's R_j value has reached zero, then it is assumed that a victim has been rescued from the square. This victim is no longer of interest to us and D_j and R_j are reset to 0. Otherwise, if the square's D_j value has exceeded the maximum health level then it is assumed a victim has died in the square. This victim is again no longer of interest to us and D_j and R_j are reset to 0. This means that the equations in the above two items must be adjusted slightly. Let P ($reset_{j,t}$) be the probability that square j is reset during this "tidying" phase at time t, with

$P(reset_{j,t+1}) = P(R_{j,t+1} = 0) + P(D_{j,t+1} > l_d)$

then

$P(D_{j,t+1} = x) = (1 - P(reset_{j,t+1}))P(D_{j,t+1} = x)$ where $x != l_d$

$P(D_{j,t+1} = 0) = P(reset_{j,t+1})$

$P(R_{j,t+1} = x) = (1 - P(reset_{j,t+1}))P(R_{j,t+1}= x)$ where $x != 0$

$P(R_{j,t+1} = 0) = P(reset_{j,t+1})$

• **Deadness and depth in an empty square** Finally, if a square j is empty at the beginning of the time step, we use a further parameter, p_a, to define the probability that a victim will appear in that square. If a victim does appear, the (D_j ,R_j) levels it has are determined with uniform probability (greater than 0).We define a temporary binary variable, a_j , to determine whether or not a new victim appears in the square j. Then,

$P(D_{j,t+1} = x \mid D_{j,t} = R_{j,t}= 0, aj = 1) = 1 / (l_d-1)$ (Where x = 1, 2, . . . ld)

$P(R_{j,t+1} = x|D_{j,t} = R_{j,t} = 0, a_j = 1) = 1 / (lr-1)$ (Where x = 1, 2, . . . lr)

$P(D_{j,t+1} = 0|D_{j,t} = R_{j,t} = 0, a_j = 0) = P(R_{j,t+1} = 0|D_{j,t} = R_{j,t} = 0, a_j = 0) = 1$

We can apply these functions to the example in figure 4.1, with victims in three squares,

[2 :< 1, 1 >, 5 :< 3, 2 >, 15 :< 4, 1 >] and the joint action [A_0 : Dig, A_1 : Move left, A_2 : Move right], as follows:

• **Agent locations**

Agent A_0 will remain in place: $P(L_{0,t+1}= 5) = 1$.

Agents A_1 and A_2 will each move one square: $P(L_{1,t+1} = 5) = 1$, $P(L_{2,t+1} = 13) = 1$.

• **Deadness in squares with victims**

Squares 2, 5 and 15 contain victims. Note that if the deadness in square 15, D_{15}, increases, both D_{15} and R_{15} will be set to 0. Also, if the depth in square 2 decreases (impossible as there is no agent digging there) then D_2 and R_2 will be set to 0. (We do not show $P(reset_j)$ for the squares where it neither D_j nor R_j is one step away from resetting, making $P(reset_j)$ trivially zero):

$P(D_{2,t+1} = 2) = (1 - P(reset_{2,t+1}))p_d$, $P(D_{2,t+1} = 1) = (1 - P(reset_{2,t+1}))(1 - p_d)$

$P(D_{5,t+1} = 3) = p_d$, $P(D_{5,t+1} = 2) = (1 - p_d)$

$P(D_{15,t+1}) = 0) = P(_{D15,t+1} = 5) = p_d$, $P(D_{15} = 4) = (1 - p_d)$

(where $P(reset_{2,t+1}|\mathbf{a}) = P(R_{2,t+1} = 0|\mathbf{a})$)

• **Depth in squares with victims**

We extract the Dig information from the actions for each square: $n_d(2) = 0$, $n_d(5) = 1$, $n_d(15) = 0$.

Then, $P(R_{2,t+1} = 0|n_d(2) = 0) = 0$, $P(R_{2,t+1} = 1|n_d(2) = 0) = 1)$

$P(R_{5,t+1} = 2|n_d(5) = 1) = 1 - p_r$, $P(R_{5,t+1} = 1|n_d(5) = 1) = p_r$

$P(R_{15,t+1} = 0|n_d(2) = 0) = P(reset_{15,t+1})$, $P(R1_{5,t+1} = 1|n_d(2) = 0) = 1 - P(reset_{15,t+1})$

(where $P(reset_{15,t+1}) = P(D_{15,t+1} = 0)$)

• **Deadness and depth in squares with no victims** All the remaining squares with state values < 0, 0 > have the same functions. We define a temporary binary variable, a_j , to determine whether or not a new victim appears in the square j.

Then, using our settings of $l_d = l_r = 4$,

$P(a_{j,t+1} = 1) = p_a$, $P(a_{t+1} = 0) = 1 - p_a$

$P(R_{j,t+1} = 0|a_{j,t+1} = 0) = P(D_{j,t+1} = 0|a_{j,t+1} = 0) = 1$

$P(R_{j,t+1} = x|a_{j,t+1} = 1) = P(D_{j,t+1} = x|a_{j,t+1} = 1) = 1/4$ where x = 1, 2, 3, 4

Observation function: Agents are able to see the squares (deadness, depth, and any other agents in the square) to the left, the right, above and below them, as well as their own square. Since all agent actions are fully observable, we assume that we can also observe all agent locations. We can consider this analogous to supposing that all the agents have radios, but no time to communicate more than their own position. Additionally, we define a problem-specific parameter, v, for the visibility. For every other square, the agent will be able to see the agent-deadness D in that square with probability v and the depth R in the square with independent probability v. This 'visibility' parameter could be justified as some level of communication with a centralised observer, say a helicopter viewing the scene or a Software agent communication with them broadcasting the information from other region to them at periodic intervals.
We assume no error in the observation: either a variable is completely and correctly observed or it is not observed at all.
Thus, in example above , consider agent A_0 : he observes the positions of every other agent: [A_0 : 5, A_1 : 6, A_2 : 12]. A_0 also observes completely squares 1, 4, 5, 6, 9:
[1 :< 0, 0 >, 4 :< 0, 0 >, 5 :< 3, 2 >, 6 :< 0, 0 >, 9 :< 0, 0 >].

For any other square j with values < D_j,R_j >, A_0 observes j :< null, null > with probability 1 − v and j :< D_j,R_j > with probability v.
Similarly, A_1 observes
[A_0 : 5, A_1 : 6, A_2 : 12, 2 :< 1, 1 >, 5 :< 3, 2 >, 6 :< 0, 0 >, 7 :< 0, 0 > , 10 :< 0, 0 >] and all other

squares with probability v.

Finally, A_2 observes $[A_0 : 5, A_1 : 6, A_{02} : 12, 8 :< 0, 0 >, 12 :< 0, 0 >, 13 :< 0, 0 >]$.

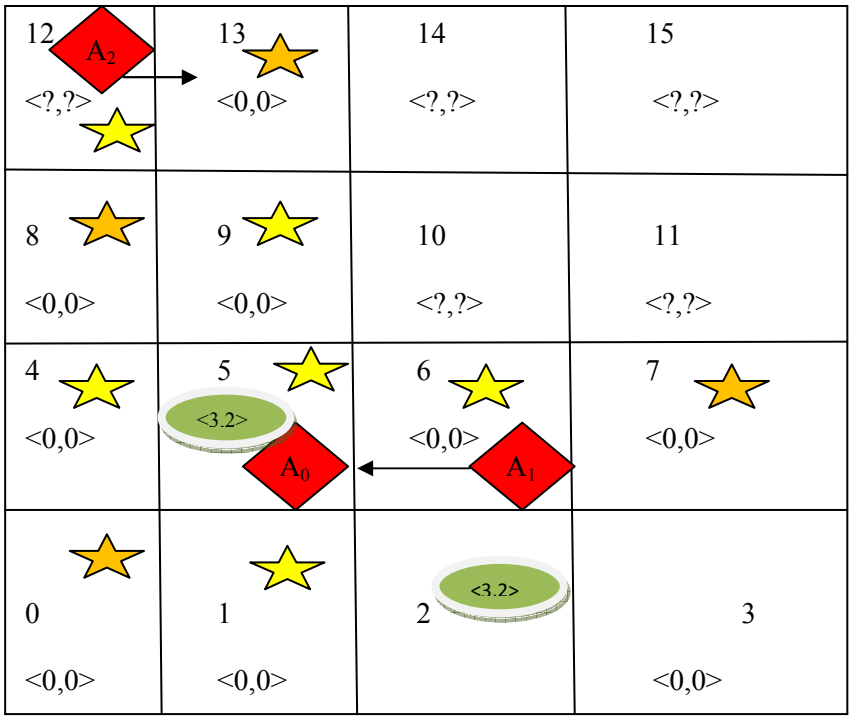


Figure 5.2 illustrates the logic behind observation function.

Figure above shows a possible view for agent A_0 when v = 0.3. The yellow stars represent the squares which are observed completely, because they are near to the agent. The orange stars represent squares which on this occasion have become visible as a result of the global visibility parameter: note in particular that neither square 2 nor square 15 and their victims are visible

to the agent. Finally, the small yellow star by agent A_2 indicates that it is visible as all agent locations are visible.

Reward function: The reward function is a function of both the previous state and the current state. For each square, if a victim disappears because they have died, then the reward is decremented by one point. If a victim disappears because they have been saved, then there is no change to the reward. Consequently, for this problem rewards will always be less than or equal to 0.

In example , there are two victims which may change status between the timestep shown and the next timestep: the victim in square 2 may be rescued (although as we have shown this is actually impossible because there is no agent present), and the victim in square 15 may die. Of these, only the victim in square 15 can affect the reward. We have stated that this victim will die with probability p_d. If it does, then the reward for the timestep will be −1; otherwise, the reward will be zero.

The above definitions allow us to define beliefs over the values (D,R) of a square (and thus over the state, since locations are observable), and beliefs over the observations of other agents, given their locations:

Agent locations: We are certain for all squares how many rescue agents they contain for all agents where they are located

The square is observed: We are certain of both its parameters The square is not observed and has not been observed for t_i timesteps:

$$P(x_t = v_t|x_{t-i} = v_{t-i}) = \sum_v P(x_t = v=t|x_{t-1} = v)P(x_{t-1} = v|x_{t-1} = v_{t-i})$$

where the 1-timestep probabilities depend on p_d, p_r, p_a as appropriate, and the dig observations in that square.

The square has never been observed: This is just as above, but with $P(x_0 = v_0)$ set to the problem- specific prior probabilities. Here, we assume that all squares are empty to begin with. Given these equations, and using its environment model, our agent can calculate probabilities (beliefs) for each of the state variables. In all of our experiments, we assume that the agent knows the environment model, which is the model described on the previous pages. These probabilities form our agent's belief state: that is, the beliefs about the D and R values of the squares and the locations of the other agents. Similarly, we must define our beliefs about the observations of the other agents. Just as our beliefs about the state of each square are multinomial, the other agents' beliefs about the state of the square will be multinomial. Therefore, in the full POMDP model, our beliefs about other agents' beliefs over the state of the square would take on corresponding Dirichlet distributions. However, we are not trying to maintain beliefs about the other agents' belief states, only about their observations. Now, our own beliefs about the state of the square define exactly what we believe other agents will see if they see that square, as the observation function is deterministic and consistent for all agents. Because we know the location of the agent, we know of the (up to) four surrounding squares it definitely sees. Finally, we know that there is a v probability it will see any other square.

Now, the problem described above is used as given for experiments involving partially observable states. However, we modify the problem slightly in order to investigate specific aspects of our general model. Specifically, problems where it is the actions rather than the states which are partially observed, can be investigated, and problems which are dynamic or open, can also be dealth upon . In order to handle these cases, we can make slight modifications to the problem described below.

Firstly, three modifications apply when actions are partially observable: Removing agent locations from the state. Although the agent locations are a part of the state, in one subproblem

we choose to treat them separately, making them only partially observable while the state remains full observable. Technically, this is a partially observable state problem. However, providing the actions are deterministic, deductions about the agent locations, depend only on the agents' choice of actions.

Uncertain actions. We can add some uncertainty to the outcome of an action: with probability m_p, the intended action is carried out, while with probability $(1 - m_p)/4$, a random action (selected from all five possibilities including the intended action) is carried out.

That is, an agent intending to move (such as A_1 and A_2 in the example in figure 5.1) may find himself moving in the wrong direction, or shuffling on the spot and inadvertently digging. Similarly, an agent intending to dig (such as A_0 in the example in figure 5.1) may unexpectedly slip into a nearby square without digging.

Penalising actions. In order to investigate how agents can use the reward function to make inferences about actions, we can adapt the problem in such a way that the choice of action affects the reward directly. In this adapted problem, we give moves a cost of 0.1 points, representing the fact that each move uses up some of an agent's resources. We also penalise Digs which take place in an empty square, giving them a cost of 0.5 (Digs in a square with a victim incur no cost). These penalties are applied to the intended action, not the outcome, forming an analogy with the effort the agent must put into the action.

Consider the example in figure 5.1 in the light of such a penalising function, with a visibility parameter of 0: agents A_1 and A_2 will each incur a cost of 0.1, so that the total reward for the step is −0.2 if the victim in square 15 survives, and −1.2 if the victim in square 15 dies. Every agent will observe this reward and consequently deduce that two move actions were made (since no other combination of actions could cause this fractional reward) and one

Dig action. A_0 and A_1 can observe each other and so will both know that A_0 carried out the Dig

action and therefore A_0 made a move action (although not in which direction). A_2 which cannot observe either of the other agents can update beliefs about the agents—for example, if A_2 has observed square 5 recently and seen the victim there, then they may believe that an agent in square 5 is likely to dig. Furthermore, if A_2 can see square 6, they will know that the agent in square 6 did not Dig, or they would have incurred the penalty.

Now, the above discussion assumed that all actions were successful, but if the action penalties are combined with action uncertainties, then A_0, who performed the Dig, will know that A_1 and A_2 both intended move actions, and will know whether A_1 achieved a move. However, A_1 will no longer be certain whether A_0 intended a Dig or a Move. Consequently, A_1 will have to update his behaviour models for each agent assuming that A_0 intended a Dig with probability mp and a Move with probability $(1-m_p)$, and vice versa for A_2. In general, agents will not be able to make such precise deductions about the intended actions of others and will have to apply similar probabilistic rules for many of the others when learning about their behaviour. Secondly, as well as partially observable actions, we will be investigating our model in the context of dynamic environments, and open environments. In the problem described above and depicted in figure 5.1, we have not explained how such environments are included. In more detail, dynamism occurs when the environment changes during the course of the problem. In the disaster problem, the change may be to the value of any one of the parameters. Here, however, we can also investigate dynamism in the arrival rate and death rate parameters; and in the move penalty value. Openness refers to agents appearing or disappearing during the course of the problem.

To investigate dynamism, we will change problem parameters such as p_r or p_a at a timestep t. We assume the other agents know about all changes. The belief state at time t forms the prior for the belief state at time t + 1, but the belief updates at t+1 use the new parameters. No further changes are necessary. To investigate openness, we will introduce or remove agents after some number of

steps. Again, all agents in the system know about the changes instantly. When new agents are added to the system, they will be placed in the same initial location at square 0. New state and observation variables must be added describing the locations of every new agent. When agents are removed from the system, the corresponding state and observation variables must be removed from the state.

The basic simulation was done using Java program based multi-agent project in simulation. The simulation showed that the algorithm mentioned above converges fast with additional number of agents. The snapshot in figure below shows the timeline of agents' action and victim being saved as the time progress. The sum total of rewards accumulated by an individual agent is the main factor which drives the agents towards rescuing the victims. This model can be enhanced further when integrated with the information communication between various agents, so that agents moving to geographical regions are penalized for fruitless actions. If an agent moves to a region, where victim has already died or has been rescued, then there is no point moving to such a region and so either it has been misinformed or deliberately it has moved due to wrong behavior. Every agent needs to maximize its reward without getting penalized and this maximizes the overall rescuing of victims trapped under the rubble in the problem mentioned above.

```
Timeline describing the agents action:
Time slots (in microseconds)
0:00:00: Simulation starts
0:00:01: Agent A0 DIGS LOC:5,A1 MOVES LEFT LOC:5,A2 MOVES RIGHT LOC:13
0:00:02: Agent A0 DIGS LOC:5,A1 NO ACTION LOC:5,A2  NO ACTION LOC:13
0:00:03: Agent A0 DIGS LOC:5,A1 DIGS LOC:5,A2 MOVES RIGHT LOC:14
0:00:04: Agent A0 DIGS LOC:5,A1 DIGS LOC:5,A2 MOVES RIGHT LOC:15
0:00:05: Agent A0 NO ACTION LOC:5,A1 NO ACTION LOC:5,A2 MOVES RIGHT LOC:14
0:00:06: Agent A0 MOVES DOWN LOC:1,A1 MOVES DOWN LOC:1,A2 MOVES RIGHT LOC:15
0:00:07: Agent A0 MOVES RIGHT LOC:2,A1 MOVES RIGHT LOC:2,A2 MOVES DOWN:11
0:00:08: Agent A0 DIGS LOC:2,A1 DIGS LOC:2,A2 MOVES DOWN LOC:7
0:00:09: Agent A0 DIGS LOC:2,A1 DIGS LOC:2,A2 MOVES DOWN LOC:3
0:00:10: Agent A0 DIGS LOC:2,A1 DIGS LOC:2,A2 MOVES LEFT LOC:2
0:00:11: Agent A0 NO ACTION LOC:2,A1 NO ACTION LOC:2,A2 NO ACTION LOC:2
0:00:12: Agent A0 NO ACTION LOC:2,A1 NO ACTION LOC:2,A2 NO ACTION LOC:2
0:00:13: Agent A0 NO ACTION LOC:2,A1 NO ACTION LOC:2,A2 NO ACTION LOC:2
0:00:14: Agent A0 NO ACTION LOC:2,A1 NO ACTION LOC:2,A2 NO ACTION LOC:2
0:00:15: Agent A0 NO ACTION LOC:2,A1 NO ACTION LOC:2,A2 NO ACTION LOC:2
0:00:16: Simulation stops
Result:
Victims saved :2 Rewards collected : -.14
```

Chapter 6

Conclusion and Future Research Goals

6.1 Thesis Summary and Conclusion

This thesis has introduced the concept of human perception model (HPM) for modeling the human agents based on various special characteristics of humans. Besides, this it then goes on to incorporate these HPMs in Hierarchical hybrid model of agents for driving the search and rescue operations in case of disaster or emergency large scale operation over a region.

After introducing this framework for agents, a Bayesian approach along with Markov decision process modeling (MDP) and Partially Observable Markov Decision Process modeling (POMDP) is described for scalable multi-agent learning for better coordination so as to optimize the rescue and search operation and maximize the number of victims who can be saved in case of disaster hitting over a geographical region.

Typically, in a large multi-agent system, such as disaster scenario or military operation, no agent will be able to see the entire scene of be certain about how the other agents are viewing the scene or planning to behave. This is particularly true if communication is limited, such as in disaster scenarios in which time constraints or network failures may limit communications or military operations where secrecy is important. In order to coordinate with other in such scenarios, agents must make inferences about one another from their own observations and respond accordingly.

Markov Decision Process (MDP) was chosen for implementing the framework as compared to other research in this domain. This provides formal building blocks on which a principled approach can be built meaning that there is a framework available with guarantees in the system.

In particular, Bayesian MDP models such as [33],[34],[35], in these authors offer a principled solution to the exploration-exploitation problem within unknown systems. However, each of these

approaches is limited to specific use. The first contribution of this work is to develop an approach to this problem using a Bayesian Learning mechanism generalizing the previous work on learning models of other agents.

Furthermore, due to the difficulties associated with the inherent complexity, the previous approached were not scaled up to the large systems of interest. One approach uses finite state machines to model other agents [36]. This insight was used to develop an approximate scalable algorithm applicable to general disaster model described and analyzed in Chapter 4.

This algorithm is developed on the fact that all the agents are fully aware of the action in their vicinity but are uncertain about the behavior of others.

This algorithm can be modified and enhanced further when agents can see the actions of others but are not fully aware of their immediate state and so are not sure about the belief of other agents. Finally, a more realistic algorithm can be developed based on POMDP in which

6.2 Research Contribution

The main contribution of this thesis is a new concept for modeling humans as Human Perception Models (HPMs), design of an integrated hybrid hierarchical model for enhancing the collaboration of human assisted team in disaster response operation carried out over a large geographical region and finally, this thesis outlines a multi-agent learning algorithms for team coordination for optimizing the output of rescue and search operation in aftermath of natural or terrorist attack.

In more detail, previous work [34],[35],[32] has described online Bayesian learning models for a number of variants on partially observable systems, evaluating these models on small problems. In this thesis work, the model has been extended to a generalized model which can be applied to arbitrary partially observable multi-agent systems. However, if nothing is observable then will be

obviously impossible to learn anything. Bayesian Network diagram can be useful in visualizing how to apply our model to any particular case of the system, identifying the dependencies between the hidden and observed in order to write down a factorized belief update.

For solving Bayesian MDP model, the agents must evaluate the Bayesian belief updates, which may be non-trivial and then solve a best response equation over all the belief states. To reduce the complexity for solving this equation, statistical clustering can be used which reduces the number of states and observation space in to more compact higher level cluster space. This approach has not been previously been treated within an explicit Bayesian framework and this work demonstrates the efficacy of the Bayesian approach within this model.

Appendix

INTEGRATION OF WLAN BASED ADHOC NETWORK WITH ALREADY EXISTING 3G NETWORK FOR NON-COOPERATIVE INCENTIVE BASED SCHEDULING

This work has been put here in this thesis as extra research work which has been done for integrating the sensor ad-hoc network formed in disaster response scenario where a ad-hoc network using 802.11 based wireless link needs to be formed. This is because, if we rely just on 3G cellular infrastructure then, there are chances of network being clogged or heavily congested and some of the users may not be able to communicate properly. Generally in such a network, which is WLAN based, although the short range communication between nodes is generally good, but in case, if one or more of the mobile unit nodes are facing low fading situation then existing 3G cellular infrastructure can be used. Such an integration well help in making the communication between various heterogeneous agents hassle free on which necessary for proper functioning of sensor adhoc network.

The following sections describe the background behind Unified Cellular Adhoc network and he changes which has been to the proportional fair scheduler in 3G CDMA EVDO based network Base Station for encouraging all the users to forward data for each other in case one of them is in low fading situation.

INTRODUCTION

Wireless Networks fall in two main categories: Ad hoc networks which are formed based on the need for communication without dedicated infrastructure in place and Cellular Networks with infrastructure in place. This study encompasses integrated architectures for wireless relay Networks like ICAR and UCAN and the relay network formation algorithms and scheduling algorithms used in these types of architectures. The advantage of these types of multi-hop networks has been established in various research studies[37], [38], [39].

Relaying allows multiple wireless hops for data to reach its destination in cellular mesh networks. The same concept is utilized in 802.11 based wireless ad-hoc relay networks where a node facing low channel conditions and link rate can get his data forwarded through some other relay node in its vicinity to the Base Station or next relay node. The short distance relaying of data increases the throughput of the individual nodes and also there in an increase in overall throughput and decrease in end-to end delay due to data being transmitted over faster links. Another major advantage is low power consumption due to data forwarding over shorter links. The formation of these types of relay network formation has been along with the selection of Gateway (GW) node which acts as the node representing this relay network to Base Station has also been discussed in [40], [41], [42].The scheduling of these GW nodes is done by the base station in a proportional fair manner either based on standard CDMA1x EVDO scheduler or on the basis of number of nodes represented by individual gateway node.

The relay network (RN) formation can occur either on single frequency in which all the hops between the various nodes involved in RN can exchange the information over that frequency as described in [37] or the formation of relay network can occur on multiple frequency on a spectrum as announced by Base station as described in [41]. This is termed as Multi frequency multi hop Wireless relay Networks in which the mobile/semi-mobile nodes have two wireless interfaces the ad-hoc 802.11 interface and the other cellular (HDR) interface between. The switching happens between two interfaces as and when required.

In this work, the various relay network formation algorithms in UCAN and [42] will be first introduced and then how modifications can be done to improve the performance of these algorithms and along with ways to conserve more battery power will be shown. Next a new packet relay mechanism will be shown for multihop uplink packet scheduling and Matlab simulation has been done for it and results obtained.

The relay network formation can occur in cooperative and non cooperative mode wireless network scenarios. The report also encompasses scheduling algorithms run by Base Station to in these scenarios and outlines the dominant game strategy established on the basis of Nash Equilibrium in cooperative and non cooperative cases. For non-cooperative case, in which Proportional fair scheduling do not works as every node is selfish and does not volunteers to relay traffic for other nodes unless some incentive is assigned to them for encouraging them for participation as relay nodes , a Mathematical modification has been proposed on the parameter used for scheduling in non-cooperative network and performance benefit has been mathematically derived for this scheduling approach in terms of individual node throughput and overall throughput of the network.

The mathematically derived scheduling parameter has then been simulated using C programming and the effect of it has been demonstrated on the throughput of a node which is willing to behave cooperatively, a node which is willing to behave cooperatively and being able to qualify as a relaying node on the conditions set by the Base Station.

The throughput has also been calculated for a non-cooperative node which do not relays traffic for other node.

The security issues have not been explored in the context of the benefits which Base Station assigns to the relaying and cooperative nodes as this can be taken advantage of nodes which are malicious and can volunteer to forward nodes in return for certain incentive from the Base Station but then don’t forward and drain their battery power and resource for other nodes.

A BRIEF OVERVIEW OF UCAN ARCHITECTURE & MULTI FREQUECNY RELAY NETWORK FORMATION ALGORITHM

UCAN Architecture Overview

UCAN (Unified Cellular and Ad hoc Network Architecture) has been described below as the modifications have been proposed relay network formation, uplink packet scheduling and incentive based scheduling algorithm are based on this architecture only.

The UCAN architecture is based on the key idea of opportunistic use of the IEEE 802.11 interfaces to improve the 3G wide-area cell throughput as over short distances (250 m) 802.11 wireless connectivity offers data rates of the order of 11Mbps. The UCAN architecture is as shown below:

Figure A-1: 3G Cellular Network integrated with Wireless LAN Sensor Adhoc Network

As depicted above the mobile clients involved in this architecture viz. Destination client, Relay Client and Proxy Client exchange data packets with each other and with the HDR Base Station. The clients monitor the downlink data burst from the BS and estimate their current downlink rate. Based on the channel rate conditions some of the nodes can volunteer to act as Proxy Client

(Gateway node) for the relay network and are then termed as cooperative in nature by the Base Station. The nodes which finally relay traffic for other nodes are termed as Relaying nodes and may be given incentive by the Base Station in return for it to motivate them to keep on participating in the formation of relay networks.

When a mobile client which is actively receiving HDR data frames faces low downlink data rate situation, it sends a RREQ message to the intermediate relay clients which forwards it to the Gateway Node for that relay network which finally then forwards it to the Base station. BS then forwards the packet destined for the Destination client to the Gateway node instead of directly sending it. The data packets are forwarded to the destination client through IP tunneling via intermediate relay client using high bandwidth 802.11 links.

Various relay network formation algorithms and Gateways discovery algorithms has been proposed for single frequency relay networks and Multi Frequency relay networks in [37],[38].In this report the details of these algorithms are not mentioned but a brief overview is provided.

For single frequency Relay Network formation UCAN has two approaches:

- Greedy Proxy Discovery algorithm: In this mode every node maintains immediate neighbor average downlink channel rate conditions. When RREQ message is issued, it is unicast to the neighbor with the highest downlink channel rate. The messages then greedily traverses through a set of relay clients and finally to the HDR base station.
- On Demand Proxy Discovery algorithm: In this mode the destination client broadcasts the message to all the neighbors within a given range so this is reactive in nature. The neighbors with high channel quality then contend to serve as proxy by sending application message to the HDR base station.

Scheduling in UCAN is based on HDR proportional fair scheduling in which clients are scheduled based on minimum $\{T_k(t)/R_k(t)\}$ where numerator is the moving throughput average and the denominator is instantaneous downlink rate for the client. A client is scheduled when its downlink channel rate is high is order to improve the overall downlink throughput while short term fairness in terms of client's throughput is also considered.

For Relay Networks when a node facing poor downlink rate is served by a relay node its rate increases , but since BS has to maintain fairness between the relay node and node whose data is being relayed, the scheduling is done of the basis of destination client's rate and not on the basis of relay node rate. But the data transfer for the destination client does occur at the relay node's rate and so this leads to increase in throughput for the destination client but at the same time relay node is allotted more scheduling slots in comparison to the destination client node leading to fair scheduling by Base Station.

This scheduling approach of PF is based on assumption that the nodes participating in the networks are cooperative. In non cooperative networks selfish nodes will not participate and forward data for other nodes and so another approach is required to motivate them to join in formation of RNs. This modifies approach and the modifications made to the scheduling parameter and increase in throughput due to that will be discussed in next section.

Multi Frequency Relay Network Formation

For multi frequency Relay Network formation some algorithms has been proposed in [42] as mentioned below. In these schemes the relay networks operates over multiple frequencies. Base Station can schedule large number of nodes facing poor channel conditions and so to aid in the formation of such networks, it advertises the available bands over which the these RN should form and also assists later in assignment of orthogonal frequencies to the various links in these RNs. The formation of these types of RNs is done first by Gateway discovery and then making the nodes join the relay networks and learn the route to BS through intermediate relay nodes.

- GW Discovery: Every node forming the relay network periodically broadcasts a neighbor advertisement (NADV) message over the control channel as soon as it receives new frequency band information. The NADV contains identification of the node and the downlink channel link rate received by that node. The neighboring nodes on receiving this messages compare it with their own received signal quality and if it has better value than all one hop neighbors then

  It assumes the role of GW node.

- Route Discovery algorithms like Baseline, in which all nodes initiate route discovery just after GW discovery without following any particular scheduling as a result leads to congestion in the relay network during formation phase because every node send RREQ almost simultaneously.

  To avoid congestion problem, algorithms like FF (Farthest First), NF (Nearest First), and LOF (Locally outmost first) has been suggested in [e].

- LOF Algorithm is a distributed algorithm in which achieves the benefits of FF and NF and low latency of the Baseline.

This allows each node to make a schedule for its route discovery based on relative distance of this node from the BS in comparison to GW node. The outermost node is local to the relay network and its discovery is clubbed with the GW discovery. After RN is formed, BS broadcasts the relay network information to all the nodes in the network which then aids the nodes in calculating their relative distance from BS with respect to the GW node.

Based on this information the nodes are able to calculate their scheduling time to initiate their route discovery and some of the nodes learn the route passively.

- Formation of RNs is followed by frequency assignment of each link between individual pair of nodes. Based on the spectrum allotted to the relay network by the BS, orthogonal select frequencies are assigned from the common pool of the frequencies available and so

the goal is to avoid interference and at the same time allow multiple exchanges of information packets. If the frequency pool gets exhausted then a node can choose the same frequency which has already been allotted but keeping into consideration the frequencies being utilized by nearby one hope neighbors.

MODIFICATIONS SUGGESTED IN RELAY NETWORK FORMATION

Algorithms for formation of relay networks can be enhanced with cross-layer optimization of several parameters like residual energy of node, delay involved, end to end throughput and interference. Several constraints can be considered on individual node level basis and overall relay path basis.

The node constraints can be categorized as cooperation level, interference level, residual energy, and sufficient neighborhood connectivity.

The end to end connectivity and throughput, latency are other parameters which can be categorized for relay path analysis.

To make the analysis more concise, only sufficient neighborhood connectivity and interference level in the vicinity of nodes have been considered. So in all the Relay network formation algorithms outlined in the previous sections, whenever a node accumulates downlink channel conditions of its one hop neighbors, it should have a list of prospective relay nodes and maintain virtual connection with them. To maintain virtual connectivity amongst themselves they should keep on exchanging “KEEP_CONNECTION_ALIVE” messages at regular periodic intervals. This is required since due to mobility of nodes and high variability of fading and shadowing effects the downlink rate keeps on varying and also with time the energy level of individual nodes keeps on decreasing.

The virtual connectivity helps in avoiding the unexpected call drop, since many a times the node which is relaying the traffic for the destination client has no available bandwidth or its energy simply nullifies. To prevent this if the destination client and all other nodes have information of alternate relay nodes in their neighborhood even if it is multiple hops ahead, a switching can be done by the node forwarding the data to the relay node can itself switch without waiting for the Base Station to step in. Of course since a couple of data packets may have been lost due to dropping of connection so every node has to buffer the past data for a particular number of time slots if it is participating in the relay network formation and so on switching to different node it will start transmission from some previous time slot. This approach applies in case of non-real time transmission of traffic but in case of voice call this approach cannot be applied where dedicated connectivity like circuit switched network is required.

SUFFICIENT NEIGHBORHOOD CONNECTIVITY CRITERIA:

For sufficient neighborhood connectivity in wireless networks of n randomly placed nodes each node should be connected to O (log n) neighbors.

If a node has less than .074(log n) then node is asymptotically disconnected and if node has greater than 5.1744(log n) then node is asymptotically connected. So if a node has k nodes then for sufficient connectivity k should be greater than 5.1744(log n).

Based on this criteria during relay network formation, virtual connection should be established between various nodes and aim should be to achieve this criteria so that connectivity is not lost in case of relay node losing its battery power or bandwidth.

MINIMUM INTERFERENCE CRITERIA:

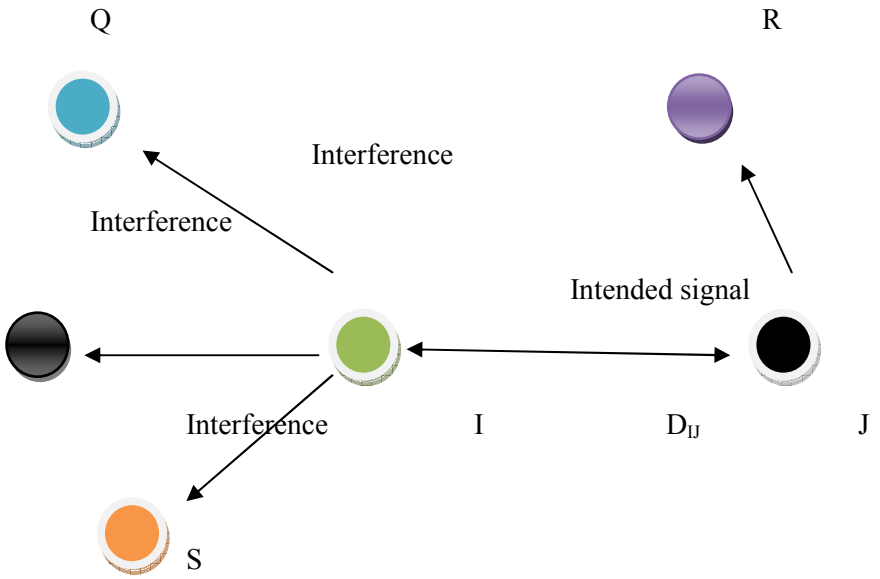


Figure A2: Interference among mobile and fixed nodes using CDMA based wireless communication

Based on the above, due to ongoing communication between node I and J there is interference on the link between node pairs (I,Q) , (I,T) , (I,S) and (J,R). The total interference received in the wireless networks due to I, J communicating is:

$$\text{Interference} = (1/G) \sum [(\alpha^2 P_{(i,j)} / D_{i.r}^{\square}] \text{ ---------- (1)}$$

In equation (1) we have α as the correlation coefficient between the waveforms on which communication occurs between two pairs of nodes. In CDMA also the orthogonal codes are not purely orthogonal as there is correlation between them.

$P_{(i,j)}$ is the Power level of the transmission between the two nodes I and J and $D_{i,jr}$ is the distance between the two nodes I and R. and G is the processing gain.

If transmitted power is adjusted at node I such that destination node J receives a power P_{ref} - i.e.

$P_{ref} = P_{(i,j)} / D_{i,j}^{\square}$ then can modify the equation (1) to get:

$$\text{Interference} = (1/G) \sum_{r=1,\, r \neq \{i,j\}}^{n} [(\alpha^2 P_{ref} D_{i,j}^{\square}) / D_{i,r}^{\square}] \text{ ---------- (2)}$$

So based on the above criteria above, we should have three major modifications in the relay network formation algorithms discussed so far in the previous section. Firstly a set of cooperative nodes is formed when BS broadcasts who all nodes are willing to behave cooperatively and participate in the formation of Relay Network. Then out of these nodes sufficient neighborhood criteria needs to be satisfied and last but not the least Interference quantity as derived in (2) needs to be minimized.

UPLINK RELAY NETWORK FORMATION BASED

For relay networks, the previous section dealt with the formation of relay network based on the Downlink channel condition, (link rate) received by individual node based on bursts of data received by them from Base Station. But we didn't consider the uplink channel condition for the one hop neighbors. If that parameter is also considered for while forming the relay networks then any uplink message can also be transmitted with considerably reduced latency and hence can also aid in increased throughput. The fast uplink access to Base Station can lead to more prompt centralized coordination by the Base Station. The following algorithm for uplink packet forwarding applies for cooperative networks.

So during the time when NADV messages are exchanged if two more parameters (distance from the Base Station and Uplink channel rate) concerning uplink channel conditions are included in those messages then it can help in establishment of a virtual Uplink relay network so that if a node has to send a control channel message to Base Station in case downlink failure occurs, it can send it on the uplink relay path to the Base Station.

The following uplink relay network formation algorithm assumes that every node is getting updated information for the two parameters at regular periodic interval.

THE ALGORITHM:

1. Every node maintains updated uplink information which consists of parameters like Base Station distance and uplink channel link rate regarding the potential nodes in its vicinity.
2. The NADV packets contain this information and are exchanged at periodic intervals between the nodes at one hop distance.
3. Whenever a node wants to send a control packet to Base Station, it scans either the relay nodes or Base Station in its range. If no entity is in the range, then it sits idle and does nothing. If it finds a relay node which is closer to the BS and has better uplink condition than other one hop neighbors its forwards that packet to that node otherwise if finds the BS in its range of transmission (one hop) , it forwards that packet to the BS directly.
4. Relay selection among potential neighboring nodes can be done on the basis that all those relays in hop circle range are checked for and the relay which is closest to Base Station and having better link rate is selected.

   If $D_{i, BS}$ is the distance between relay node I and BS and $D_{J, BS}$ is the distance between relay node J and BS, and both lie in the range circle of the node which wants to send control channel message to the Base Station. Further R_I and $R_{J,}$ are uplink channel rates of node I and J. Then following Pseudo algorithm can be used to select the Relay Node:

If (($D_{i, BS} < D_{j,BS}$) && ($R_I > R_J$))

select Node I

else if (($D_{i, BS} > D_{j,BS}$) && ($R_I > R_J$))

select Node I

else if (($D_{i, BS} > D_{j,BS}$) && ($R_J > R_I$))

select Node J

else if (($D_{i, BS} < D_{j,BS}$) && ($R_J > R_I$))

select Node J

A modified version of this algorithm has been simulated. The modification is based on the position of the node which wants to send uplink control information , it has to make a switch between selecting a fast link relay node or a node which is closest to the node. In the above pseudo algorithm channel rate is the dominating factor but if node is close enough then it needs to select a relay node which is closest to Base Station so that the packet instead of looping around when it is close to its destination goes straight to it.

Nodes are distributed randomly in 4*4 km rectangular region and they are mobile moving in arbitrary directions and so their link rate also varies in every other time slot. The goal is to maintain a relay path for a particular node facing fading situation over a longer period of time using one hop neighbor having best link rate or closest to BS. For a far-off node from BS in absolute fade, but nevertheless in a dense network so as to find plenty of neighbors ready to relay its data. First such a node will forward its data to a node having fastest link and closest also to the BS, if not this then node with fastest link and in its transmission range. Transmission range of nodes also varies with Battery power variation. Battery power varies very slowly when the UE is

not active and varies with larger amount when it transmits. After getting closer to the BS after a certain limit , the node will only look for other nodes which are closer to BS and ultimately it will reach BS which will lie in the transmission range of the end node.

This approach is required since in the end also if a node is looking for high rate link then may be the packet will get into a loop as node at greater distance from BS may be in good situation than a node closer to BS but in congested place. This algorithm serves to form a uplink relay network for transmitting control channel message to the Base Station whenever required

Base Station is assumed to be at (0,0) coordinates and all other nodes are in positive quadrant moving around. If (x_BS^2 +y_BS^2)^(1/2) is less than Transmission radius then in range of the Node and so we will print out the relay path. This relay path will remain valid till all the nodes forming the path are in transmission range of each other. The node0 is assumed to be in deep fading situation is assumed to be at 4,4 at far off distance and it has to establish a Uplink route to the Base Station

SCHEDULING ALGORITHMS FOR COOPERATIVE AND NON COOPERATIVE WIRELESS NETWORK

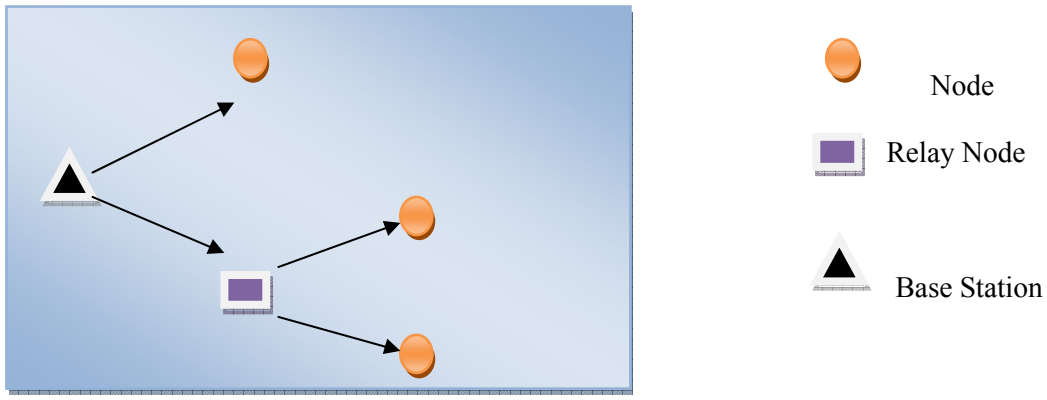


Figure A3: Figure showing Base station scheduling relay and ordinary nodes

We know that all standard scheduling algorithms like C/I(Carrier Power to Interference Noise Power ratio) based scheduler, Proportional Fair Scheduler (PF), Round Robin(RR) scheduler. etc. perform quite well in cooperative networks and encourage the nodes in joining the relay network formation where non-selfish nodes are willing to relay traffic for other nodes without any credit being offered for this behavior.

In non-cooperative wireless networks, nodes operating in the system are selfish in behavior and do not forward data traffic for other nodes as the main objective is to increase their own utility function. They can be made to act a s relay nodes if paid come credit or incentive by the Base Station at the cost of the node which wants to get forwarded its data traffic .

Even if we consider Fair scheduling in Cooperative network, then relay nodes should be allocated three times the bandwidth and radio resources than a node which is directly connected to the Base Station as shown below in the following figures:

From the above figure we can see that only if BS allocates 3X resources to a node which is directly connected and is not a relay node then, the Relay node is at loss in comparison to that node even though it is also directly connected to BS.To maintain fairness across whole network, either Relay node should be allocated 9X resources so that the two end nodes to which it is relaying traffic also are treated fairly by BS or only the RN be allocated 5X resources so that nodes directly connected to BS are at advantage then the end nodes but in case Relay node needs to be compensated for forwarding traffic of other nodes.

Based on this we can think that in a non cooperative network where every node is trying to maximize its utility function in terms of throughput or any other valuable output, all selfish nodes will work for only themselves and not for any other node. So existing schedulers like PF, RR and C/I are not able to implement fairness in non-cooperative environment consisting of selfish nodes as the utility function in PF and C/I is to maximize throughput and in RR there is no control over throughput by individual node as all nodes are fairly allocated slots by BS without taking into consideration any other individual node parameter.

MODIFICATION DONE IN PF SCHEDULER FOR NON-COOPERATIVE RELAY NETWORKS AND MATHEMATICAL DERIVATION

In Proportional fair Scheduler, time slot allocation is done by the Base Station based on the instantaneous data rate R(t) supported as well previous throughput average over a certain period consisting of window of fixed number of slots , T(t). Time slot is allocated to the user with the

scheduling parameter (SP) = MAX(R (t) /T(t)) for all nodes present in the network. In non cooperative environment as explained in the previous section selfish node in order to maximize their own throughput will not forward data packets for other nodes and so no relay network formation will occur.

Also we have seen that Relay nodes needs to be compensated and allocated more radio resources or given some incentive or credit to carryout relaying in non cooperative environment.

To encourage Relay nodes to continue participating in the Relay network formation and other nodes to join relay network and behave cooperatively, a modification is done in the SP as shown:

(SP) = MAX(C*R (t) / (T (t) +B (t)) ---------- [3]

A incentive C is attached to numerator term for all the nodes participating in the Relay Network formation and also an extra term B(t) is added in the denominator which denotes the residual battery power of the node. So a node which is behaving as a relay node will have more battery consumption and so this will lead to increase in the SP and hence will be more frequently schedules in comparison to the nodes which are getting their data traffic relayed or nodes which are not at all connected to the Relay Network.

We will take an example of two nodes one (A) behaving as an relay node and other (B) whose traffic is relayed. Since A is relay node its channel rate is 2X and of B is X. We have seen in [UCAN] that when B joins A's Relay network there is an increase in individual and overall throughput by 33%. The following mathematical derivation shows what parameters C1 and C2 can be attached to numerator term of SP of A and B to maximize individual throughput and

maintain the overall increase in throughput. In this derivation the battery power is not considered in the denominator but is used while simulating the mathematics of scheduling.

First calculation is done for the case when there is no relay network and fair scheduling by BS we have:

Throughput for A relay node	$T_{pA} = R_A * (1/2) = 2x * (1/2) = x$
Throughput for B	$T_{pB} = R_B * (1/2) = x * (1/2) = x/2$
Overall Throughput	3x/2

Now since Ca incentive is given to A and Cb incentive is given to B we have and Ca> = Cb
So scheduling parameters of A and B respectively are:

$$SP_A = Ca*R_A / T_{A(t)} \text{ and } SP_B = Cb*R_B / T_{B(t)}$$

While calculating scheduling parameter BS considers the actual rate of destination client so as not to give it unfair advantage but its data is relayed at the rate of Relay client. Also since effect of allotting incentive parameter has to be seen so throughput is assumed to be unity.

Let $C1 = Ca*R_A$ and $C2 = Cb*R_B$

Based on the above considerations the number of scheduling slots of A and B are respectively C1/C1+C2 and C2/C1+C2

Throughput for A relay node	$T_{pA} = R_A * (C1/C1+C2)$
Throughput for B	$T_{pB} = R_{A*} (C2/C1+C2)$

Overall Throughput 2x

So percentage change in throughput for A and B nodes in comparison to the case in which no relaying is done are as follows:

$$\%\Delta T_{pA} = [[R_A * (C1/C1+C2) – x] /x] * 100$$

$$\%\Delta T_{pB} = [[R_A * (C2/C1+C2) – x/2] /(x/2)] * 100$$

Since $R_A = 2x$ and $R_B = x$ so on solving above two equations we have

$$\%\Delta T_{pA} = [(C1- C2) /(C1+C2)] \text{ and}$$

$$\%\Delta T_{pB} = [(3C2 – C1)/(C1+C2)]$$

Further putting C1 = Ca * 2x and C2 = Cb* x we have

$$\%\Delta T_{pA} = [(2Ca- Cb) /(2Ca+Cb)] \text{ and}$$

$$\%\Delta T_{pB} = [(3Cb – 2Ca)/(2Ca+Cb)]$$

$$\%\Delta\text{Overall Throughput} =[2x – (3x/2)] /(3x/2) *100 = 33 \%$$

For A and B to give them equal increase in throughput from their previous values we equate their percentage increases and so we have

$$(2Ca- Cb) /(2Ca+Cb) = (3Cb – 2Ca)/(2Ca+Cb)$$

⇨ $(2Ca- Cb) = (3Cb – 2Ca) \Rightarrow 4Ca = 4Cb \Rightarrow \mathbf{Ca = Cb}$

For an equal increase in throughput values of A and B.

Also we can assign more incentive to A in comparison to B as A is the relay node, let's say we want to assign Ca and Cb so that there is 3:2 ratio in percentage increase in throughput values for nodes A and B respectively. This implies we have following equations:

$$2Ca - Cb = 3 \text{ and } 3Cb - 2Ca = 2$$

Solving these we have **Cb = 2.5 and Ca = 2.75.**

Similarly we can assign different values to Ca and Cb to increase relative advantage to relay node A in comparison to B whose traffic is relayed. This approach leads to motivation for selfish node to participate in Relay network formation and since their utility function is to increase the throughput so as to get maximum number of scheduling slots so they will participate and contribute to increase in individual throughput and overall throughput of the network.

Similarly we can solve the above two equations so as to give comparatively more advantage in terms of throughput to a node which is cooperative as well as relaying. Base Station will schedule a node which is having stronger credit over a longer duration of time and it has been relaying traffic for other nodes over consecutive time slots. This way in a non-cooperative environment the nodes which are misbehaving and not willing to participate in relay network formation for other nodes will be serviced basic minimum rate by the Base Station. In this way those nodes will be motivated to act cooperatively and participate in the relay network formation and since they are selfish so as to increase their throughput, they have to participate leading to increase in individual throughput and overall throughput of the wireless network.

SIMULATION AND RESULTS

In this section the simulation environment created for evaluating the performance of the modified PF scheduling algorithm and results obtained has been discussed. The simulation has been carried

out by programming a mobile environment using C programming language in MSVC++ Integrated development environment.

The simulation environment for scheduling algorithm consists of a Base Station and mobile nodes, UEs (User equipments like PDAs) roaming around in the region. The channel conditions and distance of the nodes from the BS goes on varying. The various parameters for a node are throughput, link rate from BS, distance from BS, scheduling parameter based on which BS schedules them in a particular time slot, cooperative, incentive and relaying flags and residual battery power which goes on draining with the number of slots in which the node is scheduled.

The cooperative flag is randomly set based on random number generated by ran3 () function used for simulating the mobile environment. The scheduling flag is set if the node is scheduled by the Base Station and incentive value goes on accumulating if the node has been working as a relay node and is cooperative also. The throughput for a node in a timeslot is the link rate multiplied by the slot length in seconds. Average throughput of a node is data transferred for it by the Base Station for it

For evaluating the performance benefit of a particular node, firstly it is assumed to be non-cooperative and all other nodes apart from that can behave in whatever possible way as decided by random function and the average throughput of that node is calculated in every consecutive timeslot. Since it is non-cooperative so it will not act as relaying node either and so it will not accumulate any incentive so probability of it being scheduled in any time slot decreases with time. This behavior is depicted in the figure [1] as shown below.

In the second case the same node is assumed to be cooperative and relaying always (It parameters cooperative and relaying flag are set accordingly) and all other nodes can behave either way as cooperative or non-cooperative. In this case also average throughput of the particular node is calculated over subsequent time intervals and it is found to be increasing and higher than all other

nodes. This is because for this cooperative and relaying node the numerator term goes on increasing till it reaches a maximum limit and is so scheduled more and encourages other nodes to join the relay network and cooperate so as to get their throughput also. This behavior is depicted in the figure [2] as shown below.

In these simulations above the battery power of the relaying node has also been added to the denominator term so that it's scheduling parameter increases and as a consequence it is scheduled more by the Base Station. Since it is working as a relaying node so it is scheduled more for forwarding data of other nodes as well as reception of its own data hence, its battery power drains more and keeping this in the denominator acts an incentive for the relaying node. The scheduling parameter [**C*R (t) / (T (t) +B (t)]** is calculated for every node in every time slot and then Base Station looks for the maximum value and selects the node having the highest value of this parameter.

Bibliography

[1] Schurr, N., Patil, P., Pighin, F., and Tambe, M. 2006. Using multiagent teams to improve the training of incident commanders.

[2] Scerri, P., Pynadath, D., Johnson, L., Rosenbloom, P., Si, M., Schurr, N., and Tambe, M. 2003. A prototype infrastructure for distributed robot-agent-person teams.

[3] Landgren, J. and Nulden, U. 2007. A study of emergency response work: patterns of mobile phone interaction.

[4] Fan, X., Sun, B., Sun, S., McNeese, M., and Yen, J. 2006. RPD-enabled agents teaming with humans for multi-context decision making.

[5] Haiying Tu; Allanach, J.; Singh, S.; Pattipati, K.R.; Willett, P.; , "Information integration via hierarchical and hybrid bayesian networks," Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on , vol.36, no.1, pp. 19- 33, Jan. 2006.

[6] Sycara, Katia P.; , "Agent Based Aiding of Human Teams," Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on , vol.1, no., pp.4-5, 15-18 Sept. 2009

[7] Wei Chen; Durfee, E.; Dumas, M.; , "Human agent collaboration in a simulated combat medical scenario," Collaborative Technologies and Systems, 2009. CTS '09. International Symposium on , vol., no., pp.367-375, 18-22 May 2009

[8] Freedy, A.; Sert, O.; Freedy, E.; McDonough, J.; Weltman, G.; Tambe, M.; Gupta, T.; Grayson, W.; Cabrera, P.; , "Multiagent Adjustable Autonomy Framework (MAAF) for multi-robot, multi-human teams," Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on , vol., no., pp.498-505, 19-23 May 2008

[9] Berna-Koes, M.; Nourbakhsh, I.; Sycara, K.; , "Communication efficiency in multi-agent systems," Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on , vol.3, no., pp. 2129- 2134 Vol.3, 26 April-1 May 2004

[10] Freedy, A.; Sert, O.; Freedy, E.; McDonough, J.; Weltman, G.; Tambe, M.; Gupta, T.; Grayson, W.; Cabrera, P.; , "Multiagent Adjustable Autonomy Framework (MAAF) for multi-robot, multi-human teams," Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on , vol., no., pp.498-505, 19-23 May 2008

[11] Rabiner, L.R.; , "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE , vol.77, no.2, pp.257-286, Feb 1989 doi: 10.1109/5.18626

[12] Komatsu, T.; Utsunomiya, A.; Ueda, K.; Oka, N.; , "Can you communicate with me? - An experimental design how humans regard artificial agents as communication partners," Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on , vol., no., pp. 193- 198, 31 Oct.-2 Nov. 2003

[13] Adams, J.A.; Paul, R.; , "Human management of a hierarchical control system for multiple mobile agents," Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology'., 1994 IEEE International Conference on , vol.3, no., pp.2780-2785 vol. 3, 2-5 Oct 1994

[14] Zhuomin Sun; , "Multi-Agent Based Modeling: Methods and Techniques for Investigating Human Behaviors," Mechatronics and Automation, 2007. ICMA 2007. International Conference on,vol.,no.,pp.779-783,5-8Aug2007

[15] Persson, M.; Wide, P.; , "Using a Sensor Source Intelligence Cell to Connect and Distribute Visual Information from a Commercial Game Engine in a Disaster Management Exercise," Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE ,vol.,no.,pp.1-5,1-3May2007

[16] Xue Wang; Sheng Wang; Daowei Bi; , "Distributed Visual-Target-Surveillance System in Wireless Sensor Networks," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactionson,vol.39,no.5,pp.1134-1146,Oct.2009

[17] Kaupp, T.; Brooks, A.; Upcroft, B.; Makarenko, A.; , "Building a Software Architecture for a Human-Robot Team Using the Orca Framework," Robotics and Automation, 2007 IEEE International Conference on , vol., no., pp.3736-3741, 10-14 April 2007

[18] Kaupp, T.; Makarenko, A.; Ramos, F.; Upcroft, B.; Williams, S.; Durrant-Whyte, H.; , "Adaptive human sensor model in sensor networks," Information Fusion, 2005 8th International Conferenceonvol.1,no.,pp.8pp.,25-28July2005

[19] Zhiyong Wang; Weisheng Xu; Jijun Yang; Jiazhen Peng; , "A Game Theoretic Approach for Resource Allocation Based on Ant Colony Optimization in Emergency Management," Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on,vol.,no.,pp.1-4,19-20Dec.2009

[20] Zhiyong Wang; Weisheng Xu; Jijun Yang; Jiazhen Peng; , "A Game Theoretic Approach for Resource Allocation Based on Ant Colony Optimization in Emergency Management," Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on,vol.,no.,pp.1-4,19-20Dec.2009

[21] Wooldridge, M. (2002). An introduction to multi-agent systems. Chichester: Wiley.

[22] D. L. Hall, "Application of the JDL Data Fusion Model to Condition Based Maintenance Systems," Proceedings of the 1994 Data Fusion Systems Conference, Johns Hopkins University, Laurel, MD, 15 pages, 24-27 October 1994.

[23] Mitchell, T. M. (1997). *Machine learning*. Maidenhead, England: McGraw-Hill

[24] Mackay, D. J. C. (2003). Information theory, inference, and learning algorithms. Cambridge, England: Cambridge University Press.

[25]Toni, F., & Bentahar, J. (2008). Computational logic-based agents. Autonomous Agents and Multi-Agent Systems, 16 (3), 211–213.

[26]Toussaint, M., Harmeling, S., & Storkey, A. (2006, Dec). Probabilistic inference for solving (PO)MDPs (Tech. Rep.). Edinburgh: University of Edinburgh.

[27]Excelente-Toledo, C. B., & Jennings, N. R. (2005, August). Using reinforcement learning to coordinate better. Computational Intelligence, 21 (3), 217–245.

[28]Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. Cambridge, Massachusetts, USA: MIT Press.

[29]Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems, 11 (3), 387–434.

[30]Judea Pearl, Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference, Chapter 1, Morgan Kaufmann

[31]Ross, S., Chaib-draa, B., & Pineau, J. (2008). Bayes-adaptive POMDPs. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), Advances in Bibliography 204 neural information processing systems 20 (NIPS 2007) (pp. 1225–1232). Cambridge, Massachusetts, USA: MIT Press.

[32] Ross, S., Pineau, J., Paquet, S., & Chaib-draa, B. (2008). Online planning algorithms for POMDPs. Artificial Intelligence Research, 32, 663-704.

[33]Dearden, R., Friedman, N., & Andre, D. (1999). Model-based Bayesian exploration. In Proceedings of the 15th annual conference on uncertainty in artificial intelligence (p. 150-15). San Francisco, CA: Morgan Kaufmann

[34]Chalkiadakis, G., & Boutilier, C. (2003). Coordination in multiagent reinforcement learning: a Bayesian approach. In Proceedings of the second international joint conference on autonomous agents and multiagent systems (pp. 709– 716). New York, USA: Association for Computing Machinery.

[35]Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. In Proceedings of the third international joint conference on autonomous agents and multiagent systems (pp. 136–143). Washington, DC, USA: IEEE Computer Society Press.

[36]Marecki, J., Gupta, T., Varakantham, P., & Tambe, M. (2008). Not all agents are equal: scaling up distributed POMDPs for agent networks. In Proceedings of the seventh international joint conference on autonomous agents and multiagent systems (AAMAS 08). New York, USA: Association for Computing Machinery.

[37] H. Luo and et. al, “UCAN: a unified cellular and ad-hoc network architecture”

[38] Integrated Cellular and Ad Hoc Relaying Systems: iCAR by Hongyi Wu, Chunming Qiao, Swades De, and Ozan Tonguz

[39] Channelization for Dynamic Multi-Frequency,Multi-Hop Wireless Cellular Networks ,JaeSheung Shin, Raju Kumar, Parthu Kishen, Thomas F. La Porta

[40]Dynamic Multi-Frequency, Multi-HopWireless Cellular Networks JaeSheung Shin, Parthu Kishen, Thomas F. La Porta, Fellow, IEEE

[41]On the relaying capability ofnext generation GSM cellular networks. G.M. Aggelou and R. Tafazolli. IEEE Wireless Communications, 8(1):40–47, 2001.

[42]CDMA/HDR: a bandwidth-efficient high-speed wireless data service for nomadic users, P. Bender et al. IEEE Communications Magazine, pp.70-78, vol. 38, July 2000

[43]Cooperative Multiplexing and Scheduling in Wireless Relay Networks, Yi Shi, Wei Zhang, and Khaled Ben Letaief, Fellow, IEEE

[44]System-Level Performance of Cellular Multihop Relaying with Multiuser Scheduling Mohamad Charafeddine,Ozguir Oymant, and Sumeet Sandhut

[45] “Multihop "Basketball" Packet Scheduling in Uplink DSCDMA Systems Y. Hwang, R. Jäntti, and S.L.Kim

[46]Y. D. Lin and Y. C. Hsu, “Multihop cellular: A new architecture for wireless communication,” in IEEE INFOCOM'2000, 2000, pp. 12731282