MULTI-AGENT SYSTEMS FOR DATA-RICH, INFORMATION-POOR ENVIRONMENTS

A Thesis in
Information Sciences and Technology

by

Viswanath Avasarala

© 2006 Viswanath Avasarala

Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

August 2006
The thesis of Viswanath Avasarala was reviewed and approved* by the following:

Tracy Mullen
Assistant Professor, Information Science and Technology
Thesis Advisor
Chair of Committee

David Hall
Associate Dean of Research, Information Science and Technology

Steven Haynes
Assistant Professor, Information Science and Technology

Murali Haran
Assistant Professor, Department of Statistics

Joe Lambert
Chair, Graduate Programs Advisory Committee

*Signatures are on file in the Graduate School
ABSTRACT

The recent development of sensors integrated with memory, power supply and wireless networking capabilities marks a new era in sensor technology, with wide ranging implications for both military and civilian domains. The capability for ubiquitous and distributed sensing has lead to the possibility of data-rich and information-poor environments, where the ability to collect data has overtaken the ability to understand its relevance and importance to the overall system goals. If the benefits of the sensor technology developments are to reach end users, we need to address two key questions. First, what data should be gathered given resource constraints like limited sensor battery power? Second, what information should be shared with humans, and between humans, given their cognitive constraints? This thesis focuses on development of agent-based information management algorithms and architectures that can deal with the massive amounts of data generated, without overloading the human operators. Intelligent agent technology with its emphasis on autonomy provides a valuable paradigm for this problem.

This thesis mainly focuses on designing and building a market-based resource allocation architecture for sensor management in distributed sensor networks. A second domain, supply chain management, examines the question of what information should be shared, and involved development of a collaborative sense-making application.

A market-based agent design is proposed for the distributed sensor management problem, where the different system units are regarded as various market entities. This approach has the ability to create a comprehensive sensor management paradigm that can
optimally distribute non-commensurate sensor network resources (e.g., sensor attention, battery power, and transmission capacity) among the distributed consumers, operating in a co-operative or semi-cooperative environment.

A team-based agent design is proposed for collaborative sense-making in a multi-echelon supply chain. The various supply-chain entities, including the data generating entities (like RF sensors) are treated as team members with specific roles in a multi-agent team, based on the multi-agent team framework, Collaborative Agents for Team work (CAST). This approach holds the promise of addressing the information needs of the individual agents without the causing the problems of information overload, using the CAST based pro-active information and knowledge delivery policies.
TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................ viii
LIST OF TABLES .......................................................................................................... xi
ACKNOWLEDGEMENTS ............................................................................................... xii

Chapter 1  Introduction ...............................................................................................1
   1.1 Problem Definition and Motivation ...............................................................1
      1.1.1 Sensor Management in Distributed Environments .........................4
      1.1.2 Supply Chain Management .............................................................7
   1.2 Problem Scope ...............................................................................................8
      1.2.1 Sensor Management .....................................................................8
      1.2.2 Supply Chain Management ..........................................................10

Chapter 2  Problem Background ...............................................................................13
   2.1 Sensor Management ...................................................................................13
      2.1.1 Market Algorithms ......................................................................18
         2.1.1.1 Formal model of a Resource Allocation Problem ..........19
         2.1.1.2 Economic Equilibrium and Optimization .....................20
         2.1.1.3 Finding the Equilibrium .....................................................21
         2.1.1.4 Auctions .............................................................................23
         2.1.1.5 Combinatorial Auctions ....................................................25
            2.1.1.5.1 Formulation of the Winner Determination Problem ..27
            2.1.1.5.2 Winner Determination Algorithms .........................27
   2.2 Supply Chain Management .......................................................................10

Chapter 3  Market Architecture for Sensor Management ....................................31
   3.1 Introduction .................................................................................................31
   3.2 CCA Protocol .............................................................................................38
   3.3 Pricing Mechanisms ...................................................................................48
   3.4 Analysis of CCA .......................................................................................50
   3.5 Simulation Environment ...........................................................................52
   3.6 Summary of Results ..................................................................................58
      3.6.1 Real-time Performance ...............................................................59
      3.6.2 Resource Utilization .................................................................60
      3.6.3 Task Deadlines ............................................................................64
   3.7 Effects of Strategic Behavior .................................................................65

Chapter 4  Real-Time Winner Determination in Combinatorial auctions ..........70
   4.1.1 Winner Determination for Resource Allocation using CAs .............72
4.1.2 SGA ......................................................................................................73
4.1.3 Representational Schema.................................................................74
4.1.4 SGA Operators .................................................................................75
4.1.5 Seeding the GA ...............................................................................78
4.1.6 Avoiding Explicit Bid Formulation .................................................79
4.2 Results ....................................................................................................80
4.3 Real-time Performance ........................................................................87

Chapter 5  Agent Learning for Task Prioritization in Sensor Networks.......89

5.1 Requirements of Agent Learning......................................................90
5.2 Implementation Details.................................................................93
5.2.1 Market Reports .................................................................93
5.2.2 Agent Learning .................................................................94
5.3 Results .............................................................................................97

Chapter 6  Approximate Techniques for Market-based Algorithms ........100

6.1 A-MASM Architecture.................................................................100
6.1.1 Adapting SGA for A-MASM ...............................................102
6.2 Utility-Estimation using Radial Basis Network ...........................104
6.2.1 Radial Basis Network Theory .............................................104
6.2.2 Performance ........................................................................106
6.3 Performance of Approximate Methods ........................................109
6.4 Scalability Analysis ......................................................................114

Chapter 7  Comparison of MASM to Information-Theoretic Sensor Manager 117

7.1 Information-Theoretic Sensor Manager ........................................117
7.2 Enforcing Resource Constraints in ITSM .....................................118
7.3 MASM-ITSM Comparison .........................................................119
7.4 Interpretation of MASM’s Superior Performance ......................120

Chapter 8  Conclusion .........................................................................126

8.1 Contributions ...............................................................................126
8.2 Future Work ...............................................................................127

Bibliography .........................................................................................130

Appendix A   A Team-Based Multi-Agent Architecture for SCM ............141

A.1 Problem Background .....................................................................141
A.1.1 Multi-Agent Systems for Supply Chain Management ............141
A.1.2 Team-based Agents .................................................................142
A.2 Team-based Agents for SCM .......................................................144
LIST OF FIGURES

Figure 1-1: JDL data fusion model .................................................................5
Figure 1-2: JDL Level 4 process refinement .......................................................6
Figure 2-1: A generic sensor management technique .........................................14
Figure 2-2: Exhaustive partition of 3 items ........................................................28
Figure 3-1: Market architecture for sensor management ..................................32
Figure 3-2: MASM architecture ....................................................................34
Figure 3-3: Flowchart of MASM ....................................................................39
Figure 3-4: Illustration of calculation of bid prices for resource bundles using QoS chart ............................................................45
Figure 3-5: Average time taken on 2.8 GHz Pentium IV processor for winner determination by CPLEX (averaged over 10 runs) ..........................60
Figure 3-6: Price variation of the first three sensors with schedule number for a sample run with tatonement $\tau = 0.005$ .........................................................62
Figure 3-7: Energy utilization for the first three sensors for a sample run with tatonement $\tau = 0.005$ .................................................................................62
Figure 3-8: Energy utilization for the first three sensors for a sample run with tatonement $\tau = 0$ .........................................................................................63
Figure 3-9: Time taken for communication for a sample run vs schedule number for a sample run with tatonement $\tau = 0.005$ .........................................................63
Figure 3-10: Time taken for communication for a sample run vs schedule number for a sample run with tatonement $\tau = 0$ .........................................................64
Figure 4-1: Regression line for average optimality versus problem size for uniform distribution .................................................................83
Figure 4-2: Regression line for average optimality versus problem size for bounded distribution .................................................................83
Figure 4-3: Estimated percentage optimality (with their 95% confidence intervals) versus problem size for uniform distribution, using a cut-off time of 200 CPU-Sec on 2.8 GHz Pentium IV processor .................................................................85

Figure 4-4: Estimated percentage optimality (with their 95% confidence intervals) versus problem size for bounded distribution, using a cut-off time of 200 CPU-Sec on 2.8 GHz Pentium IV processor .................................................................85

Figure 4-5: Correlation of revenue obtained by SGA and Casanova for uniform distribution with problem size of 2000 bids. .................................................................86

Figure 4-6: Correlation of revenue obtained by SGA and Casanova for uniform distribution with problem size of 2000 bids. .................................................................86

Figure 4-7: Real-time performance of SGA and Casanova on a 2.8GHz Pentium-IV processor (averaged over 20 runs). .............................................................................88

Figure 4-8: Real-time performance of SGA (seeded with Casanova) and Casanova on a 2.8GHz Pentium-IV processor (averaged over 20 runs). ...........................................88

Figure 5-1: Approximate price-QoS mapping generated by consumer for search task, during a simulation experiment .................................................................94

Figure 5-2: Number of targets destroyed versus search budget (averaged over 10 simulation experiments). .........................................................................................97

Figure 5-3: Convergence of search budget to optimal value, based on Widrow-Hoff learning ...........................................................................................................98

Figure 6-1: Architecture of sensor manager in A-MASM .........................................102

Figure 6-2: Schematic representation of a radial basis network ..............................105

Figure 6-3: Performance of RBF network for search task .......................................108

Figure 6-4: Performance of RBF network for track task ........................................108

Figure 6-5: Time Required for formulating resource-bids from the consumer task-bids .........................................................................................................................111

Figure 6-6: Time Required by CPLEX to solve the IP problem (averaged over 10 runs) .........................................................................................................................112

Figure 6-7: Comparison of time requirements for E-MASM and A-MASM ..........113

Figure 6-8: Optimality of SGA for different problem sizes (averaged over 10 runs) ........................................................................................................................114
Figure 6-9: Scalability of E-MAAS and A-MAAS ................................................................. 115

Figure 7-1: Comparison of MASM with ITSM (averaged over 10 runs) .................. 120

Figure 7-2: Change in uncertainty of target tracks, while using an information-theoretic approach........................................................................................................... 122

Figure 7-3: Change in uncertainty of target tracks, while using an utility-based approach......................................................................................................................... 123

Figure 7-4: A comparison of the number of sensors used for measurements, based on whether target tracks are currently in progress or not ................................... 125

Figure 8-1: Market-Based platform scenarios ............................................................ 128

Figure A-1: Rule for assessing the customer’s demand............................................. 151

Figure A-2: Architecture For PSUTAC ..................................................................... 154

Figure A-3: Proactive communication in a supply chain ...................................... 157

Figure A-4: An example of MALLET ........................................................................ 159
LIST OF TABLES

Table 3-1: Sensor Characteristics used in Simulation ..................................................55
Table 3-2: Parameter Values used in Simulation ..........................................................58
Table 3-3: Market Performance with Strategic Agent Behavior .................................68
Table 4-1: Example of SGA Chromosome ..................................................................75
Table 4-2: Regenerator Algorithm for SGA .................................................................77
Table 4-3: SGA Parameters .........................................................................................82
Table 6-1: Parameters used for SGA in A-MASM ......................................................113
ACKNOWLEDGEMENTS

First, I wish to convey my greatest gratitude to my parents, A.V.S.N. Murthy and A. Nagamani and my brother A. Srinivas for supporting me in this arduous journey. I am grateful to my parents for giving me the best possible education from my childhood and having unflinching confidence in me throughout.

I would like to express my sincere gratitude to Dr. Tracy Mullen for her support and encouragement during the last four years. She introduced me to the exciting field of market-oriented programming and her wide knowledge and thorough supervision has shaped this research work. Also, she has been a very good friend to me, offering me valuable personal advice and help whenever I needed it.

I am greatly indebted to Dr. David Hall for providing me financial support throughout my graduate studies. Many a times, when I felt this research effort was going nowhere, his insightful suggestions helped me see light again. I thank Dr. Haynes for painstakingly reading my thesis and offering me some valuable comments. I also express my gratitude to Dr. Haran for helping me break down the analysis of CCA protocol. I thank Dr. Amulya Garga for his help during the formative stages of the project and his guidance in the choice of course work.

I am indebted to Himanshu Polavarapu for his help in formatting my thesis and in conducting the elaborate statistical analysis of my experimental results. I am thankful to Padmapriya Ayyagari for suggesting the use of incremental algorithms for tackling the computational complexity of combinatorial auctions. My thanks to Hari Prasad,
Rambabu Pothina, Randheer Shetty, Sandeep Mudunuri and other friends at Penn State for their wonderful companionship.

The journey begins now...
Chapter 1

Introduction

1.1 Problem Definition and Motivation

Recent years have seen great developments in the field of sensor technology and its applications, both in military and civilian domains [1, 2]. Sensors, integrated with memory, power supply and wireless networking capability made distributed and ubiquitous sensing a reality [3]. However, the huge data collection capacity afforded by such improved sensor systems places great strains on human, computational and storage resources. Lack of sophisticated high level algorithms to appropriately harness the benefits of these sensor developments has created data-rich, information-poor (DRIP) environments. High-level DRIP activities, such as sense-making, decision-making and resource allocation, require gathering and coordinating information spread across sensors, information processes, software agents, and humans. Requiring these interacting entities to share all their local information is infeasible since this could lead to information overload or a violation of privacy issues. Thus for the benefits of recent sensor technology developments to reach end users, without overloading them, automated and distributed information management algorithms need to be developed that can provide decision-making entities with access to significant time-critical information, while filtering out irrelevant data.
We believe that multi-agent technology with its emphasis on autonomy, modularity, and distributed design provides a natural paradigm for this problem domain. Agent coordination and cooperation frameworks include market-oriented programming, negotiation-based interactions, and team-based interactions (see [4] for an exhaustive survey). To demonstrate and test the effectiveness of multi-agent-based design for automated sense-making of data, we have chosen two different domains which have great influence from sensor technology, sensor management (SM) and supply chain management (SCM). The bulk of this thesis focuses on sensor management in distributed networks. For this study, the sensor manager is mainly concerned with directing the data-collecting entities, the sensors, to satisfy the information requirements of the higher-level users in the best possible way. That is, we approach the information-processing in a top-down fashion. First, the users submit requests for information and the SM is required to task the sensors to best satisfy the information requests. However, in the second domain, supply chain management (which is described in more detail in the Appendix), our information-processing approach to a distributed RFID supply chain management problem is bottom-up. Data-generating entities like RFID sensors generate periodic readings of various system variables. The information-processing algorithm governs access to the data generated so that individual agents are not overwhelmed and at the same time, have timely information available to take appropriate actions.

Both these domains and approaches have the following common attributes, which makes them interesting cases for studying multi-agent based design for distributed information management systems:
1. *Relevance and importance of data to overall system goals:* In both domains, the ability to collect data has overtaken the ability to understand its relevance and importance to the overall system goals. The key to the successful utilization of the new data collection technologies is the ability to generate useful information and knowledge, from the collected data.

2. *Information consumers with independent goals:* Both the supply chain entities and the consumers of a sensor network can have independent goals and objectives and function in a semi-cooperative environment.

3. *Real-time constraints:* These environments are also characterized by strict real-time considerations where time pressure is a crucial consideration in the decision making process.

4. *Relevant information is distributed amongst various independent entities:* Straight-forward decision making in these domains can be cast as an optimization problem. However, the variables of the optimization routine are spread as private information among the various distributed domain entities.

Thus, collaborative sense-making, or the ability of distributed entities to make collective sense of the environment in which they operate, is not a straightforward task. Information management architectures and algorithms based on a multi-agent system approach dovetails nicely with many of the requirements for sense-making in distributed systems. This chapter briefly introduces the challenges of information processing in SM and SCM and outlines the contributions this thesis offers to tackle them.
1.1.1 Sensor Management in Distributed Environments.

Sensor management can be defined as “a process which seeks to manage or coordinate the use of sensing resources in a manner that improves the process of data fusion and ultimately that of perception, synergistically” [5]. Multi-sensor systems rely on data fusion techniques to combine data from multiple sensors and related information to achieve more specific inferences than achievable by using a single, independent sensor. Sensor management system’s responsibilities include automation of sensor allocation and moding, pointing and emission control, prioritization and scheduling of service requests, coordinating fusion requests with data collected from different sensor and sensor modules, supporting reconfiguration and degradation due to loss of sensors or sensor modes and communication of desired actions to the individual sensors [6].

A functional model of data fusion, Joint Directors of Laboratories (JDL) data fusion processing model [6], has been proposed that illustrates the primary functions, relevant information and databases, and interconnectivity required to perform data fusion. The model comprises four levels, which form a hierarchy of processing (see Figure 1-1). Level 1 processing, known as object refinement, fuses positional and identity data from multiple sensors to determine entity identities and to form tracks.
Level 2 processing, known as *situation refinement*, aims to infer the meanings or patterns in the order of battlefield, by fusing the spatial and temporal relationships between entities. Level 3 processing performs *threat refinement* to assess enemy threat, including estimation of their lethality, composition, evaluation of indication and warnings of impending events, targeting and weapons assessment calculations. Level 4 performs *process refinement*, which is an ongoing monitoring and assessment of the fusion process to refine the process itself and to regulate the acquisition of data to achieve optimal results (see Figure 1-2). Level 4 processing should consider mission constraints and requirements so that SM actions do not impede mission objectives. Level 4 processing
includes sensor management functions which entail determination of sensor availability, sensor scheduling, task prioritization, sensor health monitoring, handling communication channels, etc.

Figure 1-2: JDL Level 4 process refinement

Sensor management algorithms map into Level 4 data fusion whose concern is the optimization of sensor or information sources utilization and algorithms to achieve the most useful set of information. Most research efforts in the area of data fusion have concentrated on the lower levels of data fusion hierarchy such as development of algorithms of tracking, situation assessment and threat refinement. Process refinement and optimization in heterogeneous multi sensors has been an under researched area and
therefore lacks coherent architectures and algorithms [7]. However, if the benefits of the recent developments in sensor technology are to reach the end users, development of data fusion level four algorithms is critical. The recent developments in sensor technology have not had the complement of corresponding developments in sensor management algorithms, leading to data rich and information poor environments [9]. Modern sensors are integrated with computational power, energy, communication and memory resources. Simultaneous consideration of these non-commensurate measurements is a requirement for efficient use of a distributed heterogeneous sensor network, thus making sensor management a more ardent task than a simple, single-value optimization problem. This research project developed a comprehensive sensor management algorithm that accounts for the heterogeneity of the sensors, threat levels in the environment, and provides for distributed and decentralized control.

### 1.1.2 Supply Chain Management

Ubiquitous sensing capabilities have great implications for supply chain management. Real-time information from sensors can lead to more efficient manufacturing, distribution and logistics. Many companies including Wal-mart have invested heavily in Radio Frequency Identification (RFID) technology to revolutionize their supply chains [10]. The basic idea behind RFID technology is to create smart shelves that monitor inventory levels. Low-inventory generates an automatic signal to the store manager. This information propagates throughout the supply chain entities, including the distribution center and manufacturers. However, this supply chain
mechanism involves significant data generation that can cause a problem of information overload to the supply chain manager. Thus, adequate, automated response systems throughout the supply chain are critical to reduce the data processing requirements of managers. This study’s approach to this problem is to model the supply chain as a multi-agent system, where each supply chain unit, including the data generating sensors is treated as agents. Our multi-agent design aims to create an information sharing environment where only the essential information required for coordination is communicated, so that individual agents are not overwhelmed with data. This requirement entails that agents anticipate each other’s information needs. For this purpose, we use the concept of team-based agents where agents have a shared mental model. Team-based agents are aware of each other’s roles in the team and can thus reason about other’s information requirements. This design prevents problems in supply chain that might occur due to a lack of timely intervention while simultaneously avoiding the problem of information overload at the same time.

1.2 Problem Scope

1.2.1 Sensor Management

The sensor manager has to account for a number of factors in the optimization of sensor utilization. The various parameters include threat levels in the environment, bandwidth and power requirements of the sensors, and expected performance level of a sensor for a particular task. The sensor manager might also have to deal with requests for
sensor resources from multiple information-seeking consumers. Furthermore, some of the sensors and consumers might be humans and so the sensor manager has to account for any “human in the loop” problem. The difficulty in converting all the concerned factors into commensurate measures that can be used in an optimization code is one of the factors that makes this problem a difficult one. Another factor is real-time constraints of the problem domain. Additionally, in current sensor-rich environments, where the consumers and sensor resources have spatio-temporal distribution, centralized control and optimization of the problem might not be feasible. In this situation, the sensor management algorithm requires distributed and decentralized control and must provide simultaneous consideration of diverse, incommensurate measures like bandwidth, sensor battery power, network processing power, etc. Previous sensor management techniques have relied on converting the diverse incommensurate measures into ad-hoc heuristic measures for use in some optimization algorithm. These solutions to sensor management lack generalizability and suffer from being “point solutions” that are very specific to the domain in which they have been developed [11].

An ideal solution to the sensor management problem includes the development of a system architecture and control algorithms that:

a) is generalizable and can be adapted to wide range of sensor network domains
b) provides for distributed, decentralized control
c) provides “human in the loop” capability
d) results in optimal (or sufficiently optimal) allocation of sensor resources.
This research project developed a comprehensive sensor management algorithm that possesses the above attributes, and thus can successfully account for the heterogeneity of the sensors, threat levels in the environment and provide for distributed and decentralized control.

1.2.2 Supply Chain Management

Collaborative sense-making, or the ability of business partners to jointly make sense of the environment, is becoming an increasingly important capability that senior executives should pursue in order to effectively manage their supply chains. The integration of the new sensing technologies with supply chains has created the possibility of generation massive amounts of potentially useful information [11]. Traditional decision support systems do not have the ability to deal with such magnitudes of data. Moreover, overwhelming executives with too much information is dangerous. A recent study by Sutcliffe and Weber [12] concludes that for top-level executives, collecting information is less important than the interpretation of information. Another danger for executives is over reliance on intuition, especially in new or dynamic situations, which can be biased and limited by human cognitive capabilities.

Although prior work on multi-agent architectures for supply chains exists [13-16], they have not adequately leveraged the findings of the supply chain research community in their design. We believe that a comprehensive multi-agent design should have a firm grounding in the relevant research findings of management science. For this purpose, we have used the work of Hult et al. [17] to guide our design process. Based on an extensive
survey of supply chain and related literature, Hult, et al. [17] formulated a model to explain supply chain efficiency as revealed by its cycle-time in terms of its achieved memory, knowledge acquisition activities, information distribution activities and shared meaning. Achieved memory is defined as “the amount of knowledge, experience or familiarity with the supply chain process.” Shared meaning is “the extent to which participants develop common understanding about data and events.”

The following guidelines have been identified from the work of Hult, et al. as being relevant to the proposed agent architecture design:

a.) Knowledge acquisition: Each member in a supply chain should have a systematic knowledge acquisition approach, guided by its achieved memory. Extending Grant [18]’s view of firm to supply chains, supply chains are regarded a knowledge integrating entities, whose primary role is application of the knowledge acquired. Knowledge acquisition or memory creation has been found to a prerequisite for development of shared meaning across supply chain which in turn decreases cycle time.

b.) Shared Meaning: Creation of shared meaning enables members in a supply chain to reason about and interpret other’s actions and intentions. Shared meaning is a critical mechanism for communication and co-ordination within a supply chain [19] since the participating units lack a common culture [20].

c.) Information distribution:

Although supply chain effectiveness depends on the exchange of timely and accurate information across customers, material and service suppliers, and internal functional areas [21], the tremendous rate at which modern information systems generate data can overwhelm supply chain units [20]. In a study of organizational learning, Huber
[22] stated that the excessive information that exceeds a unit’s information processing capacity can adversely effect information interpretation within the unit. In information rich environments, “informational autonomy” between the various units might improve overall efficiency by addressing the problem of information overload. Thus, information flow can be postulated to have a curvilinear relation with outcomes, with an inflection point after which dealing with more information becomes overwhelming [17]. Therefore, “information should be (is) distributed to only those who need it”. An intelligent information distribution mechanism allows the members to gather enough information about their environment while not overloading their cognitive capacities with excessive or redundant information.

This theoretical background offers the following directions for the supply chain multi-agent design research:

a.) Agents should have a systematic knowledge acquisition procedure, guided by its previous experience and knowledge

b.) Agents should have a mechanism for creation of shared meanings, since shared meaning provides the basis for supply chain co-ordination

c.) Communication mechanisms should be provided so that timely information is provided to the agents that require it while also taking care that redundant or excessive communication is avoided.

Based on the above principles, this research has developed a team-based multi-agent framework for SCM that can potentially provide a comprehensive information processing approach for multi-tier supply chains.
Chapter 2

Problem Background

2.1 Sensor Management

The terms, sensor management and sensor scheduling, though used interchangeably in research literature, are not equivalent. Sensor management refers to “...the process which seeks to manage or coordinate the use of sensing resources in a manner that improves the process of data fusion and ultimately that of perception, synergistically [5],” whereas, sensor scheduling refers to the actual allocation of measurement tasks to specific sensors. Thus, sensor scheduling is one among the many responsibilities of a sensor manager.

Denton et al. [23] proposed a generic architecture and a hierarchical control methodology for the sensor manager problem. The proposed sensor management system consists of a mission manager, sensor manager and the sensor suite, forming a hierarchical control system (Figure 2-1). The mission manager is responsible for high level mission decisions and provides the primary direction to the sensor manager by sending information requests. The sensor manager consists of information instanatiator, sensor scheduler and personality modules. The information instanstiator is responsible for converting the information level requests of the mission manger to measurement level requests understandable to the sensor suite. For example, a request for a target track from the mission manager is converted to an observation request by the information
instantiator that can be used sensor scheduling. The sensor scheduler is responsible for allocating the various measurement tasks to the individual sensors.

Figure 2-1: A generic sensor management technique (Denton et al. [23])
Scheduling research in operations research and computer science research domains serve as a good starting point for a solution to the sensor scheduling problem. Heuristic solutions like simulated annealing [24], tabu search [25], stochastic probe approaches [26], genetic algorithms [27, 28] and neural networks [29, 30] form the core of the suggested solutions. However, none of these solutions have been found to be directly applicable to the problem of sensor scheduling and for this purpose, Hintz K.J and Zhang Z. have developed a specialized scheduling algorithm, OGUPSA, for sensor scheduling [8]. OGUPSA uses a FIFO queue structure for task allocation to a group of sensors based on the three main scheduling policies of most-urgent-first to pick a task, earliest-completed-first to select a sensor and least-versatile-first to resolve ties.

As explained in Chapter 1, Sensor management is not concerned with the optimization of task scheduling alone. Past research has proposed several algorithms for different variants of this problem which include heuristics [31], expert systems [32], utility theory [33], automated control theory [34], cognition [35], decision theoretic approaches [36], probability theory [37], stochastic dynamic programming [38], linear programming [39], neural networks [40], genetic algorithms [41] and information theory [42-44].

Based on the Denton et al. [23] architecture, Hintz and McIntyre proposed a sensor management strategy that uses posnets/lattices in the mission manager to derive the priorities of various tasks. They use an information theoretic approach for the information manager to conduct the cost benefit analysis of the various information
requests from the mission manager and the OGUPSA algorithm for the sensor scheduler to schedule the measurement tasks [7].

In spite of extensive investigation in this area, the research in this domain is incomplete because of the following reasons:

a) Traditional sensor management addressed only sensor schedule optimization. However, the sensor allocation optimization problem is not independent from the allocation of the other network resources like bandwidth for communication, energy required for communication and processing power.

b) Most of the solutions are “point solutions” that do not use generic sensor management architecture, and thus, their applicability beyond their original test beds is unclear.

c) Without decentralized or distributed control mechanisms, sensor management techniques might not scale well or show acceptable real-time performance.

d) The neglect of “human in the loop” implies that the sensor management architecture cannot cater to a system that involves humans acting as sensors or information consumers.

e) Strict real time considerations of the sensor management domain place very difficult constraints on the possible computational complexity of the sensor management algorithms. For example, entropy calculations, required for information theoretic approaches, are computationally intensive, rendering them generally unusable in domains with strict real-time constraints.

f) By focusing primarily on optimizing the overall quantity of information obtained using sensor resources, traditional sensor management approaches, such as
information-theoretic SM, have essentially neglected the value of information to overall mission goals.

Researchers have only recently begun addressing these issues—using, for example, dynamic goal lattices [45], decision making theory [46], and Bayesian networks [47] to find the priorities or weights for various objectives for SM optimization. However, a comprehensive paradigm that directly considers mission objectives during resource allocation is lacking. Market-based approaches provide a rigorous methodology for incorporating information’s value to mission goals during resource allocation. When distinct organizations use the same sensor manager, complete information about any individual user’s situation might not be available. In this case, a traditional sensor manager approach has difficulty deciding the priorities associated with different resources’ requests.

The fact that market-oriented programming can provide a comprehensive platform for sensor management has been recognized by researchers, as early as 2001 [48]. However, a few stumbling blocks prevented the development of market-based sensor manager. They are:

a) The computational complexity of eliciting valuations for all possible combinations of resources to different tasks/users.

b) The computational complexity of the winner determination problem.

c) Problems associated with mapping from high-level mission goals to priorities for actionable tasks.

d) Modeling reasonable pricing mechanisms for non-commensurate resources that exist in the market.
2.1.1 Market Algorithms

Market-based algorithms have had use for resource allocation in distributed allocations in a wide range of scenarios including bandwidth allocation [49], network information services [50], digital libraries [51], distributed operating systems [52] and electric load distribution [53], etc. Market-oriented programming refers to the design and implementation of distributed resource allocation problems based on some pricing system. Application of market mechanisms to scheduling have shown promising results [54-57]. This approach uses the fundamentals of economic theory for designing and implementing resource allocation problems. The basic idea behind these algorithms is that price-based systems facilitate efficient resource allocation in computational systems, just as they do in human societies. Resource-seeking entities are modeled as independent agents, with autonomy to decide about how to use their respective resources. These agents interact via a market that uses a pricing system to arrive at a common scale of value across the various resources. Individual agents use the common value scale for making trade-off decisions about acquiring or selling goods. Market-oriented programming essentially involves designing the mechanism in which agents interact to determine prices and exchange goods. This design is usually based on the principles of micro-economic theory. This section presents an overview of market algorithms for distributed resource allocation problems. First, established is the reformulation of a distributed resource allocation problem as a market problem. Following are the basic principles of general economic theory. Various auction protocols are then briefly
described with particular emphasis on the recent most developments in combinatorial auction algorithms.

2.1.1.1 Formal model of a Resource Allocation Problem

Walsh [69] presents a formal model of a scheduling problem with a single seller. This is a modified version that models a distributed resource allocation scenario with multiple sellers in terms of the following elements:

- $A$ is a set of $m$ agents $\{a_1, \ldots, a_m\}$
- $i_k$ is the initial endowment of the $k$-th agent and $G$ be the set of all goods with $n$ elements $= \bigcup_{k=1}^{m} i_k$.
- $p$ is the set of prices of the goods $\{p_1, \ldots, p_n\}$

Agents are assumed to have a quasilinear utility, which means that a common numeraire, money, can be used to measure agent valuations. This implies that the utility of the various agents can have direct comparison.

$U_k(g)$ is the utility of agent, $k$, for holding the set of goods, $g$.

At a given set of market prices, $p$, the agent faces a maximization problem:

$$\text{Max } [ U_k(g) - \sum_{i : g \cap i_k} p_i + \sum_{j : g \setminus i_k} p_j ]$$

$$g \subseteq G.$$

The operator “\(\setminus\)” denotes the set difference operator. The set difference for two sets $A$ and $B$ is:

$$A \setminus B = \{x/x \in A \& x \not\in B\}$$
The maximum surplus value agent, \( j \), can achieve at prices \( p \) is \( H_j(P) \).

\( F \) denotes the final allocation, with \( \{f_1 \ldots f_m\} \) being the final allocation of the goods.

The global value, \( V \), of the solution, \( f \), is the sum of the utilities of the individual agents:

\[
V(f) = \left[ U_k(g_f) - \sum_{i \in g_f} p_i + \sum_{j \in g_f} p_j \right] = H_j(P).
\]

**2.2**

A solution, \( S \), is globally optimum, if no other allocation of resources has a higher global value.

### 2.1.1.2 Economic Equilibrium and Optimization

A key concept of the general economic theory and market-oriented programming is the idea of economic equilibrium. For the resource allocation problem presented in the earlier section, a solution, \( f \), is in equilibrium at prices, \( p \), if and only if:

\[
\left[ U_k(g_f) - \sum_{i \in g_f} p_i + \sum_{j \in g_f} p_j \right] = H_j(P).
\]

i.e, the agent’s allotment maximizes its utility at the given prices.

Economies with only two commodities have a unique equilibrium if each agent’s demand function is continuous and non-increasing and has positive change in desired allocation for the first commodity if its price approaches zero and vice-versa. In the more general case, the issue is more complicated, and detailed results are in [58].

If market-oriented methods are used for resource allocation, then the important question of optimality of the obtained allocation arises. The following theorem from [59,
presents the relationship between optimality and equilibrium for the standard equilibrium market, presented above.

**Theorem 2.1**

If a competitive equilibrium exists for the market formulated in section 2.1.1.1, then the equilibrium is an optimal resource allocation point.

**Proof**

The proof of this theorem is based on the assumption that the consumer utilities are quasi-linear i.e., their values are linear in terms of a common numeraire, usually referred to as money. A detailed proof appears in [59] and [60].

**2.1.1.3 Finding the Equilibrium**

The search for equilibrium is usually an iterative process, with the participants submitting bids to the auctioneer, who in turn requests new bids based on the additional information gathered. Computing the equilibrium and allocation of resources only after finding the equilibrium is a “tatonnement process”, (e.g.[61]). Various algorithms inspired by the tatonement process for searching the equilibrium have been formulated. Based on the search parameter used to find the equilibrium, these approaches classify either as price-oriented algorithms or resource-oriented algorithms. In the price-oriented algorithms, the search is based on finding a set of prices such that supply meets demand [62-64]. Walras price-adjustment is based on the following procedure. The auctioneer starts the tatonement process with an arbitrarily chosen price-set. An arbitrary ordering of goods is chosen and the tatonement process is carried out sequentially on the various
goods. Each agent computes its demand for the first good at the given price-set and communicates it to the auctioneer. The auctioneer calculates the aggregate demand of the agents for the first good, and depending whether it is positive, negative or zero alters the good’s price. The agents calculate their demand at the new price level and communicate it to the auctioneer. Iterations continue until the aggregate demand of all the consumers for the first commodity is zero. This process then repeats for the second good, and so on.

At the end of the first cycle, only the last good has a guaranteed zero demand. However, assuming gross substitutability i.e., the aggregate demand for each good is non-decreasing in the prices of other goods, the price set established after each cycle is closer to equilibrium than the previous cycle. Even if the gross substitutability assumption is violated (as in sensor networks), the tatonement process has been found to give satisfactory results [65]. More sophisticated search algorithms that involve partial derivatives of the demand functions have been developed to search for equilibrium in parallel [58, 66].

Resource-oriented algorithms use the resources allocated to various commodities as a search parameter for finding economic equilibrium. The auctioneer starts with an initial arbitrary allocation of the available resources, after which the individual agents communicate to the auctioneer the price they are willing to pay for an additional amount. Based on this information, the auctioneer changes each agent’s allocation. This process continues until all the agents quote the same price for an additional small quantity of the commodity [67].

The resource allocation algorithms are inherently anytime since the auctioneer allocates the resources feasibly, during each iteration. This is not the case with price-
oriented algorithms since for the price-set under consideration during a iteration, there is no straightforward method of allocating the resources. However, real-time versions of price-oriented algorithms are available [58].

2.1.1.4 Auctions

As explained in section 2.1.1.2, the search for equilibrium usually involves an auction mechanism where agents send bids to an auctioneer. Auctions are “a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants [68].” Wurman et al. [68] have created a taxonomy of auctions based on three features: bidding rules, clearing policy, and information revelation policy. Based on the values taken by these features, various auctions variants including English or ascending price auction, Dutch or descending price auction, first-price sealed-bid auctions and second-price sealed-bid auctions can be defined.

In an English auction, bidders raise the price of an object until only one bidder is left. The object is then sold at that final price. In a Dutch auction, the price of an object starts at a high level and endures reduction by the auctioneer until one of the bidders declares that he is willing to pay that price. In a first-price sealed-bid auction, the auctioneer receives confidential bids, and the winner is the highest bidder who then buys the object at the bid price. In a second-price sealed-bid auction, the winner buys the object at the price of the second highest bid price. It is important to note that in the standard auction implementation, the commodities are traded sequentially and reallocated.
Auction mechanisms can be evaluated by verifying if they achieve the three following different criteria when the agent’s play their equilibrium strategies [69].

a.) Individual rationality: No agent is worse-off from participating in an auction than if it had declined to participate

b.) Budget balance: A mechanism is budget balanced if the net payment over all agents is non-negative; i.e, the auctioneer does not lose money.

c.) Optimality: Resource allocation achieved by the auction is pareto-optimal. A pareto-optimal allocation is an allocation that distributes resources in such a way that no agent becomes better off without sacrificing the interests of at least one other agent.

For the standard resource allocation problem, Theorem 2-1 presented in Section 2.1.1.2, illustrates assurance that the optimality of the auction mechanism occurs if the auction protocol reaches equilibrium. However, this is not a necessary criterion for an auction protocol to produce an optimal allocation. If the auction protocol induces the consumers to reveal their true demand utilities, i.e., if revealing the true information is the best possible agent strategy in an auction mechanism, then the auctioneer can automatically calculate the optimal allocation. When truth revelation is not guaranteed, mechanisms like the Generalized Vickrey Auction (GVA) [68], an auction protocol which uses a specialized agent payment procedure can be used to make truth revelation the dominant agent strategy. This auction mechanism can be used for optimal allocations for a wide variety of resource allocation problems including multiple goods, multiple units and other agent externalities. However, since GVA does not use a price system but
relies on a direct revelation mechanism, auction results may not always translate into meaningful prices for the individual goods or bundles. However, even in situations where there is no equilibrium exists, GVF can provide optimal allocations [60].

2.1.1.5 Combinatorial Auctions

Consider a sensor simulation scenario where a distributed sensor network tracks a moving target. Maintaining a target track requires not only measurements scheduled over the various sensors but also reservation of bandwidth for measurement communication to the fusion system. Scheduling measurements may not be of any use to the consumers unless bandwidth to communicate the measurements is available. Easily seen is that agent utilities for various goods show strong complementarities. Traditional approaches to the resource allocation problems where agents exhibit strong complementarities have their basis in either sequential or parallel auction mechanisms [70]. In a sequential auction, goods are auctioned one at a time. If consumers are interested in bundles of goods, then consumer bids are based on its expectation of what items it will win in later auctions. Sequential auctions subject the consumers to the problem of exposure, where the consumer interested in a bundle of goods does not win all the desired objects.

Also, sequential auctions involve speculating how other agents will behave in future auctions. Speculation is computationally expensive and intractable when the number of items is not small. Since agents do not have sufficient information about each other, the final outcome of sequential auctions might lead to very inefficient allocations where agents end up with combinations they do not value highly. An alternative approach is to simultaneously auction all items and make the bids publicly observable. This
reduces the importance of speculation about other agent’s preference, because of the publicly observable bids. However, parallel auctions can also lead to inefficient allocations since auctions of goods are independent of each other. Also, delayed bidding by agents to gauge other agent’s preferences before revealing their own, causes additional difficulties in parallel auctions. Proposals to fix the inefficient allocations from parallel or sequential auctions vary. One approach is to allow the agents to retract their bids [71]. A simple approach is set up an aftermarket where the agents can exchange items among themselves. However, although these approaches can remove some of the earlier described inefficiencies, they do not lead to an economic efficient allocation in general and are often communication or computation intensive.

Combinatorial auctions, where agents bid on a combination of items, allow the bidders to express the synergistic relationship between goods [72]. However, combinatorial auctions can support inefficient equilibriums [73] (the equilibrium need not be an optimal solution). In this case, a resource allocation algorithm that relies on equilibrium calculation is no longer sought. However, if the agent payoff mechanism used in a combinatorial auction makes truth revelation the dominant agent strategy, then the auctioneer can still optimally allocate the goods. An additional difficulty that exists in combinatorial auctions is winner determination. Winner determination is an n-p complete problem, as described below. Recent years have seen great strides into research in this problem, resulting in algorithms that are very fast even for large problems [70], [72], [73].
2.1.1.5.1 Formulation of the Winner Determination Problem

The winner determination problem in a combinatorial auction can be defined as the following [73]:

- Set of items M, the auctioneer needs to sell: M = {1,2…m}
- Set of bids B, the buyers send to the auctioneer B = {1, 2…b_n}

where each bid \( b_j \) is a tuple \(<S_j, p_j>\) \(S_j\) is the set of items \((S_j \subseteq M)\) and \(p_j\) is the price.

The winner determination problem is to assign the states winning or losing to each bid:

\[
\text{Max} \sum_{j=1}^{n} p_j x_j \quad \text{s.t.} \quad \sum_{i \in S_j} x_i \leq 1, \ i \in \{1..m\}
\]

\(x_j \in \{0, 1\}\)

As stated earlier, this problem is np-complete and has no approximation to a ratio of \(n^{1-c}\) in polynomial time (unless \(p = np\)) [73].

2.1.1.5.2 Winner Determination Algorithms

Many algorithms have been proposed in the literature for winner determination, including exhaustive enumeration, dynamic programming, allowing the allowable combinations, and specialized search procedures. Sandholm[73] has conducted a detailed analysis of the available algorithms. In an exhaustive procedure, all exhaustive partitions are formulated and evaluated. An exhaustive enumeration is defined as a partition of items where each item is included in exactly one subset of partition. For instance, Figure 2-2 shows the exhaustive partition of a 3-item example.
This procedure is computationally intensive where the upper boundary on the number of exhaustive partitions is $O(m^2)$, and the lower boundary is $w(m^2/2)$ where $m$ is the number of items in an auction. Thus, it is impossible to enumerate all the partitions unless the number of items is very small. Dynamic programming also exhibits the same drawback as exhaustive enumeration since it too searches the entire search space irrespective of the number of bids received. Other approaches depend on restricting the combinations on which bids are allowed and are, thus, susceptible to the same economic inefficiencies as sequential and parallel auctions. Approximation algorithms do not provide worst case approximation guarantees since the winner-determination problem is inapproximable.
Although the search space of the combination of bids is very large, in practice the number of bids received by the auctioneer is a very small subset of the sample space. For example, in an auction with 100 items, the total number of possible combinations is \(2^{100}\). However, for any number of consumers to generate such a large number of bids, is impossible.

Specialized search algorithms which can exploit this fact have been proposed. These algorithms are based on ordering either items [73] or bids [70] in a tree structure and performing a branch-and-bound tree search.

The algorithm that branches on items (called as CABOI from now on) employs two search procedures. The first procedure generates children in the search tree and the second procedure searches the children for an optimal solution. Also, a number of heuristics are often used to make the search fast including:

a. Keeping the highest bid for a combination,

b. Removing uncompetitive bids,

c. Decomposing the problem into smaller, independent units,

d. Searching for and eliminating noncompetitive tuples of bids.

The number of nodes in the search tree for CABOI has an upper bound of \((n/m)^m\) where \(n\) is the number of bids, and \(m\) is the number of items. This bound is polynomial in the number of bids and exponential in the number of items in the auction. The average performance of the algorithm is much better than the upper bound. The performance of the algorithm on a wide variety of bid distributions has shown very promising results [73].
CABOB is a search algorithm that branches on bids, where each search node has two children, the sub-space where the bid is labeled as winning and the sub-space where the bid is labeled as losing. Structural improvements to reduce search tree size, efficient data structures, and optimizations at search nodes based on driving towards, identifying and solving tractable special cases form the backbone of this algorithm. The number of leaves in the search tree formed is bounded by \((n/\lceil m/k \rceil + 1)^{\lceil m/k \rceil}\), where \(n\) is number of bids, \(m\) is the number of items, and \(k\) is the number of items in the bid with the smallest number of items. This is polynomial in the number of bids. Although the worst case bound is exponential in terms of number of items, the average case performance is significantly better.
Chapter 3

Market Architecture for Sensor Management

3.1 Introduction

As summarized in chapter 2, extensive research methodologies have been suggested in research literature for sensor management. However, none of these provide a comprehensive sensor management paradigm for reasons explained in 2.1.

We believe that market-oriented programming [74] provides a valuable paradigm for this distributed resource allocation problem since markets have the inherent ability to deal with non-commensurate entities. As explained in section 2.1.1, markets can be used for finding optimal allocations in a cooperative environment and there is a one-to-one mapping between a sensor management scenario and traditional market. The market design proposed for sensor management domain is shown in Figure 3-1. The mission manager (MM) allocates the various high-level tasks/goals and their corresponding budgets to the consumer agents in the market. The sensor manager holds an auction where the goods owned by the individual sensors like measurement schedules, battery power and processing capacity and bandwidth of the transmission channels are auctioned off to the consumers. Consumers bid for various goods required for accomplishing the tasks allotted to them, using the budget provided by the Mission Manager. The various consumers in the markets could be working for one MM under one mission or for different MMs. This research does not model the process used by MM to
decide consumer responsibilities or to decide goal budgets. The focus of this research is the design of algorithms for optimally allocating network resources to the various tasks for known high-level goal priorities.

Figure 3-1: Market architecture for sensor management

The market design in Figure 3-1 cannot be directly adapted to implement a market mechanism for sensor management because of a number of issues presented below. First, as shown in Figure 3-1, usually consumers in a sensor network are interested in commodities like target tracks, environmental searches. On the other hand, the individual sellers in the sensor management domain can produce only measurement data or communication bandwidth. Thus, clearly, some method for bundling goods produced by various sellers is essential to create commodities that consumers are interested in and can bid for. Also, in a traditional auction mechanism, a single item allocated only to a single consumer. This is not the case with the sensor network domain. For example, if consumer A is interested in scanning an area at rate $r_1$ and consumer B needs to scan the same area
at rate, $r_2 > r_1$, then a single commodity, i.e. a scan at rate $r_2$ communicated to both agents will satisfy both of them. Thus, there can be a one-to-many mapping between the commodities in the market and allocations to the agents. Another difficulty is the fact that the simple mechanism of labeling the bids as either winning or losing may push the optimum allocation beyond consideration. For example, if consumer $A$ sends a bid for scanning area, $a_1$, and consumer $B$ sends a bid for scanning area, $a_2$, then the actual optimum allocation might be scanning $a_1 \cap a_2$. However, this allocation is not bid for. In addition, different combinations of sensors can be used to track the target, where each combination of sensors provides a different quality of service (QoS). To correctly bid, consumers would need to know the operating parameters of different sensors, and calculate the QoS for various combinations of sensors. A more natural approach is to allow consumer agents to bid on high-level tasks.

In addition to these difficulties, sensor management generally has very strict real time constraints. This implies that the market mechanism should be able to produce feasible if not optimum allocations in very little time.

For these reasons, traditional auction mechanisms do not directly apply to sensor management scenarios. A specialized architecture, Market Architecture for Sensor Management (MASM) has been developed for this purpose. MASM closely resembles the architecture proposed by Denton et al. [23]. Figure 3-2 shows this study’s single-platform design, which derives from the sensor management architecture proposed by R.V. Denton and his colleagues. The main MASM components are the mission manager and the sensor manager, which are described below.
A. Mission Manager (MM)

This component assesses mission-level decisions, such as deciding the priority of the various goals for accomplishing mission objectives, and allocating these goal responsibilities to consumer agents. A model by which MM calculates the priorities of the various goals to the overall mission objective is not part of this research. However, a discussion of a few aspects to provide a frame of reference is given here. Within the
mission manager, approaches such as goal lattices can be used to measure the criticality of various low-level goals to the overall mission goals, and thus help to determine their respective budgets. Kenneth Hintz and Gregory McIntyre [75] used goal lattices to compute the relative weights of actionable tasks (such as tracking) on the basis of high-level mission goals. Rajani Muraleedharan and her colleagues [76] used goal lattices to determine weights for combining various objectives to optimize routing in a sensor network. Newer developments include dynamic goal lattices [45] to support more dynamic goal generation from a set of predefined goals.

B. Sensor Manager (SM)

The SM component allocates sensors to various tasks, schedules sensors, and diagnoses faults. It also allocates all necessary network resources, including communication bandwidth and battery power, so it can consider trade-offs such as battery power versus communication that often must be hard-wired into other systems. Individual sensors operate per the SM schedule and communicate the measurements to a fusion center—generally another sensor that performs any necessary data fusion processing before sending the information back to the consuming processes. The SM selects the fusion center based on such constraints as battery power, processing capability, and communication requirements and sends its output to the requesting consumer. The SM provides a competitive market for sensor resource buyers and sellers. Sensors and transmission channels are modeled as sellers so that they can “sell” their schedules (that is, their “attention”) and raw bandwidth, respectively. However, sensor network consumers often seek higher-end products such as target tracking, target identification, and environmental searches. To address these commensurability issues, the SM bundles
raw resources into products that consumers want using a service chart database and a specialized bid formulator module. The service chart specifies detailed domain information such as sensors’ field locations and characteristics and the available communication bandwidth. The SM accepts consumer bids and transmits them to the bid formulator. These bids are in the form of \(<t, p>\), where \(t\) is a task description (including the minimum task quality acceptable to the consumer) and \(p\) is the price the consumer is willing to pay. Using the service chart, the bid formulator enumerates the different possible allocations for each task and uses the consumer bid price to determine price quotes for each possible allocation. In this process, the bid formulator must evaluate a particular allocation’s value to the overall consumer task. For example, a consumer might be willing to pay more for a specific sensor with “excellent” applicability to a perceived threat than for another sensor with “good” applicability. Such applicability might be based on target characteristics, sensor capabilities and the observing environment. The bid formulator must also consider synergistic constraints—for example, sensors A and B working together might provide more accurate tracking than the more powerful sensor C working alone. Once the bids are formulated, the SM conducts an auction to set prices and allocate available sensors as needed. In combinatorial auctions, which sell bundled goods, buyers can express their preferences for a combination of goods, thus efficiently representing synergies (or lack thereof) between goods. For example, most buyers would much prefer to buy the combination of a left shoe and a right shoe over either one independently. In a combinatorial auction, bidders offer bids in the form \(B_i = <b_i, p_i>\), where \(b_i\) is the bundle of resources and \(p_i\) is the price the bidder is willing to pay for the bundle \(b_i\). The winner determination problem is to find a resource allocation that
maximizes revenue given the constraint that each resource can be sold to no more than
one bidder. MASM provides its service chart/bid formulator functionality in two different
modes, either exact service mappings (E-MASM) or approximate service mappings (A-
MASM). When the number of sensors is small and the real-time constraints are relaxed,
then E-MASM mode provides an exact service mapping. For each possible combination
of resources that can be used for a given task, the utility of assigning the combination to
the task is explicitly calculated, using domain information and task specific utility
functions provided by service chart. For example, given $n$ sensors in the network, and $m$
bids, then $2^n m$ bids on resource combinations might have to be formulated. A standard
combinatorial auction winner determination [77] then determines the optimal allocation.
An approach to accelerate the bid formulation auctions and the winner determination
optimization is to restrict the type of bids considered for resource allocation. For
example, a bound on the maximum number of items in a bid could be used. Polynomial
algorithms for bids with certain special structures [73] are available. However, imposing
constraints on the types of bids leads to market inefficiency, since consumers are not
allowed to bid on certain combinations of items. An alternate approach is to use domain-
specific knowledge to intelligently restrict the number of resource bids formulated. For
example, the types of resources that could be used to accomplish a particular task might
be limited, which reduces the combinations of resources for consideration.

When the number of sensors is large and real-time constraints are strict, explicit
mappings are no longer feasible, and the A-MASM mode is used. In this mode, MASM
provides dynamic service mapping, using a function-estimation neural network in place
of the service chart database. A neural network trained offline is used to calculate the
utility obtained for allocating a given combination of sensors to a consumer task. Instead of the bid formulator explicitly formulating combinatorial bids for each consumer bid, MASM searches the search space directly using a polynomial, anytime evolutionary algorithm. Chapters 5 and Chapters 6 describe the approximate algorithms used in A-MASM in greater detail.

### 3.2 CCA Protocol

MASM uses discrete time slots to schedule resources. Most consumer tasks, like tracking a target, require acquiring resources over multiple time-slots. However, having consumers submit bids for multiple time slots requires using scarce communication resources. If consumer bids are allowed to remain active over multiple allocation rounds, communication costs involved in the market will reduce drastically. For this purpose, a Continuous Combinatorial Auction (CCA) protocol has been formulated. The CCA protocol runs each of the following steps (except initialization, which runs once at the start of operations) during each round of scheduling (see Figure 3-3 for flowchart of CCA).
1. *Market Initialization:* The auctioneer initializes the prices for all the resources. It informs the consumers of the set of tasks that it will accept bids for.
2. **Bids Update:** At the beginning of each round, consumers can

a) send new bids,

b) remove their current bids from the auction, or

c) modify the parameters of their existing bids

Consumer bids are of the type \(<t, p>\) where \(t\) is the task description which includes the task type and final task quality desired by the consumer, and \(p\) is the price that the consumer is willing to pay. For example, the task description for a bid to search a particular grid, \(x\), so that the uncertainty of target presence (as measured by entropy) < 0.001 is as follows:

\[
\text{(type: search} \\
\text{entity: grid no } x \\
\text{quality: (entropy < 0.001))}
\]

The auctioneer predefines the set of tasks for which the consumer can bid and the bid format.

3. **Round Initialization:** The auctioneer accepts new bids or updates to existing bids during each round of scheduling from the auction. If no message regarding a particular bid is received by the auctioneer, the bid stays active and competes for resources in the current auction round. The auctioneer initializes the total-cost of the resources expended on each new bid, \(b\), \(C_b\) to zero.

4. **Resource Bid Formulation:** Since consumer bids are for high-level tasks, the auctioneer needs to compose bids for actual resources from them. This responsibility is handled by the bid-formulator module in E-MASM (explicit formulation of bids for resources is not required in A-MASM). Let the consumer bid \(b\) on a high level task \(T_{w}\)
at time \( t \) at price \( p_b \) (\( q_t \) describe the task parameters at time \( t \)). A high level task, such as tracking a target to a required accuracy, might require resources over multiple rounds of scheduling. For each time slot \( t \), the auctioneer constructs bids on each resource set, \( S \), that can be allotted to task, \( T \). The auctioneer needs to calculate the price associated with the bid on resource set \( S \), for task \( T \), based on the consumer bid price \( p_b \). It has to be noted that this price is not charged to the consumer. Consumer is charged only at the end of task completion or if he chooses to withdraw a bid before a task could be completed by SM. The price for a resource set \( S \) has to be determined by the auctioneer to prioritize between different tasks during a particular schedule. For this purpose, a novel mechanism of price calculation has been devised. For a resource set \( S \), the auctioneer computes the bid price for a resource set as the percentage of the consumer task completed by the resource set. Computation of the percentage of task completed by a resource set \( S \), is in terms of readings of a canonical sensor, \( A \), as follows. Let a particular task \( T_{q_t} \) require, on average, \( n_a \) consecutive schedules of the standard sensor \( A \) to be completed. Instead, if a resource bundle \( S \) is used at time \( t = 1 \), the expected number of standard sensor readings required is reduced to \( n_{a}' \). The percentage of task completed by resource set \( S \) is equal to the percentage savings in the number of sensor \( A \) readings required.

\[
f_{S,b,t} = \frac{(n_a - n_{a}')}{n_a} \quad 3.1
\]

The fraction \( f_{s,b,t} \) has price-based interpretation also. The percentage of task completed by resource set \( S \) can be approximated as the ratio of cost of resources in \( S \) (as determined by the market) to the expected value of the total cost of resources required by SM for accomplishing the task, as shown in \( Eq. \ 3.2 \) and \( Eq. \ 3.3 \).
\[ p_{s,b,t} = p_b \cdot f_{s,b,t} \quad 3.2 \]
\[ f_{s,b,t} = p_t(S)/E(c(T_{q,t})) \quad 3.3 \]

where \( p_t(S) \) is the cost of resource set \( S \) at time \( t \) and \( E(c(T_{q,t})) \) is the expected cost of accomplishing task \( T_{q,t} \).

The cost of resources in \( S \) at \( t \) can be calculated as the sum of the prices of the resources comprising \( S \) at time \( t \) (Eq. 3.4)

\[ p_t(S) = \sum_{i \in S} p_t(i) \quad 3.4 \]

where \( p_t(i) \) is the price of the \( i \)-th commodity in \( S \) at time \( t \).

The calculation of expected cost of accomplishing task \( T_{q,t} \), \( E(c(T_{q,t})) \), is not straightforward. For illustration, consider a simple sensor scheduling scenario involving one task \( T \) and two sensors, sensor A and sensor B having the same operating characteristics. Let the prices of sensor A and sensor B be time invariant constants \( p_a \) and \( p_b \). Assume that the scheduling algorithm is to pick either sensor A (with probability \( q \)) or sensor B (with probability \( (1-q) \)) and assign it to \( T \) during each round of scheduling. If \( T \) requires a total of \( n \) consecutive schedules to be completed, then \( E(c(T_{q,t})) \) has the form:

\[ E(c(T_{q,t})) = \sum_{k=1}^{n} \frac{n!}{(n-k)!k!} q^k (1-q)^{n-k} (kp_a + (n-k)p_b) \quad 3.5 \]

For a more general market-based sensor management scenario, the probability that a particular sensor has allocation to a task during any time interval is difficult to calculate, particularly since new tasks may arrive at random. Also, the expected time
taken by the SM to complete the task, and the prices of individual sensors at different times are also stochastic. These issues, combined with the fact that the calculations should be computed in real-time make exact computation \( f_{S,b,t} \) difficult. One approach would be to use historic data to estimate \( E(c(T_q)) \) heuristically. This study uses a much simpler approach, which adopts a particular sensor as the canonical sensor to calculate expected costs. Let a particular task \( T_q \) require, on average, \( n_a \) consecutive schedules of the standard sensor A to be completed. Instead, if a resource bundle \( S \) is used at time \( t \), the expected number of standard sensor readings required is reduced to \( n_a' \). The expected percentage savings in cost of accomplishing the task achieved after using \( S \), assuming sensor A is used throughout for \( T_q \) at a constant price \( p_a \) is:

\[
f_{S,b,t} = (n_a - n_a')p_a/n_a \cdot p_a = (n_a - n_a')/n_a.
\]

There is a possibility that \( f_{S,b,t} \) is negative (\( n_a < n_a' \)). For example, in spite of allocating sensing resources, the inaccuracy associated with a target estimate might increase with time. To avoid negative prices for bundles of resources, the bid prices are calculated as:

\[
p_{b,b,t} = p_b f_{S,b,t} - p_b f_{\phi} \]

where \( \phi \) is the null set

Calculation of \( n_a \) and \( n_a' \) can be made faster, by storing task specific performance data of the canonical sensor as QoS charts. For example, consider a consumer bid for searching a particular grid for potential threats, where QoS is measured in terms of entropy. The consumer bids to reduce the entropy of a grid from \( e_1 \) to \( e_2 \). The QoS chart,
the expected fall in entropy with the number of readings with the standard sensor, is
given in Figure 3-4. If resource bundle $S$ is expected to reduce the entropy form $e_1$ to $e_i$, then

$$f_{S,b,i} = \frac{(n_1 - n_f)}{(n_1 - n_2)}$$  \hspace{1cm} 3.8

Section 3.6 illustrates the creation of the QoS chart for a sample task (Figure 3-4).

Resource formulation is the slowest link in CCA protocol. Heuristics could speed the bid formulation step. For example, for certain tasks, it may be feasible to use only a few kinds of resources. However, in the worst case scenario, $n$ consumer bids and $m$ resources may require $m2^n$ resources bids. This is clearly infeasible in the case of large systems. So, E-MASM is not scalable to large systems, limiting it’s effectiveness. A-MASM formulation avoids explicit bid formulation, and hence maintains polynomial run-times, both in number of resources and consumers.

5. Resource Allocation: Resource bids, obtained from Step 3 are exclusive-OR bids in the form, $<S_1, P_1> \ xor <S_2, P_2> \ldots xor <S_n, P_n>$. This bid indicates that the consumer is willing to pay a price, $P_1$, for the resource bundle, $S_1$, and a price, $P_2$, for $S_2$, but only the maximum of $P_1$ and $P_2$ for $S_1 \lor S_2$. The auctioneer needs to translate these bids to OR bids, so that standard integer programming formulations [77] can be used.
Figure 3-4: Illustration of calculation of bid prices for resource bundles using QoS chart

OR bid of the form \(<S_1, P_1> \text{ or } <S_2, P_2> \ldots \text{ or } <S_n, P_n>\) indicates that consumer is willing to pay \(P_1+P_2\) for the bundle \(S_1 \cup S_2\). Translation of bids to OR format can be accomplished by addition of phantom items [79]. The idea is to translate an exclusive-OR bid, \(B-xor\), of the form, \(<S_1, P_1> \text{ xor } <S_2, P_2> \ldots \text{ xor } <S_n, P_n>\) into a \(B-or\) bid of the form \(<S_{1b} \cup b, P_1> \text{ or } <S_{2b} \cup b, P_2> \ldots \text{ or } <S_{nb} \cup b, P_n>\), where \(b\) is a phantom item. The phantom item, \(b\), ensures that a maximum of only one bid from the OR bids can be labeled as the winner (since each item can be allocated to maximum of one bid). Once the bids are translated into the “OR” format, the winner determination problem becomes a standard integer programming (IP) problem, that can be characterized as:
where $x_j$ is 1 if the bid is accepted in the final allocation and 0 otherwise. The IP problem can be solved using a commercial software package like CPLEX. The winner determination problem is np-hard, and in theory, this resource allocation step could prove computationally expensive for E-MASM. Performance of winner determination algorithms depends greatly on the characteristics of the probability distribution which generates bids. For example, the time taken by a thousand bids, and hundred items problem on a 2.8 GHz Pentium IV processor varied between 0.001 seconds to 5000 seconds, depending on the bid distribution. However, we found that the problems generated by the sensor network simulation are relatively easy for CPLEX (see Section 3.6.1).

6. Round Termination: The auctioneer updates the costs of resources expended on a particular bid by adding the price of its allocated bundle. For each consumer bid $b$, the cost of resources allocated to the bid is updated as

$$C_b = C_b + p_t(S_i), \quad 3.10$$

where $S_i$ is the bundle allocated to $b$ during the current round.

Also, the auctioneer verifies if the task quality required by each consumer bid is achieved. Bids for which the requested task is completed are removed from the auction and the consumer is sent the completed task details. If the task of a particular bid $b$ is
complete, the bidding consumer is charged the minimum of the bid price or the cost of resources spends on the task by SM.

$$payment_b = \min(C_b, p_b)$$  \hspace{1cm} 3.11

where $payment_b$ is the fee charged to the consumer, $p_b$ is the bid price, and $C_b$ is the total cost of the resources allocated to the bid. This fee structure ensures that no consumer is charged more than his bid price for any task. When $C_b < p_b$, the consumer has a positive surplus of $p_b - C_b$. An alternate fee structure that divides the surplus between the SM and consumer is as follows:

$$payment_b = \min(C_b, p_b) + H(p_b - C_b) \gamma (p_b - C_b)$$  \hspace{1cm} 3.12

where $H(x)$ is the Heaviside step function and $\gamma$ is the percentage of surplus given to SM. If the consumer withdraws a bid before the task demanded in the bid is completed, the SM charges the consumer

$$payment_b = \min(C_b, f_{b, p_b}) + H(f_{b, p_b} - C_b) \gamma (f_{b, p_b} - C_b)$$  \hspace{1cm} 3.13

where $f_{b, p}$ is the percentage of the task demanded in the consumer bid that is already completed by the SM. This is calculated using QOS charts (as described in the resource bid formulation step).

Finally, the auctioneer updates the prices of the resources based on the demand in the current round (Section 3.3 explains how prices are updated). The auctioneer also releases a market report that contains statistics about auction outcomes in the previous schedules, once every $n$ rounds. The auctioneer updates the resources about their schedules during the current round and sleeps until the next round of scheduling begins.
An issue with CCA is that bids stay in existence for multiple rounds and deadlines for task completion are not accepted by the auctioneer. However, consumers can enforce deadlines for tasks, by observing the QoS offered by the SM for a task at the current prices and changing bid prices accordingly. For example, the failure of the allocated resources to provide the desired QoS required to meet the task deadline is an indication that the consumer should increase its current bid price.

3.3 Pricing Mechanisms

Each task allocation is accompanied by some resource expenditure, including battery power, processing power and bandwidth. When these resources are scarce, it is essential to model the cost of these goods when a task allocation is decided. In a distributed sensor network, each sensor has a finite battery power and finite processing power. Also, transmission channels available in a sensor network have a limited bandwidth. For example, in the DARPA sensor network [48], radio frequency channels provide a bandwidth of 56 kb/s per channel. Bandwidth is required for the sensor manager to communicate schedules to the individual sensors and collect requests from consumers. Individual sensors need to communicate measurements to the fusion center and, thereby use bandwidth. Since sensors are distributed, synchronization of communication to avoid message collisions is infeasible or, at best, inefficient. Unsynchronized communication between the sensor networks using the limited number of transmission channels leads to collisions and error prone messages [48]. The loss of data may effect timely communication of data and hinder the efficient functioning of
sensor network. It is necessary to spread the communication load over the transmission channels and prevent the over congestion of transmission channels. For this purpose, the SM should also model bandwidth cost of the various tasks.

To set prices for individual resources, CCA uses a pricing protocol similar to the tatonement process. Tatonement is an iterative procedure for finding equilibrium prices based on the search parameter (refer to 2.1.1.3 for details). To use a tatonement process in MASM, the supply and demand functions at a particular price for a resource need to be modeled. MASM estimates these functions using the current rate of utilization of a resource. Prices for individual resources are set to zero, during sensor network initialization. After each round of scheduling, prices \( p_t \) are updated depending on whether the current rate of utilization of the resource \( r_t \) is greater than the available rate for utilization \( c_t \).

\[
p_{t+1} = p_t + \tau^* (r_t - c_t),
\]

where \( \tau \) is the constant which determines the rate at prices are updated.

Calculation of \( r_t \) and \( c_t \) is dependant on the resource being modeled. For example, for sensors, we have used the available battery power. Let Sensor A be endowed with initial battery power \( p_i \) and assume that Sensor A needs to be available for a total operating time of \( T \). At time \( t \), if the available battery power is \( b_t \), then \( r_t \) and \( c_t \) can be calculated using Eq. 3.15 and Eq. 3.16.

\[
r_t = (b_i - b_t)/t \tag{3.15}
\]

\[
a_t = (b_t)/(T-t) \tag{3.16}
\]
Ideally, the tatonement process would update the price of one resource and run a winner determination algorithm to find the new demand for resources. Then, price updates for the second resources occurs, and so on. However, because of time and communication constraints, all the price updates take place simultaneously using the current rate of utilization, during each round. However, the expectation is that the results between the two approaches will not be very different, since the rate of utilizations are moving averages and do not vary significantly based on the usage during the current time slot.

3.4 Analysis of CCA

Many simple scheduling approaches like first-come-first served, highest-priority first, earliest-deadline-first have been designed which have low computational requirements (see [8] for an exhaustive survey). For this reason, these algorithms have been used extensively in time constrained environments like sensor networks and CPU scheduling etc. CCA has greater computational complexity than the standard scheduling algorithms. However, CCA-based scheduling offers important advantages, as compared to the standard algorithms. This section describes these advantages using illustrative examples.

The analysis, first, provides a description of how CCA schedules tasks for a standard scheduling problem. Consider a simple scheduling problem with N tasks and M identical machines. The i-th task requires n_i machine schedules and has p_i priority.
Theorem 3.1 For the standard scheduling problem defined above, assuming that task priorities correspond to the bid-prices on the task; CCA is a priority-based scheduling algorithm.

Proof: A higher priority task offers a higher price per unit schedule. If a given machine can be allocated to either of two different tasks, the auctioneer obtains greater revenue by allocating it to the higher priority task. As a result, tasks get completed in the order of their priority.

Therefore, for the simple scheduling problem described above, CCA offers no advantages, when compared to a priority-based scheduling algorithm. The latter can be used with much lower computational load. However, sensor management scheduling differs from the simple scheduling algorithm in the following ways:

1. Resource utilization has costs associated with it. For example, a sensor reading costs battery power. Communication of messages also involves energy consumption. Therefore, fusing multiple sensors’ readings for target tracking, involves the additional cost of sensor’s communicating their reading to the fusion node. CCA-based scheduling measures the trade-offs between the costs involved and the utility-obtained from allocating a resource bundle to a task (see Section 3.6.2 for illustrative results). Standard scheduling algorithms do not offer this cost-benefit analysis.

2. Sensor capacities are not additive. In the scheduling problem described, it was assumed that machine capacities are additive. That is, if n machines are used for completing a task, n schedules of the task are completed. This is not the case in sensor management. For example, assume allocating sensor A to a task during the current round completes x% of T (as measured by QOS-charts) and sensor B’s reading completes y% of T. Allocating
the combination of sensor A and sensor B will complete z\% of T, which is generally different from (x + y)\%. CCA considers the synergy between resources before allocating them to tasks. This offers an obvious advantage as compared to the standard approaches like priority-based queues. For example, the readings of two-bearing only sensors can be used to obtain both range and bearings information about a target. These synergies become self-evident, when a combinatorial auction based resource allocation mechanism is used.

3.5 Simulation Environment

A simulation environment consisting of a two dimensional search area involving multiple targets, multi-consumer and multiple sensors has been developed for testing MASM and comparing its performance with other sensor management approaches. The design of the sensor network including the communication channel is inspired by the DARPA sensor network that was implemented to carry out research in sensor management domain [48]. The various sensor parameters are based on realistic sensor models and are obtained from [44]. A significant note is that this sensor network is different from the emerging field of “smart-dust”, where the number of sensors is to the order of a few hundred thousands. We also extended MASM to smart dust environments [79]. However, smart-dust related MASM work is not a part of this thesis.

To generate a simple urban warfare model with a sensor network which has bandwidth and battery power constraints, the simulation model was developed based on some modifications to the multi-sensor, multi-target simulation model developed by McIntyre
and Hintz [1, 43]. The simulation model is a 10 km by 10 km urban area with arbitrary road structures. This model has a few simplifying assumptions, since the primary focus of the current research is different. The various components of the simulation are explained below.

A. **Consumers:** The environment consists of a set of consumer agents, which search for and destroy targets. Consumer agents have the ability to attack any position within a range of $r$ meters and any target that falls within $\gamma$ meters of attacked position is destroyed. The consumers have no sensing resources, and they depend on the sensor network for obtaining information about the environment. They bid for sensor resources during each round of scheduling and update their status based on information provided by the sensor manager. Initially, consumers move along the city with constant velocity $v_c$, searching for targets. They use the sensor network’s resource to search for potential targets, and initialize target tracks if the probability of target existence within their range exceeds a threshold $p_{threshold}$. Upon initialization of a target track, consumers can attack a target if the confidence interval of the target’s position estimate is less than $\gamma$ meters. Hence, they must track the target to the required accuracy before attacking it. This is again accomplished by buying sensing resources from the sensor network. Consumers are assumed to have a utility, $u_t$, for destroying a target. To divide the overall utility into utilities for search and track tasks, consumer agents initially use equal priorities. During the simulation run, consumers update the search to track budget ratio using the learning described in Chapter 5.

B. **Targets:** Targets are randomly distributed throughout the search area. They move randomly along the city roads with constant velocity, $v_t$, corrupted by a Gaussian white
noise with variance $Q$. Two different types of targets are in the model ($T_1$ and $T_2$). $T_1$ targets do not have offensive capabilities. $T_2$ targets destroy the consumer if they stay within $r$ meters of the consumer for more than $t_{\text{kill}}$ schedules. Only $T_1$ targets were used in the simulation experiments, unless otherwise specified.

C. **Sensors:** The simulation models several different kinds of sensors, including sensors that provide range and bearing, bearings-only sensors and electronic support measure (ESM) sensors. Measurements of two bearings-only sensors which are not located at the same position, can be combined to create both range and bearing estimates and can be used as a pseudo-sensor. A formal way of modeling sensors is to model their properties, such as bandwidth, wavelength, duration of waveform, signal power per pulse, receiver noise strength diameter of radar aperture. A much simpler modeling technique, in which a sensor’s characteristics are characterized by three parameters, its probability of detection, $P_D$, probability of false alarm, $P_{FA}$ and beamwidth, is used in this simulation. The simulation environment has eight different sensors of five different types, located on two different platforms orthogonal to each other. The operating characters of the various sensors are given in Table 3-1. Both the platforms are 100 km away from the search area. Since the distance of the sensors from the city is large, use of a small angle approximation $s = r \theta$, where $s$ is the length of area that falls under the sensor’s beamwidth, $\theta$ is a beamwidth of the sensor in radians and $r$ is the distance of the sensor platform to the city (100km) is possible. For a detailed description of the sensor modeling techniques adopted in the simulation, refer to [17]. Each sensor has a battery with $e_{\text{initial}}$ units of energy. For the purpose of brevity, all the sensor tasks are assumed to cost zero.
energy, except the task of transmitting messages. The energy expended in transmitting a message of \( m \) bytes over a distance of \( d \) meters is calculated as \( \alpha d^2 m \).

<table>
<thead>
<tr>
<th>Sensor No.</th>
<th>Type</th>
<th>Range</th>
<th>Bearing</th>
<th>Axis</th>
<th>( P_D )</th>
<th>( P_{VA} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doppler</td>
<td>90m +/- 10%</td>
<td>1° = 6( \sigma )</td>
<td>x</td>
<td>0.95</td>
<td>0.001</td>
</tr>
<tr>
<td>2</td>
<td>Radar</td>
<td>30m +/- 10%</td>
<td>0.1° = 6( \sigma )</td>
<td>x</td>
<td>0.95</td>
<td>0.001</td>
</tr>
<tr>
<td>3</td>
<td>FLIR</td>
<td>NA</td>
<td>0.1° = 6( \sigma )</td>
<td>y</td>
<td>0.99</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>ESM</td>
<td>NA</td>
<td>1° = 6( \sigma )</td>
<td>x</td>
<td>0.5</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>IR</td>
<td>NA</td>
<td>100( \mu )rad=6 ( \sigma )</td>
<td>x</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>6</td>
<td>Radar</td>
<td>30m +/- 10%</td>
<td>0.1° = 6( \sigma )</td>
<td>y</td>
<td>0.95</td>
<td>0.001</td>
</tr>
<tr>
<td>7</td>
<td>Doppler</td>
<td>90m +/- 10%</td>
<td>1° = 6( \sigma )</td>
<td>y</td>
<td>0.95</td>
<td>0.001</td>
</tr>
<tr>
<td>8</td>
<td>Radar</td>
<td>30m +/- 10%</td>
<td>0.1° = 6( \sigma )</td>
<td>y</td>
<td>0.95</td>
<td>0.001</td>
</tr>
</tbody>
</table>

**D. Communication Channel:** For communication purposes, a Radio Frequency (RF) communication channel with capacity \( C \) is used. All the messages are assumed to be of uniform size of \( M \) bytes. The communication protocol used is contention based protocol (like CSMA/CD) where each agent with a message to communicate senses to see if the channel is busy and transmits if it is not. If two entities attempt to transmit at the same time, they back-off and wait a random amount of time before retrying. The time taken for communication, via this channel, for a fixed number of messages is stochastic. SM enforces the bandwidth constraint by restricting the probability that time taken for communication is greater than \( t^{\text{com}} \) to less than \( \beta \% \). Thus, bandwidth constraint is a chanced constraint.
E. Sensor Manager (SM): Since the number of sensors is not large, using the E-MASM formulation is possible. Bids on two types of tasks, search and track, are accepted by SM. The QoS for search tasks is in terms of entropy and for tracking, norm of the estimate covariance is used. For search tasks, QoS charts are created as follows. Assume that the canonical sensor A has probability of detection $P_D$ and probability of false detection $P_{FA}$. Let the initial probability of a threat being in the grid be $p_t = 0.5$ (with $p_t = 0.5$, the grid has the highest possible entropy). The probability of positive reading is:

$$p^a_{x=1} = P_{FA} . \quad 3.17$$

The probability of negative reading is:

$$p^a_{x=0} = 1 - P_{FA} . \quad 3.18$$

Using the Bayesian update, in case of a positive reading, the probability of a threat being present is updated, according to Eq. 3.19 as,

$$p^e_{x=1} = P_D * p_t / p^e_{x=1} , \quad 3.19$$

where $p^e_{x=1}$ is:

$$p^e_{x=1} = P_D * p_t + P_{FA} * p_{nt} . \quad 3.20$$

In case of negative reading, probability of threat being present is updated to

$$p^e_{x=0} = p_t * (1-P_D) / p^e_{x=0} , \quad 3.21$$

where $p^e_{x=0}$ is defined as:

$$p^e_{x=0} = (1-P_D) * p_t + p_{nt} * (1-P_{FA}) . \quad 3.22$$
Finally, the expected value of the updated probability after one reading by sensor A, if the target is not present is:

$$p^e_{\nu NT} = p^e_{\nu x=1} * p^a_{x=1} + p^a_{x=0} * p^e_{\nu x=0}.$$  \hspace{1cm} 3.23

The expected value of updated probability after one reading by Sensor A, if target is present, $p^e_{\nu T}$, can be calculated as follows:

$$p_t * \text{entropy}(p^e_{\nu T}) + (1-p_t) * \text{entropy}(p^e_{\nu TP}),$$  \hspace{1cm} 3.24

where $\text{entropy}(p)$ is:

$$\text{entropy} (p) = -p \log(p) - (1-p) \log(1-p).$$  \hspace{1cm} 3.25

In a similar fashion, we can calculate the expected value of entropy after $n$ readings of sensor A. We use a similar approach to generate QoS mappings for all tasks for which SM accepts bids. The parameter values used in the simulation are shown in Table 3-2.
Table 3-2: Parameter Values used in Simulation

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>#time slots</td>
<td>500</td>
<td>Total number of resource allocation schedules</td>
</tr>
<tr>
<td>n_c</td>
<td>5</td>
<td>No of consumers</td>
</tr>
<tr>
<td>n_t</td>
<td>10</td>
<td>No of targets</td>
</tr>
<tr>
<td>n_{t_o}</td>
<td>2</td>
<td>No of targets with offensive capabilities</td>
</tr>
<tr>
<td>v_c</td>
<td>100 mps</td>
<td>Velocity of consumers</td>
</tr>
<tr>
<td>P_{threshold}</td>
<td>0.99</td>
<td>Detection threshold</td>
</tr>
<tr>
<td>v_t</td>
<td>50 mps</td>
<td>Velocity of targets</td>
</tr>
<tr>
<td>Q</td>
<td>0.01</td>
<td>Variance of Gaussian white noise of target motion</td>
</tr>
<tr>
<td>r</td>
<td>50 m</td>
<td>Maximum distance that consumers can attack</td>
</tr>
<tr>
<td>\gamma</td>
<td>1.5 m</td>
<td>Radius of destruction around attacked position</td>
</tr>
<tr>
<td>\rho</td>
<td>99%</td>
<td>Required confidence interval length of target’s position estimate</td>
</tr>
<tr>
<td>t_{kill}</td>
<td>10 units</td>
<td>Maximum number of schedules allowed to destroy targets with offensive capabilities</td>
</tr>
<tr>
<td>u_t</td>
<td>1.0 m</td>
<td>Utility for destroying a target</td>
</tr>
<tr>
<td>\tau</td>
<td>0.005</td>
<td>Tatonement factor</td>
</tr>
<tr>
<td>C</td>
<td>2 Mbps</td>
<td>Bandwidth of communication channel</td>
</tr>
<tr>
<td>M</td>
<td>1 Kb</td>
<td>Size of communication message</td>
</tr>
<tr>
<td>t_{com}</td>
<td>2 millisecond</td>
<td>Maximum time allowed for communication</td>
</tr>
<tr>
<td>\beta</td>
<td>0.01</td>
<td>Required probability that time taken for communication is greater than t_{com}</td>
</tr>
<tr>
<td>\alpha</td>
<td>1 pJ/bit/m^2</td>
<td>Energy required to send messages per unit distance per unit message size</td>
</tr>
<tr>
<td>e_i</td>
<td>2.5 KJ</td>
<td>Energy of sensor batteries</td>
</tr>
</tbody>
</table>

3.6 Summary of Results

This section presents the results of a CCA protocol algorithm on the simulation test-bed. Comparative results of MASM with Information-Theoretic Sensor Manager are given in Chapter 10.
3.6.1 Real-time Performance

For the parameters used in the simulation, CPLEX took an average of 0.023 seconds for calculating the optimal allocation from formulated resource bids. We also evaluated the average time taken by CPLEX for different problem sizes. These problems were generated by varying the number of consumers in the simulation. The average time taken by CPLEX for different problem sizes is shown in Figure 3-5. Clearly, the winner determination is very fast with CPLEX, for the bid-distributions being generated by the sensor network. Also, since bid formulation using QoS charts is linear in the number of bids, MASM has very low computational requirements. However, the number of bids is exponential in the number of sensors in the network. Therefore, the computational requirements of MASM increase very fast with network size. For this purpose, MASM algorithms with only polynomial complexity both in network size and number of consumer bids have been developed. These algorithms are presented in Chapter 6. Chapter 6 also provides a scalability analysis of both approximate and exact approaches to MASM.
3.6.2 Resource Utilization

As explained in Section 3.3, a tatonement process was used for enforcing resource constraints. The method in which the tatonement process was set up for battery power constraints’ using current and available rates of utilization is described in Section 3.3.

Figure 3-6 shows the price variations of the first three batteries using a tatonement rate, $\tau = 0.005$. Figure 3-7 shows the energy utilization of the first sensors with tatonement rate $\tau = 0.005$. Figure 3-8 shows the fall in the energy of the first three sensors’ battery with successive schedules when tatonement process is switched off with $\tau = 0$, respectively. When tatonement is not used, the rate of sensor utilization is not controlled. Sensors batteries are exhausted long before the end of simulation. When tatonement is used, the rate of sensor utilization is controlled by the market by adjusting the prices of sensor schedules. As a result, the sensor batteries last until the last stages of
the simulation. This indicates that the tatonement process is successful in ensuring uniform usage of sensors and in keeping them functional until the end of network operation.

A similar procedure is adopted for allotting bandwidth. The supply of capacity is constant and is proportional to $t^{com}$. Calculation of the demand for bandwidth is more involved. The following procedure was adopted. During round $t+1$, let the price of the channel is $p_t$ units/sec and bandwidth consumption is $w$. Let $\eta$ be the $(1-\beta)$ one-sided upper confidence interval of the expected time taken to communicate, where $\beta$ is the MM specified allowed probability that time taken to communicate is greater than $t^{com}$. At the start of network operation, values of $\eta$ for different bandwidth utilizations $w$, are calculated using monte-carlo simulations and stored. The demand at $p_t$ is proportional to $\eta$. The price of the channel during the current round is

$$ p_{t+1} = p_t + \tau * (\eta - t^{com}) $$

The price updates for the process works as a chanced constraint on channel capacity. That is, if the channel is too congested, then prices of the channel will increase until demand for channel capacity decreases. However, it is possible that during certain schedules, the actual time taken for communication is more than the prescribed limit. Figure 3-9 and Figure 3-10 show the time taken for communication for two sample simulation runs with $\tau = 0.005$ and $\tau = 0$, respectively.
Figure 3-6: Price variation of the first three sensors with schedule number for a sample run with tatonement $\tau = 0.005$

Figure 3-7: Energy utilization for the first three sensors for a sample run with tatonement $\tau = 0.005$
Figure 3-8: Energy utilization for the first three sensors for a sample run with tatonement \( \tau = 0 \)

Figure 3-9: Time taken for communication for a sample run vs schedule number for a sample run with tatonement \( \tau = 0.005 \)
For the run in Figure 3-9, the number of time slots when time taken to communicate crossed the specified threshold is 5. This is within the 0.01% tolerance limit specified by the MM. The SM could enforce even chanced constraints very effectively using price-based resource rationing. For the run in Figure 3-10, the number of time slots where time taken to communicate crossed the specified threshold is 124. The time taken to communicate drops to zero after the 250th schedule, since all the sensor batteries are exhausted by this time.

3.6.3 Task Deadlines

As explained in Section 3.2, deadlines on tasks can be implemented by the consumers by adjusting their bidding prices. To test the efficiency of relying on bidding
strategies for imposing deadlines, we ran simulation experiments where Type 2 targets are also present.

When targets of Type 2 are present, consumers have a strict deadline to destroy them after initiating target tracks within $t_{\text{kill}}$ schedules. We conducted experiments where $n_{t_2}$ targets are of Type 2. Consumer policy is to increase track bids by a factor $k$, if the detected targets are of type 2. Optimal $k$ values can be calculated by using market reports provided by SM. This simulation used a constant value of 3. Consumers recorded an average track time of 7.1 schedules for $T_2$ versus an overall average of 15.3 schedules, indicating the efficiency of markets in enforcing task deadlines.

3.7 Effects of Strategic Behavior

The current model assumes a cooperative environment, where bidders truthfully reveal their preferences. However, this assumption may not be accurate for some real-world environments, and strategic bidding efforts by consumers could adversely affect the performance of the system. Analyzing the effects of strategic bidding is difficult because Bayes-Nash equilibrium is difficult to calculate except for the simplest of markets. A preliminary analysis of effect of the strategic bidding can be conducted by formulating some simple strategic bidding formulations. For example, Walsh et al. [80] have analyzed the effects of strategic bidding on a combinatorial auction based supply chain formation algorithm. It is important to note the difficulty of analyzing the effects of strategic bidding actually undermines the benefits of lying about true utilities for consumers. An added advantage with MASM is that there is disengagement between the
users and the actual sensor network. Though the consumers use the sensor network, the various network parameters including the prices of individual resources (which might indicate network bottlenecks) and the actual network parameters, like position of sensors etc is invisible to the individual user. Hence, the threat presented by malicious entities that have clandestinely gained access to use the sensor network is minimal.

However, it is possible that sensor networks could have consumers in a non-cooperative environment, where each agent has an interest only in maximizing its own utility. Such scenarios require an incentive compatible auction methodology, to make truth revelation the dominant strategy, to make the allocations optimal. For example, a payment mechanism based on General Vickrey Auctions (GVA) [73] might be used to make truth revelation a weakly dominant strategy. GVA involves computation of $n+1$ winner determination problems for every combinatorial auction to calculate the agent payments. However, GVA auctions are not necessarily incentive compatible when using only approximate techniques for winner determination [81]. In addition to the computational complexity, the unique pricing mechanism used by CCA protocol precludes direct adaptation of GVA mechanisms. Further research is required for developing a real-time compatible incentive compatible payment methodology for CCA.

To analyze the effect of strategic bidding, it can be assumed that agents play Bayes-Nash strategies [82]. However, calculation of Bayes-Nash equilibria is difficult, except for the simplest of markets. An easier method for analyzing market behavior is to devise a reasonable strategic bidding policy for consumers and study resulting market behavior.
A simple strategic bidding policy for MASM consumers is to overstate their task utilities. To understand the logic behind this policy, the pricing policy of CCA should be considered. If a MASM consumer bids a price $p$ for a particular task, the prices it has to pay for resources allocated to its bid is not directly based on $p$. Instead, for any resource bundle allocated to the task during resource allocation, the consumer usually pays only the sum of the prices of the resources comprising the resource bundle (see round termination step in CCA). Therefore, a consumer that overstates his utility has the advantage of getting preferential treatment during resource allocation, while not having to pay any additional value for resources as compared to honest consumers. To analyze the impact of strategic bidding, experiments were conducted where a certain number of agents overstated their utility by a factor, $k$. The global performance of the market-based resource allocation is reflected by the number of targets successfully destroyed by the consumers during the simulation experiment. An individual consumer’s performance is measured by its surplus defined as the difference between the total utility it obtained from destroying the targets and the total price it paid to SM for buying resources during the simulation. Since the utility for destroying a target was initialized to one (see Section 3.6), a consumer gets a utility of $n$, for destroying $n$ targets.
Table 3-3: Market Performance with Strategic Agent Behavior

<table>
<thead>
<tr>
<th></th>
<th>Number of strategic agents = 0</th>
<th>Number of strategic agents = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Performance</td>
<td>8.3</td>
<td>6.1</td>
</tr>
<tr>
<td>Honest Consumer Surplus</td>
<td>1.22</td>
<td>-0.012</td>
</tr>
<tr>
<td>Honest Consumer Surplus - (CCA – SS)</td>
<td>0.438</td>
<td>-0.012</td>
</tr>
<tr>
<td>Strategic Consumer Surplus</td>
<td>NA</td>
<td>2.3</td>
</tr>
<tr>
<td>Strategic Consumer Surplus- (CCA-SS)</td>
<td>NA</td>
<td>-1.83</td>
</tr>
</tbody>
</table>

Two sets of experiments were conducted. In the first set of experiments, all the consumers bid honestly. In the second set of experiments, two out of the five consumers overstated their utilities by a factor of two. The average overall performance and average surplus achieved by the honest and strategic agents are shown in Table 3-3. When two agents indulged in strategic bidding, overall performance deteriorated. The average number of targets destroyed decreased from 8.3 to 6.1. As expected, strategic agents benefited from overstating their bid prices and their average surplus increased from 1.22 to 2.3. The average surplus of the honest agents has decreased from 1.22 to -0.012 as a result of strategic bidding.

As shown by the preliminary analysis, CCA protocol encourages strategic behavior in a non-cooperative environment. However, the benefits of strategic bidding can be mitigated by using surplus sharing mechanisms where the SM charges the consumers a fixed percentage of their surplus on each task bid. For example, a variant of CCA, CCA-SS (Combination Auction Algorithm with Surplus Sharing) has been implemented where consumers are levied an additional charge of fifty percent of their
expected surplus as calculated from their task bids. Under this mechanism, the overall surplus to the strategic agent decreased to -1.83. That is, they fare worse than the honest consumers.

Though surplus sharing provides an effective mechanism that works as a disincentive against overstating task utilities, detailed experimentation is required to analyze the complete implications of strategic behavior for some particular environment. For example, in the current simulation environment, assume that the Bayes-Nash equilibrium would constitute the consumers overstating their utilities by an inflation factor, $k'$ (since all consumers are similar). Quasi-Equilibria defined as the value of $k'$ such that if all consumers overstate their utilities by a factor $k'$, no consumer can improve its surplus by more than some threshold percentage $I > 0$ by using any factor $k''$ not equal to $k'$ [80], can be evaluated by using simulation experiments. Various surplus sharing mechanisms can be attempted using different values of $k'$ to minimize the deviations from honest behavior for that particular environment.
Chapter 4

Real-Time Winner Determination in Combinatorial auctions

An auction is “a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants” [83]. Market-oriented programming uses market mechanisms like auctions to build computational economies to solve distributed resource allocation problems. In standard auction mechanisms, individual goods are auctioned independently of each other. In a scenario where agent utilities for various goods show strong complementarities, these mechanisms can lead to inefficient allocations. Combinatorial auctions (CAs), where agents bid on a combination of items are useful in environments where agents exhibit strong complementarities. For example, consider resource allocation in a sensor network, where the available resources must be allocated to different target tracks. To maintain a target track, not only should measurements be scheduled over the various sensors, but bandwidth for measurement communication to the fusion system should also be reserved. Scheduling measurements may not be of any use to the consumers unless bandwidth to communicate the measurements is also available. Combinatorial auctions can be used in such environments, to allow the bidders to express synergistic relationship between goods. Combinatorial auctions have been used for resource allocation in a number of domains with strict real-time constraints, like Sensor Networks [84, 85], Supply Chain Management [80] and Computer Grids [86]. However, two issues acted as stumbling
blocks to the adaptation of CA in time constrained environments. They are 1) the computational complexity of eliciting valuations for all possible combination of resources to different tasks/users, and 2.) the computational complexity of finding the optimal allocation in the combinatorial auction, aka winner determination problem, which is NP-hard [100]. This chapter focuses on algorithms to improve the performance in step 2. Chapter 6 describes an approach to adapt the proposed algorithms so that step 1, i.e., explicit formulation of utilities for all possible combination of resources, can be avoided.

Recent years have seen great strides in the performance of winner determination algorithms for combinatorial auctions [64, 77, 87]. Depending on the distribution from which the bids are generated, either CABOB or Integer Programming approaches using standard commercial software packages like CPLEX [77] have proven to provide the best real-time performance. For many realistic distributions, CPLEX and CABOB perform extremely well with average running times of less than a second for combinatorial auctions involving thousands of bids. However, for some complex bid distributions, both CPLEX’s, and CABOB’s real time performances may not be adequate for environments with strict real-time constraints. For example, CPLEX and our implementation of CABOB required more than 10,000 seconds running time for a problem size of 1000 bids for certain bid distributions. Casanova, a local stochastic search procedure, is an approximate winner determination algorithm, based on Novelty+[88, 89]. Casanova provided good quality solutions, very quickly even for fairly large problem instances for a variety of bid distributions. However, in our experiments, we found that Casanova’s performance falls significantly with increase in problem sizes.
Another shortcoming of current auction algorithms involves the bid formulation process. Standard winner determination algorithms require bids on sets of resources as input. However, bid-formulation can be computationally expensive. For example, if there are \( n \) tasks and \( m \) goods, \( n \times 2^m \) bids may have to be formulated for computing the optimal allocation. This is clearly infeasible, when dealing with large allocation problems.

To overcome the above mentioned shortcomings, we formulated a novel genetic algorithm based search procedure, SGA (Seeded Genetic algorithm), for approximate winner determination in combinatorial auctions.

### 4.1 SGA Description

This section provides an explanation of the winner determination algorithm and provides a brief overview of some previous algorithms, before discussing SGA in detail.

#### 4.1.1 Winner Determination for Resource Allocation using CAs

Given that the auctioneer has a set of \( m \) goods \( G = \{g_1, \ldots, g_m\} \) to sell, and a set of \( n \) bids \( B = \{b_1, \ldots, b_n\} \). Bidders offer bids of the form \( B_i = <b_i, p_i> \), where \( b_i \) is the bundle of goods and \( p_i \) is the price the bidder is willing to pay. The winner determination problem is to find an allocation of goods that maximizes the overall utility, given the constraint that each good can be sold to no more than one task (refer to Section 2.1.1.5.1 for a formal definition of the problem). This problem can be solved as an integer programming (IP) problem using standard IP software like CPLEX [77]. Another example of exact winner determination procedure is CABOB, which uses a specialized depth-first search
that branches on bids. CABOB considers each bid in the set of bids $B$ as the vertex of a graph with edges between vertices only when they have no items in common. CABOB then uses a depth-first search (DFS) to find the optimal solution. By selecting an initial bid as being IN a possible allocation, then remaining bids with items that do not overlap this bid can also be IN the allocation. Bids that have items that overlap must be OUT of this allocation. For example, suppose the set of bids (without prices) is $B= \{\{1, 3\}, \{2, 3\}, \{2\}, \{1\}\}$ (Figure 1.) Then if the algorithm accepts $\{1,3\}$ as IN, it will consider $\{2,3\}$ as OUT, but $\{2\}$ as IN, and $\{1\}$ as OUT. CABOB bounds the DFS by pruning paths already dominated. It does this in two ways. First, it keeps track of the current tree structure. Second, at each node it runs a linear program approximation of the winner determination problem to find a feasible, not necessarily optimal, solution that can be used as a lower bound.

The approximate winner determination search procedure, Casanova is based on a local stochastic search. Casanova starts with an empty allocation and bids are chosen from the unsatisfied bids randomly and added to the allocation vector. The probability that a bid is chosen is determined using its normalized score, the bid price divided by the number of goods consumed by the bid, and the bid’s age defined as the number of steps since its last assignment to the allocation vector

4.1.2 SGA

Solving the winner determination problem for a combinatorial auction is exponential in the number of bids in the worst case. Genetic algorithms are stochastic and
polynomial in the number of bids, rather than exponential. Thus, if a genetic algorithm can reduce the overall bid search space, it is useful for:

a. approximate real-time winner determination
b. preprocessing and reducing the winner-determination search when the search space is large.

Initially, a binary string was used to represent the genetic algorithm chromosome. In this representation, a solution is coded as a binary string of length, equal to the number of bids in the problem instance. A binary string is decoded into an allocation vector as follows: a bid is labeled winning, if the corresponding string bit is one and as losing, otherwise. However, this schema produces infeasible solutions where more than one bid containing a common item are marked as winning. The genetic algorithm has poor convergence, in spite of the use of penalty functions to penalize infeasible allocations. To overcome this problem, we devised a novel representational schema based on a ranking scheme that produces only feasible solutions that is described below.

4.1.3 Representational Schema

In the ranking-based representation schema, a chromosome is represented by a string of numbers, where each number corresponds to the rank of the corresponding bid. The solution represented by a chromosome is decoded in the following manner:

1. Start with a solution with the lowest ranking bid.
2. Add the bid with the next lowest rank to the solution if it does not create an infeasible solution; otherwise, proceed to the next lowest ranking bid.
3. Continue till no bid can be added without creating an infeasible solution.

For example, consider the following scenario with 4 bids and 4 items (Table 4-1). Decoding the allocation vector associated with the chromosome “2-1-3-4” is done as follows. The initial allocation vector is empty. Since, the lowest ranking bid is bid 2 (with rank 1), bid 2 is added to the allocation vector. The next lowest ranking bid is 1 (rank 2). However, since bid 1 contains an item which is also present in bid 2, and bid 2 has already been labeled as winning, bid 1 is labeled as losing. The procedure is repeated till the status of the highest ranking bid (bid 4 in this case) is determined. Thus, the above chromosome corresponds to a solution where bids 2 and 4 are marked as winning and the rest are marked as “losing”.

Table 4-1: Example of SGA Chromosome

<table>
<thead>
<tr>
<th>Bid No.</th>
<th>Rank</th>
<th>Items</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>2</td>
<td>{a}</td>
<td>Lose</td>
</tr>
<tr>
<td>2.</td>
<td>1</td>
<td>{a,b}</td>
<td>Win</td>
</tr>
<tr>
<td>3.</td>
<td>3</td>
<td>{b,c}</td>
<td>Lose</td>
</tr>
<tr>
<td>4.</td>
<td>4</td>
<td>{c,d}</td>
<td>Win</td>
</tr>
</tbody>
</table>

4.1.4 SGA Operators

We used the standard crossover, mutation and tournament selection operators in the genetic algorithm. Crossover operator produces two offspring chromosomes by
swapping randomly chosen crossover segments between two parent chromosomes. The mutation operator changes the rank of a randomly chosen bid to a random value. The tournament selection operator replaces all the chromosomes in a given group with equal number of replicas of the chromosome with the highest fitness from among the group.

The starting population used in SGA, consists of strings in which the integers, 1 to n (where n is the number of bids) are randomly allocated to the different bids. To prevent degeneracy in the chromosome population, we added a novel operator called replenishment (Table 4-2). To understand the requirement for this operator, consider the crossover operation between the following two chromosomes of length four:

a. chromosome A: 2---1---3---4
b. chromosome B: 3---2---4---1

If the crossover segment is of length two and the crossover point is chosen to be two, the offspring chromosomes produced from the crossover operation are

c. chromosome C\(_1\): 2--2--4--4

d. chromosome C\(_2\): 3--1--3--1

As shown, mutation and crossover operators lead to repetition of ranks in the chromosomes. These chromosomes can be decoded using heuristics. For example, assign a unique number bid\(ID\), to each bid.

If two bids have the same rank, add the bid with the lower bid\(ID\) to the allocation vector. However, as the genetic algorithm evolves and the number of generations
increases, the degeneracy in the chromosomes increases. The genetic algorithm might then prematurely converge, since the chromosomes lose the strength to produce high-quality solutions with limited variability in the bid ranks. To avoid this problem, we devised the *regenerator operator*. The regenerator takes a *degenerate* chromosome and produces a *healthy* chromosome, in which no rank is repeated. The algorithm used by the regenerator operator is as follows:

Table 4-2: Regenerator Algorithm for SGA

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>i = 0; r = 1;</td>
</tr>
<tr>
<td>b.</td>
<td>i = i + 1;</td>
</tr>
<tr>
<td>c.</td>
<td>B = set of bids with rank == i;</td>
</tr>
<tr>
<td>d.</td>
<td>While (B is not empty)</td>
</tr>
<tr>
<td>d.1</td>
<td>b = bid with lowest <em>bidID</em> from B</td>
</tr>
<tr>
<td>d.2</td>
<td>rank of b = r;</td>
</tr>
<tr>
<td>d.3</td>
<td>remove b from B;</td>
</tr>
<tr>
<td>d.4</td>
<td>r = r + 1;</td>
</tr>
<tr>
<td>e.</td>
<td>if (r &lt; no of bids) repeat steps b to e.</td>
</tr>
</tbody>
</table>

The computational complexity of the regenerator operator is linear in the length of the chromosome. The use of the regenerator operator has an additional and very important advantage. The regenerator operator ensures that, in the chromosomes, each bid has a unique rank and these ranks are distributed between one and the number of bids in the problem instance. This makes the computational complexity of chromosome
sorting, for the purpose of decoding the allocation associated with it, linear in the length of the chromosome. This greatly increases the speed of the genetic algorithm.

4.1.5 Seeding the GA

If an initial high quality solution is available, the solution can be used to seed the genetic algorithm by initializing some of the initial random population to the high quality solution. A seed chromosome, seed, is created from an initial solution as follows

a. \( n = \text{size of initial solution} \)

b. for each bid \( b \)

   if(\( b \) belongs to the initial solution)
   
   \[ \text{seed}[b] = \text{random number between } n \text{ and } (\text{no of bids} + 1) \]
   
   else
   
   \[ \text{rank}[b] = \text{random number between } 1 \text{ and } (n + 1) \]

   c. \( \text{Regenerate} (\text{rank}) \)

If the GA is seeded with a high quality initial solution, it converges very rapidly, often improving on the initial seed. Seeding has two utilities:

a. If a faster, but less efficient approximate winner determination algorithm is available, the solution provided by this algorithm can be used as a seed for the GA, enabling it to converge more rapidly.

b. Seeding can be used for approximate incremental winner determination. Assume that the auctioneer has calculated an optimal (or a high quality solution) for the bids he has received. If certain bids are retracted or new bids are added (as can happen in
the SM scenario), the auctioneer can use the original solution as a seed to obtain high quality solutions to the new problem very quickly.

Linear programming can obtain a lower bound for the winner determination problem [8]. For certain bid distributions, we have noticed that this lower bound gives a very high-quality solution. For example, for CAT distributions [10] and Sandholm’s weighted random distribution [8], the linear programming solution is usually close in value to the optimal allocation.

4.1.6 Avoiding Explicit Bid Formulation

Consider a resource allocation problem with m goods $G = \{g_1 \ldots g_m\}$ and n tasks $\{t_1 \ldots t_n\}$. Let $\Theta = \{\theta_1, \theta_2 \ldots \}$, be the power-set of G. To use standard combinatorial auction-based resource allocation, bids of the form $b_{ij} < \theta_i \rightarrow t_j, u_{ij} >$ where $u_{ij}$ is the utility of allocating resource set $\theta_i$ to task $t_j$ should be formulated as a preprocessing step. This is infeasible when the number of items to be allocated is large and the real-time constraints are strict. The representation schema used by SGA can be modified to avoid explicit bid formulations. The representation schema used by SGA, for these problems, is as follows: The chromosome is represented by a string of length m. Each chromosome member is a number between 1 and n. For example, consider a three-task, four-item example. The chromosome “2-1-3-2” indicates that the first and fourth items are allocated to Task II, the second is allocated to Task III and the third item is allocated to Task IV. The fitness of the chromosome A is evaluated as follows. For each task $j$, let $S$ be the set of items allocated to it by A. To calculate fitness of A, a domain-specific utility
function, $\varphi_T(S)$, which calculates the utility of allocating resource set, $S$, to task $T$ needs to be formulated. In a standard winner determination problem, $\varphi_T(S)$ reduces to the price of the bid on resource set, $S$, for task, $T$. The fitness of $A$ is evaluated as the sum of the utilities of the allocations to different tasks, as per $A$. If the size of the population used in the SGA is $\lambda_1$ and the number of generations that SGA is run for is $\lambda_2$, the total number of utility computations required for this representation schema is $\lambda_1 \times \lambda_2 \times m$. For the standard formulation, $n^2m$ utility computations are required. Approximate techniques like function-estimation neural-networks can be used to evaluate $\varphi_T(S)$ in polynomial runtime, when exact formulation is computationally expensive. For illustration of how machine learning algorithms have been used to estimate $\varphi_T(S)$ using domain-specific knowledge in sensor networks, refer to Chapter 6.

### 4.2 Results

Experiments were performed to test the SGA on various bid-distributions, and to validate the claim that SGA performs better than Casanova. For various bid distributions, like Sandholm’s weighted random and the CAT distributions, linear programming gives a very good initial starting solution. Results show that SGA performs extremely well on these distributions, when the linear programming solution is used as a seed. However, for these distributions, CABOB and CPLEX, which find exact optimal solutions, are also extremely fast and using approximate algorithms is not required in these situations. Therefore, we do-not present our experiment results on these distributions. For the
experiments, we selected two distributions from Sandholm’s bid distributions [87] that proved to be the toughest for the exact winner determination algorithms, which are

a. Uniform \((n, m, \lambda)\) distribution which consists of \(n\) bids, each with \(\lambda\) items chosen without replacement from the \(m\) items. Price is chosen randomly from a uniform distribution on \([0, 1]\). For our experiments, we used \(m = n/10\) & \(\lambda = 5\).

b. Bounded \((n, m, \lambda^1, \lambda^2)\) distribution which consists of \(n\) bids. The number of items \(\lambda\) is randomly chosen between a lower bound \(\lambda^1\) and an upper bound \(\lambda^2\). The price is chosen from a uniform distribution on \([0, \lambda]\). These experiments, used \(m = n/10\) and \(\lambda^1 = 1\) and \(\lambda^2 = 5\).

These distributions were the most difficult for both CABOB and CPLEX. For example, CPLEX took an average of 4300 (757) seconds on a 2.8 GHz P-IV processor for a problem size of 900 bids for uniform (bounded) distributions. For larger problems, the run time was much higher. For a sample run, a problem size of 1000 bids with uniform distribution took more than 13 CPU hours. Implementation of CABOB was slower than CPLEX for the above distributions. This is consistent with the reported results for CABOB and CPLEX. The parameters used for SGA are shown in Table 4-3. All the experiments used the same parameters, except the population size. For problem sizes with less than 1400 bids, experiments used a population size of 6000 chromosomes and for larger problems, experiments used a population size of 6500. For Casanova, experiments used the parameters suggested by the authors in the original paper. However, we found that changing the walk- probability to 0.4 gives better results, and hence, wp =
0.4 was used. We tested both Casanova and SGA on large problem sizes, varying the number of bids from 1000 to 2000 bids. We used a constant \#bids/\#items ratio of 10.

<table>
<thead>
<tr>
<th>Table 4-3: SGA Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size ( &gt; 1400 bids)</td>
</tr>
<tr>
<td>Population size ( &lt;= 1400 bids)</td>
</tr>
<tr>
<td>Tour size</td>
</tr>
<tr>
<td>Crossover Probability</td>
</tr>
<tr>
<td>Mutation Probability</td>
</tr>
</tbody>
</table>

Large problem sizes created a difficulty in analyzing the absolute performance of the individual algorithms since calculating the optimal allocation using the exact algorithms took considerable time. A statistical approach was adopted to estimate the quality of the solutions generated by SGA and Casanova. It was postulated that the average optimality for a particular distribution, for a given problem size (fixed number of bids and items) is directly proportional to the number of items in the auction. This hypothesis was tested for problem sizes of 500 to 800 bids for both the distributions, using the following approach. For each problem size, 20 random problems were generated and the average value of the optimal solution was calculated. The R-square of the least square regression line with the problem size (number of bids) as the estimator and the average optimality as the regressor was greater than 0.99 for both the distributions. Hence, the linear relation between the estimator and the regressor is fairly strong. Figure 4-1 (Figure 4-2) shows the regression line and the optimality values of all
the points generated for the uniform (bounded) distribution. No particular trend was noticeable in the variation of the optimality from the mean optimality based on problem size. For calculating the average optimality of a given problem size we extrapolated the regression lines in Figure 4-1 (Figure 4-2). This extrapolation is reasonable, since no increasing trend appeared in the variation of the optimality from the mean optimality with increases in problem sizes.

![Figure 4-1: Regression line for average optimality versus problem size for uniform distribution](image1)

![Figure 4-2: Regression line for average optimality versus problem size for bounded distribution](image2)
Figure 4-3 (Figure 4-4) gives the average estimated optimality, obtained using SGA and Casanova on uniform (bounded) bid distributions. For generating these results, SGA and Casanova were run on 50 different problems, for each problem size. The average optimality of the problems ($\varphi_{\text{size}}$) is calculated by extrapolation using Figure 4-1 (Figure 4-2). For calculating the percentage optimality yielded by SGA and Casanova, we divide the mean outcome of the approximate algorithms for a particular problem size with $\varphi_{\text{size}}$. Since these results are not exact and depend on the robustness of the extrapolation method used, the plot also shows the 95% confidence interval of the optimal solution for each problem size.

Figure 4-3: Estimated percentage optimality (with their 95% confidence intervals) versus problem size for uniform distribution, using a cut-off time of 200 CPU-Sec on 2.8 GHz Pentium IV processor.
Figure 4-4: Estimated percentage optimality (with their 95% confidence intervals) versus problem size for bounded distribution, using a cut-off time of 200 CPU-Sec on 2.8 GHz Pentium IV processor.

The 95% confidence interval is calculated using the 95% confidence interval for $\phi_{\text{size}}$. This is calculated assuming that standard deviation of $\phi_{\text{size}}$ does not depend on problem size (which is consistent with our experiment results). Clearly, although Casanova is better for smaller problems, SGA gives better optimality for larger problem sizes. Casanova search is primarily memory-less. It discards solutions generated in the past periodically and restarts the search every few seconds. Although the extremely greedy approach used by Casanova is advantageous for smaller problems, a memory-less search proves insufficient for larger problem sizes in a solution space, which is exponential in the problem size. Since SGA and Casanova are stochastic algorithms, the scatter plot of the results obtained for a particular size for uniform (bounded) distribution are shown in
Figure 4-5 (Figure 4-6). Clearly, for a large percentage of problems, SGA gives better results than Casanova for both distributions.

Figure 4-5: Correlation of revenue obtained by SGA and Casanova for uniform distribution with problem size of 2000 bids. The line shows of ratio of 1 when comparing the revenues.

Figure 4-6: Correlation of revenue obtained by SGA and Casanova for bounded distribution with problem size of 2000 bids. The line shows of ratio of 1 when comparing the revenues.
4.3 Real-time Performance

We compared the real-time performance of SGA and Casanova for the uniform bid distribution with problem size = 2000 bids (Figure 4-7). We found similar results for the bounded distribution. The extremely greedy approach used by Casanova, results in better real-time performance, initially for Casanova. SGA starts with a randomly initialized population and evolution of good quality solutions requires a fair amount of computation. However, SGA is better than Casanova for larger time scales and converges to better optimality. The advantages of SGA and Casanova can be combined by using Casanova to seed SGA. Casanova, made to run for sufficient time, reaches a plateau in its performance. The high quality solutions provided by Casanova can then be used to seed SGA. Figure 4-8 shows the comparative real time performance of Casanova and SGA, when Casanova is used to seed SGA (at t = 70 CPU-Secs). The combined algorithm has better (or at least equally good) real-time performance throughout and also converges to better optimality.

Figure 4-7: Real-time performance of SGA and Casanova on a 2.8GHz Pentium-IV processor (averaged over 20 runs)
It is interesting that SGA as a stand-alone algorithm has better overall convergence than SGA seeded with Casanova. This happens because the high quality solutions provided by Casanova lead to premature convergence in SGA.
Chapter 5

Agent Learning for Task Prioritization in Sensor Networks

In MASM, the SM accepts bids only on a set of pre-defined tasks. The consumer agent is responsible for decomposing its high level tasks or goals into a sequence of SM acceptable subtasks on which it can bid on. Furthermore, it is responsible for assigning appropriate weights or priorities to these sub-tasks, such that its overall performance is optimized. To help consumer agents formulate their bids, the SM periodically provides market reports, which contains statistics about auction outcomes in the previous schedules. Consumer agents, which are software entities that may be under human supervision, are required to analyze this information and formulate optimal bidding strategies.

For example, assume that a consumer has utility $U_t$ for destroying a target $T$. The high level task of destroying $T$ might consist of the following two sub-tasks, for which SM accepts bids: a) search for target $T$, and b) track target $T$ so that the uncertainty about its position is reduced. The consumer then has to divide its overall utility, $U_t$, into utilities for the two sub-tasks, so that it can formulate bid prices. The optimal weights of the individual sub-tasks are dependant on the system conditions. For example, it might be difficult to search for targets in some environments whereas in others, tracking tracks to high accuracy might be the bottle-neck. Also, utilities for the sub-tasks should take into consideration the competition for resources. For example, sensors that can be used for velocity estimation, like Doppler sensors, might be abundant in the network, making the track task a less competitive one. Approaches like Decision theory, Dynamic Goal
Lattices [12], and Bayesian networks [14] have had use in finding the priorities/weights of various objectives in the sensor management optimization. However, a comprehensive paradigm to take mission objectives and system state simultaneously and directly into consideration during resource allocation is not available. In this chapter, an agent-learning approach is developed that allows consumer agents to use market information to adapt dynamically to the system conditions and formulate optimal bidding strategies, so as to maximize their performance.

5.1 Requirements of Agent Learning

The agent learning approach that has been developed requires two modules:

a. Data interpretation module: In MASM, consumers can use two key resources of data; 1) The periodic market reports generated by the SM, that provide the current price trends in the market and 2) Historic consumer performance data which record the market response for previous consumer bids. This data is used by the consumer agent to create price-QoS relationships for each sub-task. The constructed relationship should be in the form of a probability distribution, \( Pr_{p,t,i}(QoS) \), where \( Pr_{p,t,i}(QoS) \) is the probability that service at level \( QoS \) is offered by SM at price, \( p \), at time, \( t \) for sub-task, \( i \). In addition to historic market data, the construction of \( Pr_{p,t,i} \) also depends on the probability distribution, \( f_t(S) \), of the states of the environment in which the agent operates, where \( f_t(S) \) is the probability that the environment is in state, \( S \), at time, \( t \). To modify the existing price-QoS mapping based on market reactions to consumer bids, consumer agents can use methods like reinforcement learning. An agent that uses reinforcement
learning learns by interacting with the environment and monitoring the results of these interactions [81]. Reinforcement learning is a trial-and-error based search procedure that attempts to discover the action that will result in the highest reward, given a system state. The agent evolves while analyzing consequences of its actions during its interactions with the environment.

b. Response formulation: The agent learning technique should have an optimization routine to determine the optimal bid parameters, based on the price-QoS mappings generated by the data-interpretation module. To formulate the optimization problem, it is assumed that the consumer agents’ overall performance can be expressed in terms of its performance of the actionable tasks. Then, based on mapping between price and quality of service generated by the data interpretation module, the overall agent’s performance can be expressed as a function of the bid prices of the consumer of the individual tasks. Let the bid price used by the consumer agent at time, t for sub-task, i be \( p_{t,i} \). Let \( p_t \) be the vector of prices used by consumer for all sub-tasks. Let \( \text{QoS} \) be the vector of QoS values for all sub-tasks at time, t, and \( \text{Pr}_{t,p}(\text{QoS}) \) be the joint distribution of the probability distribution of \( \text{QoS} \) obtained at price vector \( p_t \).

The consumer’s expected performance can be expressed as a function of \( p_t \) and \( \text{Pr}_{p_t}(\text{QoS}) \) and \( f_i \):

\[
E[\text{performance}] = \max_{p_t} h(f_i, p_t, \text{Pr}_{p_t}(\text{QoS})). \tag{5.1}
\]

subject to the constraint that the expected expenditure should not exceed consumer budget:

\[
E(\sum_{i=1}^{N} \sum_{t=1}^{T} (\text{Pr}_{p_t,i}(\text{QoS} > 0) \ p_{t,i}) < \text{consumer-budget}. \tag{5.2}
\]
Assuming the consumer tries to maximize expected performance, finding the best possible price-set can be calculated using stochastic programming techniques. However, in dynamic environments, estimating $f_t$ and $Pr_p$, $QoS$ is a difficult problem. The estimated distributions are likely to have large variation and hence, the optimized values obtained from the above equations may not be useful. Further, calculation of time-profile of prices of the consumer is not required, since the consumer optimizes Eq. 5.2 during every round of scheduling, based on new market data. Therefore, only the price vector at $t=1$ is used by consumer for participating in the market. Due to the above considerations, solving the complex optimization problem in Eq. 5.2 is not necessary. Therefore, we make the following further assumptions to simplify the calculation. We assume that the price-QoS mapping ($Pr$), the probability distribution of environment states ($f$) and the performance evaluation function ($h$) are independent of time. That is, we assume that agents are myopic that consider the current estimated values of the probability distributions to be true for all future schedules. Furthermore, we assume that the market is a fixed price market. Instead of constructing a probability distribution, consumers construct the price-to-QoS mapping as a deterministic relationship, independent of time. Under these assumptions, consumer agents use the same bid price, whenever they bid for resources for i-th subtask. The optimization problem reduces to finding a single vector of consumer budgets for the sub-tasks, where the consumer budget for sub-task i is defined as consumer’s agent overall expenditure on i-th subtask for all schedules. The equation for optimization reduces to the form
\[
\max E[\text{performance}] = \max_p h (f, \gamma, b),
\]
with the budgetary constraint
\[
\sum_{i=1}^{n} b_i < \text{consumer-budget},
\]
where \( b_i \) is the consumer budget sub-task, \( i, f \) is the probability distribution of the environment states and \( \gamma \) is the deterministic mapping between price-to-QOS. Once the overall consume budget for each sub-task is calculated, consumer’s bid-price for i-th subtask can be calculated as
\[
p_i = b_i / T \cdot p(A_i/f),
\]
where \( p(A_i/f) \) is the probability that the consumer can choose to perform i-th subtask in a particular round.

### 5.2 Implementation Details

#### 5.2.1 Market Reports

In the current simulation, SM generates market reports once, every ten rounds. The market report contains the mean, the upper and lower quantiles of bid prices for both search and track tasks and the corresponding Quality-of-Service (QoS) offered to the consumers. The QoS at a particular price is the length of the ordinate axis on the QoS maps covered by the resource bundle allocated to the consumer bid at that particular price (see Section 3.2 for details).
5.2.2 Agent Learning

As explained in the introduction, consumer agent learning involves constructing a price to QoS mapping for each sub task based on market conditions. This simulation implements a simple technique where each consumer agent use a market report provided by the SM to construct an approximate relationship between QoS and bid price for each task. A linear interpolation of values, provided in the market report, obtains an approximate QoS-Price mapping. For example, Figure 5-1 shows the price to QoS mapping, constructed by a consumer for the search task, during a simulation experiment. Based on the QoS-Price mapping, the consumers calculate the optimal decomposition of the task budget into a budget for the sub-tasks so that their expected performance is maximized.

![Figure 5-1: Approximate price-QoS mapping generated by consumer for search task, during a simulation experiment](image)

This study’s experiments used the simplified version of optimization problem as stated in Eq. 5.3 and Eq. 5.4, obtained under myopic assumptions. The optimization problem faced by the consumers can be formalized as follows. The task of the consumers is destroying targets that involve the subtasks of searching for targets (t_search) and tracking
targets (t\textsubscript{tracking}). Performance measure of the consumers depends on the altruistic nature of the simulation environment. In a non-cooperative environment, the agents might attempt to maximize their expected surplus (see section 3-7 for details). In the current simulation, the consumers use the expected number of targets successfully destroyed by them during the simulation as the performance measure. Let \(\gamma\text{\textsubscript{search}}(p)\) (\(\gamma\text{\textsubscript{track}}(p)\)) be the approximate QoS obtained at price \(p\) for search (track) task. Let \(\delta\text{\textsubscript{search}}(q)\) (\(\delta\text{\textsubscript{track}}(q)\)) be the average time associated with completing the search (track) task when service at QoS \(q\) is available during each round of scheduling. A search task is complete when a new target is found. A tracking target is complete, when the target track uncertainty achieves the required threshold. The best bidding prices, so that the expected time for completing \(T\) is minimized is

\[
\text{performance} = \min_{p_1, p_2} \delta\text{\textsubscript{search}}(\gamma\text{\textsubscript{search}}(p_1)) + \delta\text{\textsubscript{track}}(\gamma\text{\textsubscript{track}}(p_2))
\]

subject to the constraint

\[
\delta\text{\textsubscript{search}}(\gamma\text{\textsubscript{search}}(p_1)) \ast p_1 + \delta\text{\textsubscript{track}}(\gamma\text{\textsubscript{track}}(p_2)) \ast p_2 \leq U_t.
\]

This optimization problem was solved using exhaustive enumeration techniques. The value of \(p_1\) is varied between \([0, U_t]\) and the corresponding value of \(p_2\) (the highest value of \(p_2\) meeting the budgetary constraint) is calculated. The values of \(p_1\) and \(p_2\) which give the highest performance are established.

Directly using the values obtained by maximizing overall performance leads to extensive reliance on the sparse market data. For example, a particular consumer agent that is under attack from a target with offensive capabilities, might bid aggressively for
tracking resources. As a result, QoS-Price mappings for the track subtask might show a temporary shift. If all the consumer agents adapt rapidly to the new auction outcomes, they cause a permanent price increase in the market.

To avoid speculative behavior, consumers use Widrow-Hoff learning to buffer their responses, similar to Cliff and Bruten’s ZIP traders [91]. ZIP traders were originally designed to extend the Zero intelligence agent based simulations of Gode and Sunder [92]. The Widrow-Hoff learning rule with momentum [93] is used by the consumers to adapt their bid prices, based on their current bid prices and the calculated optimal prices. If the current search budget is $\beta_{\text{current}}$ and the optimal search budget is calculated to be $\beta_{\text{optimal}}$ using the optimization routine explained above, the search budget of the consumer is updated to $\beta_{\text{updated}}$ as follows:

$$
e = (\beta_{\text{optimal}} - \beta_{\text{current}})$$

$$
\tau_{\text{updated}} = \gamma \ast \tau_{\text{current}} + (1- \gamma) \ast e
$$

$$
\beta_{\text{updated}} = \beta_{\text{current}} + \lambda \ast \tau_{\text{updated}}
$$

The learning rate parameter, $\lambda$, determines the rate at which the budget is changed. During each iteration, the search-budget is updated using the current momentum, $\tau_{\text{current}}$, which is a weighted sum of the momentum during the previous iteration and the current error. Current error is defined as the difference between the current search-budget and optimal search-budgets. The momentum rate parameter, $\gamma$, determines the weight given to the past changes in the calculation of momentum.
5.3 Results

To determine the optimal search to track task budget ratios, simulation experimentations were conducted, varying the percentage of the total budget that is used for search task. To increase the speed of the simulation experiments, only the first five sensors were used. Figure 5-2 shows the average number of targets killed for different values of the percentage of the search budget in the overall task budget. It is clear that the optimal search budget percentage is around 0.05.

![Graph showing the average number of targets killed versus search budget](image)

Figure 5-2: Number of targets destroyed versus search budget (averaged over 10 simulation experiments)

To determine the efficiency of the learning algorithm, the search budget of the five consumers in the simulation were initialized to different values. The learning algorithm was allowed to modify the search budget, after each market report is available. Since the overall utility of target destruction in the simulation is equal to one, search budgets were initialized to the following values.
search-budget of i-th consumer = 0.1*i;

track-budget of i-th consumer = 1 – 0.1*i;

Figure 5-3 shows the pattern in which the learning algorithm of each consumer adapts the search-budget as the simulation progresses, for a sample run. Search budgets of the consumers converge to the optimal value for all consumers within 350 rounds of scheduling. Thus, the learning technique is very efficient in finding the optimal search-to-track budget ratios.

Figure 5-3: Convergence of search budget to optimal value, based on Widrow-Hoff learning

In spite of its simplicity, the learning approach proposed in this chapter has some important advantages, compared to conventional static approaches like goal lattices. The learning algorithm requires the agent to analyze market data and compute the optimal weights (prices) for subtasks, based on market conditions and state of the environment in
which the agent operates, during each scheduling. As a result, the sub-task weights adapt dynamically to changed system conditions. For example, assume that the target density in the environment decreases suddenly, due to some targets leaving the simulation area. The search task becomes more difficult and the optimization routine automatically increases the search-budget. On the other hand, hard-coded sub-task weights do not respond adequately to the new situation.

The agent learning technique’s optimal performance results from the assumptions used in the optimization routine, namely, that the consumer performance evaluation function and the probability distribution of the environment states, are independent of time. These assumptions are trivially true, since they are inherent to the simulation. In more general situations, myopic assumptions may be invalid. For example, if it is known beforehand that the number of targets in the environment will decrease with time, then the difficulty of the search-task increases as the simulation progresses. Therefore, the consumer agent should plan to progressively increase the percentage of the search-budget. Under these conditions, it is clear that the optimization technique used in this chapter will not yield optimal values. In general, the optimization problem, used for market participation, should not employ myopic assumptions. Learning approaches based on the formulation of Eq. 5.1 are the subject of future work.
Chapter 6

Approximate Techniques for Market-based Algorithms

As explained in Section 3.1, when the number of sensors is large and real-time constraints are strict, to use E-MASM is not feasible. If the sensor network has n sensors and m consumer bids, the bid formulator module of E-MASM has to generate $m^2n$ bids. After bid formulation, the computation of the optimal allocation involves solving the integer programming problem, which is np-hard. Both these steps are computationally expensive and hence E-MASM cannot be used for resource in large sensor networks. This chapter describes approximate techniques for sensor management that have polynomial run- times both in terms, of sensors and consumers, and hence, are scalable to much larger systems than E-MASM.

6.1 A-MASM Architecture

The modified SM architecture that uses approximate algorithms is shown in Figure 6-1. This architecture incorporates two significant changes compared to E-MASM, that reduce the real time requirements associated with resource allocation. They are:

1) Neural Network- based Service Chart: In E-MASM, the bid formulator module formulates bids for actual resources from the high-level consumer bids on tasks, using the service chart database which specifies detailed domain information such as sensors’ field locations and characteristics. This formulation involves computation of how the parameters of the task are affected by allocating a particular resource bundle to the
consumer task (refer to section 3-2 for details). Depending on the computational complexity of the process model of the consumer tasks and the measurement model of the sensors, bid formulation can be computationally expensive. The computation costs associated with this step are critical since this step has to be executed once for every resource bid during bid formulation. If an exact calculation is infeasible, function approximation algorithms like neural networks can be used to accomplish this step. The neural network should be designed so as to accept the current task parameters and the allocation vector as input and produce the resultant task parameters as output. When the exact formulation is sufficiently complex, function estimation approaches can provide time savings. This is because, given a constant network size, running times of neural networks are independent of the complexity of the system being modeled.

2) **SGA-based Optimizer**: Instead of using the IP formulation for winner determination, approximate techniques like SGA (see Chapter 5 for details) can provide resource allocation. As explained in Section 4-1-4, a special formulation of SGA allows MASM to avoid the bid formulation stage, reducing the computation costs associated with resource allocation. However, this representation requires slight modification to account for pseudo-sensors (see next section for details).
6.1.1 Adapting SGA for A-MASM

An SGA schema that does not require combinatorial bids as an input to calculate an approximately optimal allocation has been described in Section 4-1-4. This schema can be used for sensor scheduling as follows. If there are n sensors and m consumer bids, then the GA chromosome is an n-length string of real-valued integers that range between 1 and m+1. The value of the k-th member in a chromosome denotes the consumer bid to which the k-th sensor is allocated by the chromosome. If the k-th member has a value of (m+1), then the k-th sensor is not used for any task in the allocation represented by the chromosome. However, pseudo-sensors (see Section 3-4) present a problem to the above
representation. For example, if there are multiple bearing-only sensors in the environment, then the above representation schema does not define how the bearings-only sensors should be combined to create pseudo-sensors. To address this issue, the following changes were made to the representation schema. The bearings-only sensors are not directly considered in the representation schema. Instead, each possible pair of bearings-only sensor that can be combined to create a pseudo-sensor is regarded as an additional sensor and assigned a sensor number. For example, in the simulation sensor network of Table 3-1, the three bearings only sensors can be combined in two different ways. Therefore, instead of using an eight member string, a seven member string is used as the chromosome. The sixth chromosome member indicates the combination of Forward Looking Infrared Electronic Support Measures (FLIR-ESM) and the seventh chromosome member indicates the combination of the Forward Looking Infrared – Infrared (FLIR-IR) sensors. However, clearly, only one combination can be used during allocation. A feasible allocation is decoded from the chromosome as follows: If both the pseudo-sensors are allocated to consumer bids, then only the pseudo-sensor with the lowest sensor number is used. If the pseudo sensor with lowest sensor number is not allocated, i.e., if its value in the chromosome is equal to m +1, then the second pseudo sensor is considered for allocation. The second pseudo sensor is used if its value is between 1 and m.
6.2 Utility-Estimation using Radial Basis Network

The bid-formulation stage in E-MASM involves computing the utility of allocating all possible resource combinations to each task. Even approximate winner determination algorithms like SGA (see chapter 5 for details) require a large number of utility calculations. Computing the utility of allocating a resource set to a particular task can be computationally expensive [94, 95]. Approximate function estimation techniques like radial basis neural networks can be used to approximately calculate utilities, when exact computation is infeasible due to real-time constraints.

6.2.1 Radial Basis Network Theory

Radial Basis Functions (RBF) are a special class of functions, whose response increases or decreases monotonically with respect to distance from a center. For example, the Guassian function (Eq. 6.1) is a standard RBF function, for which the response decreases monotonically with distance from the center:

\[
\varphi(x) = \exp\left(-\frac{(x - \mu)^2}{\sigma^2}\right). \tag{6.1}
\]

Radial Basis Networks are two-layer neural networks that use radial basis functions for function estimation. The network showed in Figure 6-2 maps the n-dimensional input vector into one dimensional output, according to Eq 6.2 [96]:

\[
f_i(x) = \lambda_0 + \sum_{i=1}^{n} \lambda_i \varphi(\|x - c_i\|), \tag{6.2}
\]
where \( x \in \mathbb{R}^n \) is the input vector; \( \varphi(.) \) is the radial basis function; \( \| . \| \) denotes the Euclidean norm; \( \lambda_i \), \( 0 \leq i \leq n_r \), are the weights associated with neurons; \( c_i \in \mathbb{R}^n \), \( 1 \leq i \leq n_r \), are the centroids of the neurons, and \( n_r \) is the number of hidden layer neurons.

Figure 6-2: Schematic representation of a radial basis network

Each neuron in the hidden layer computes the distance of the input vector from its center. It outputs the value obtained by applying the radial basis function to the distance. The final RBF output is formed by a weighted sum of the neuron outputs. If the centers and spreads of the neurons are considered fixed, finding the parameters associated with the model so that the mean square is minimized is a linear optimization problem. However, if the parameters associated with the neurons are not considered as fixed, the training
RBF is no longer linear. Various derivative-based methods like gradient descent [97], Kalman Filtering [98], Back Propagation [99] and derivative-free methods like genetic algorithms [100] have been used for the non-linear optimization involved in the neural network training.

### 6.2.2 Performance

Two separate neural networks were trained for the search and track tasks. The number of input nodes for the neural network is the number of sensors + 1. The first inputs can take a value of 0 or 1. If the i-th input is one, the i-th sensor is used for the task. The n+1 input is the current task parameter. For tracking task, the n+1 parameter is the current covariance associated with the target track. For the search task, the n+1 parameter is the current entropy of the grid being searched.

A training set of size 1000 was generated using the following approach. The first n inputs are assigned either zero or one with equal probability. The task parameter is generated from a random distribution between the highest and lowest bounds for it. The percentage task completed for each training sample is calculated using QoS charts (see Section 3-2). This value is used as the target vector for training the neural network. For testing the neural network, a testing set of a thousand samples is generated using the same techniques as that used for generating the training set. The performance of the neural network is evaluated using the test and train error. The test (train) error is the sum of the squares of the difference between the exact function value, calculated using QoS charts and the estimated value, outputed by the neural network for a given test (train) sample.
The `newrb` function of the Matlab neural network toolbox was used to generate the neural nets. The `newrb` function implements an incremental algorithm, which starts with zero hidden-layer neurons. The number of neurons is incrementally increased till the desired error rate is reached. Matlab takes the training set matrix, and the spread of the RBF functions in the neurons and desired training error as the input. The default value of 1.0 was used for the spread. As the number of neurons is increased, the training rate decreases. But, overtraining can occur as the number of neurons increases, as the network memorizes training-set patterns and loses generalizability. Experiments were conducted for various values of goal-training error (which result in different network sizes), and the corresponding test errors were noted. Figure 6-3 and Figure 6-4 show the average test error associated with a neural network with a specific training error for search and track tasks, respectively. For both neural networks, the test error shows a curvilinear relation with train error, indicating overfitting for larger network sizes. The overall performance of the function estimation algorithm is satisfactory, with a minimum test error of less than 5% for both tasks.
Figure 6-3: Performance of RBF network for search task

Figure 6-4: Performance of RBF network for track task
6.3 Performance of Approximate Methods

A-MASM was evaluated using two different attributes:

1. Real-Time Performance
2. Optimality.

Experiments were conducted using both A-MASM and E-MASM formulation for different problem sizes. If there are n sensors in the sensor network and m consumer bids, the total number of resource bids that have to be formulated for exact evaluation is $m^2$. It should be noted that only E-MASM generates this number of resource bids.

The number of resource bids generated by A-MASM equals $G \times P \times m$ where $G$ and $P$ are the number of generations and the population size used in SGA algorithm respectively. However, in this section, the number of resource bids generated by E-MASM is used as a standard measure of the problem size for a given sensor network size and number of consumers in the market. This is used throughout this section to enable direct comparison of A-MASM and E-MASM. To test the performance of A-MASM, with increasing problem sizes, the sensor network was expanded by adding two additional sensors. A radar and a doppler, were added to both sensor platforms in the simulated sensor network, increasing the total number of sensors to ten. Each consumer bid translates into $2^9$ resource bids since the three bearing sensors can be combined to form two different pseudo-sensors.

The number of resource bids in the simulation can be increased by increasing the number of consumers. For example, to generate a problem with 20,000 resource bids, the
number of consumers is increased to 40. Consumer bid prices for tasks depend on their search and track budgets (see chapter 6 for details). Heterogeneity in the consumer bids was created by using different values for the search and track budgets for each consumer. The search budget \( sB \) of a consumer in the market was selected randomly from a uniform distribution between \( [0.05 - \lambda, 0.05 + \lambda] \). A value of 0.02 was used for \( \lambda \). The track budget of the consumer was initialized to \( (1 - sB) \) where \( sB \) is the consumer’s search budget.

Since target motion and sensor measurements in the simulation are modeled with one dimension linear equations, exact utility evaluation is not computationally expensive, in the current scenario. The average computational time for evaluation of the utility of a resource-set for a consumer bid was approximately one millisecond. The same calculation, using the RBF network, took approximately six milliseconds. As a result, approximate function estimation techniques are not required in this situation. For more complex system models, neural networks will be more efficient because the complexity of their running time is independent of system complexity being modelled, as long as the number of neurons remain unchanged.

The number of resource bids generated in E-MASM equals \( n^2m \), where \( n \) is the number of sensors, and \( m \) is the number of consumers. In the current simulation, each consumer has one task bid during each round of scheduling. Allocation procedure in E-MASM consists of two parts: a) Integer problem formulation: For each consumer bid, combinatorial bids need to be formed for every resource combination possible. Once the resource bids are formulated, the allocation problem should be expressed as an integer programming problem, as shown in Section 2.1.1.5.1.
b) Winner Determination: The IP problem formulated should be solved to determine the optimal allocation. The IP problem was solved using CPLEX, a commercial software package (see Section 5-2 for details).

Figure 6-5 and Figure 6-6 show the times required by the problem formulation step and the winner determination step, with increasing number of resource-bids in the combinatorial auction. Clearly, the slowest link in E-MASM was resource-bid formulation. Solving the IP problem for the resource-bids generated in the market proved to be extremely easy task for CPLEX.

![Figure 6-5: Time Required for formulating resource-bids from the consumer task-bids](image-url)
The simulation was run with the same parameters for E-MASM. Figure 6-7 shows the total time required for deciding the allocations with increase in the number of resource bids for both A-MASM and E-MASM. The parameters used for SGA in A-MASM are given in Table 6-1. Figure 6-8 shows the average optimality achieved by A-MASM for different problem sizes. A-MASM has performed exceedingly well with over 98% optimality for all tested problem sizes. The time savings for A-MASM are over 500% for large problem sizes. Although the problem size increases exponentially with the increase in sensor network, the computational complexity associated with resource allocation using A-MASM is only polynomial. Therefore, the real time savings are greater for larger problem sizes.
Table 6-1: Parameters used for SGA in A-MASM

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>150</td>
</tr>
<tr>
<td>#Generations</td>
<td># consumers</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.02</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>0.995</td>
</tr>
<tr>
<td>Tour Size</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 6-7: Comparison of time requirements for E-MASM and A-MASM
6.4 Scalability Analysis

The scalability of MASM depends on sensor network characteristics. Assume that the maximum amount of time available for the sensor manager for resource allocation, once the consumer bids are received, is ten cpu-seconds. Furthermore, assume that the number of consumers divided by the number of sensors in the network is constant and equal to ten (#consumers/#sensors = 1/10), and that each consumer submits one and only one bid during each round of scheduling. The size of the network that A-MASM and E-MASM can handle, so that allocation takes place within the cut-off time, depends on the maximum number of sensors that can be used for each task during any particular schedule (see Figure 6-9). Spatial restrictions often mean that even if a sensor network

![Figure 6-8: Optimality of SGA for different problem sizes (averaged over 10 runs)](image-url)
has thousands of sensors, only a few at a time will be able to track a given target. Assuming that the SM allocates a maximum of five sensors per task during any schedule, E-MASM and A-MASM can handle network sizes of approximately 1800 and 4000 sensors respectively (these values are obtained by extrapolating the curves shown in Figure 1). On the other hand, if SM allocates a maximum of ten sensors per task during each round, then the scalability of E-MASM and A-MASM reduces to network sizes of approximately 60 and 130 sensors, respectively.

![Figure 6-9: Scalability of E-MASM and A-MASM](image)

As explained, this scalability analysis is based on a number of assumptions about sensor network characteristics. To verify if MASM can be applied to a particular network, the communication costs associated with sensor management, the complexity of utility calculations and the number of bids that need to be formulated during each round of allocation need to be considered. For E-MASM, the number of bids is exponential in
the maximum number of sensors that can be allocated to a task in a particular round, and linear in the number of consumers. For E-MASM, the number of bids that need evaluation depend on the SGA parameters, as explained in Chapter 4.
Chapter 7
Comparison of MASM to Information-Theoretic Sensor Manager

7.1 Information-Theoretic Sensor Manager

To compare the performance of MASM, we implemented an information theoretic sensor manager (ITSM). Information theoretic approaches are quite common in sensor management. Hint K.J and McIntrye [44] used information gain (the entropy change in environment for a given sensor allocation) as the predicate for their hierarchical sensor management architecture. An information theoretic sensor manager (ITSM) schedules sensors to minimize the entropy of the environment. The amount of information gained can be measured by the change in entropy prior to and following a sensor measurement. Entropy, \( H(x) \) of a probability density (mass) function is defined as

\[
H(x) = \sum p(x_i) \log p(x_i),
\]

for the discrete case, and

\[
H(x) = \int p(x) \log(p(x)) dx,
\]

for the continuous case, where \( p(x) \) is the probability density (mass) function for the continuous (discrete) distribution. Assuming a normal distribution of target location, the information gain, i.e., the difference between the a priori and a posteriori entropies is

\[
I = \log(\sigma_a / \sigma_b),
\]
where $\sigma_a$ is the error covariance of the distribution, and $\sigma_b$ is the error covariance before the measurement. The error covariance matrix, $P$, maintained by the Kalman filter process can be calculated offline and can be used in the calculation of the information gain, using the relation, $\sigma = \sqrt{|P|}$, where $|P|$ is the norm of matrix, $P$. ITSM calculates the information gain, associated with each possible allocation and schedules the resources as per the allocation with the highest information gain. A key issue in the use of ITSM is the priorities that need to be assigned to the various tasks. We tested different values to find the optimal priorities for the tasks in the current scenario, and arrived at the values of 0.05 for search task and 0.95 for track task.

7.2 Enforcing Resource Constraints in ITSM

To find the optimal allocation, ITSM uses an exhaustive enumeration based technique. For each possible allocation, the total entropy decrease for the search and track tasks (weighted by the respective task priorities) is calculated. Resources are allocated according to the allocation that results in maximum entropy reduction. However, no straightforward methodology exists for implementing bandwidth and battery power constraints in ITSM. The bandwidth constraint specified in the simulation environment is in the form of a soft constraint: the probability that time taken for communication exceeds a certain limit, $t^{\text{com}}$, should be less than a threshold, $\beta$ (see Section Error! Not a valid link. for details). To enforce this constraint, ITSM does not consider allocations that require bandwidth which has more than $\beta$ chance of crossing the $t^{\text{com}}$ limit. The probability that the time taken for consumption for particular bandwidth consumption


exceeds $t^{\text{com}}$ is determined using monte-carlo simulations. ITSM does not consider energy constraints in deciding the optimal allocation. The issues that arise from not modeling the energy constraints are addressed by modifying the experimental set up (see next section).

### 7.3 MASM-ITSM Comparison

Incorporating battery power constraints into ITSM is not straightforward. For evaluating ITSM, we conducted two sets of experiments. For the first set of experiments (IT-ZE), all the energy requirements of the sensor network are assumed to be zero. For the second set of experiments (IT-E), energy consumed for different tasks is the same as for MASM. In this case, if a sensor exhausts its battery, it is not considered in future allocations. ITSM-ZE and ITSM-E experiments give the upper and lower bounds for the performance of ITSM with a reasonable energy constraint enforcement policy. In the simulation, the consumer agent’s primary goal is to destroy as many as targets as possible. Therefore, the performance of a particular sensor management approach can be evaluated by calculating the average number of targets destroyed using that approach in the simulation experiments. Ten experiments were conducted with MASM, IT-ZE and IT-E sensor management approaches. The parameters used in the simulation are given in Table Error! Not a valid link. The average number of targets destroyed using each approach is shown in Figure 7-1. It is clear that MASM is more successful in meeting the consumer objectives, i.e, in destroying the targets, than ITSM. Interestingly, MASM outperforms ITSM even when energy costs are assumed to be zero when ITSM approach is used.
7.4 Interpretation of MASM’s Superior Performance

The reasons for MASM outperforming ITSM are as follows:

1. MASM allocates resources to maximize the utility of allocation to the consumers (as indicated by their bid prices). The utility of an allocation to a consumer depends on how well the allocated resource set contributes to the consumer’s goals. Therefore, resource allocation takes goal related parameters directly into consideration. On the other hand, ITSM concentrates on maximizing information content, neglecting the value of information to the goals. For example, consider the following simple scenario. A single sensor is used to track two targets $T_1$ and $T_2$ simultaneously. Target motion is simulated by the equation:

$$x_{T}(t+1) = x_{T}(t) + w_{T}$$
where \( x_T(t) \) is the target position at time \( t \) and \( w_T \) is white gaussian noise with constant covariance \( Q = 0.005 \). Targets can be attacked and destroyed if the 99% confidence interval of their position is less than \( \beta_{\text{threshold}} = 0.5 \) unit. The sensor makes one measurement during each time period, and the measurement equation is:

\[
    z(t) = x(t) + v(t),
\]

where \( x(t) \) is the state vector, and \( v(t) \) is zero mean white noise with constant variance, \( R = 0.03 \). Let the initial uncertainties in the position of \( T_1 \) and \( T_2 \), \( \beta_1 \) and \( \beta_2 \) be equal to 1 unit.

Two sensor scheduling approaches were implemented. The first approach schedules the sensor to maximize the information gain from the sensor measurement. The second approach schedules the sensor to maximize the utility of measurements, which is defined as the inverse of the total number of sensor measurements required to bring the targets to threshold uncertainty. The optimal measurement is determined using exhaustive enumeration techniques. The change in uncertainty of the two approaches is shown in Figure 7-2 and Figure 7-3. ITSM oscillates between the two targets without collecting enough information on any one target long enough to successfully destroy either. The above experiments give an unfair advantage to the utility-based approach since the optimization routine considers multiple sensor schedules simultaneously. In spite of this bias, these experiments offer an insight into the handicap suffered by ITSM due to its inability to take goal related parameters, like \( \beta_{\text{threshold}} \), and utility-based calculations directly into consideration. On the other hand, markets provide a principled way to take
the utility of a given allocation to high-level goals directly into consideration during scheduling.

Figure 7-2: Change in uncertainty of target tracks, while using an information-theoretic approach
2. Prioritization using prices: ITSM schedules sensors so as to optimize the information gain from the environment. The information gain from allocations to different tasks can be weighted differently, so that the sensor scheduler prioritizes between the various tasks. MASM, consumers bid using the task budgets and the task prioritization of the SM, is based on the consumer bid prices. To determine the optimal task budgets based on overall goal utility, learning techniques (see Chapter 6 for details) can be used. For the current solution, consumers performed best when the search and track budgets were initialized to 0.05 and 0.95 respectively. Although both ITSM and MASM prioritize between the various tasks in the environment, task prioritization in
MASM has some inherent advantages. This is because a price-oriented approach has the ability to reserve resources for future use by high priority tasks, even if no high priority tasks are currently in progress. For example, in the simulation, consider a situation where the first consumer is tracking a target and the rest of the consumers are in search mode. Both MASM and ITSM give highest priority to the track task because the first consumer has a high-budget for a track-bid and bids accordingly. However, during the tracking task, the prices associated with the sensing resources increases since the rate of their battery power utilization during tracking is high (refer to Section Error! Not a valid link. for a description of how prices vary with rate of utilization). After the tracking task is completed and when only detection tasks are in progress, it is possible that prices of the sensor schedules increase beyond the reach of consumer bid prices for the detection task. Consequently, sensors have use at a slower rate during the detection phase, effectively reserving sensors for future use by tasks of higher priority. However, ITSM has no method of prioritizing between two tasks, except when both the tasks are currently in progress. For example, Figure 7-4 shows the number of sensors used during different rounds of scheduling using MASM. The number of sensors used when tracking tasks are in progress is comparatively higher than the number of sensors used when only detection tasks are in progress. When only detection tasks are present, a significant percent of sensors are resting, thereby preserving their battery power for future use.
Figure 7-4: A comparison of the number of sensors used for measurements, based on whether target tracks are currently in progress or not.
Chapter 8

Conclusion

8.1 Contributions

To demonstrate and test the effectiveness of multi-agent based design for automated sense-making of data, this study addressed two different, data-rich, information-poor environments: a) Sensor management in distributed networks and b) Supply chain management.

For the sensor management domain, we used market-based agent framework that allows the system to consider complex tradeoffs involved in resource allocation. Although markets have been suggested as a valuable framework for resource allocation in sensor networks [48], a few stumbling blocks prevented the development of a market-based sensor manager. They were

a) The computational complexity of eliciting valuations for all possible combinations of resources to different tasks/users.

b) The computational complexity of the winner-determination problem.

c) The problems associated with mapping from high-level mission goals to priorities for actionable tasks.

d) The modeling of reasonable pricing mechanisms for non-commensurate resources that exist in the market.

The techniques described in this thesis, including approximate techniques for handling computational complexity (MASM-A), pricing mechanisms for enforcing resource constraints and learning techniques to adapt to system dynamics address these
issues adequately and have shown their efficiency in the experiments conducted in the simulation environment.

For the supply chain management problem, I demonstrated how knowledge-centric agent-design can be used for sense-making in a dynamic, emerging scenario. Also, developed is a framework for information sharing using multi-agent team architectures like CAST. This framework has the potential to prevent agent-overloading due to information-sharing in distributed, stochastic supply chains.

8.2 Future Work

My future work plans include extending both the market-based sensor management and the supply chain management investigation.

For the market-based sensor management:

1. Multi-platform sensor management: This research initially focused on the sensor management requirements of a single platform with multiple heterogeneous assets (Figure 8-1). I plan to extend this model to multiple coordinating platforms. Current auction algorithms are centralized, but in the multi-platform sensor management (see Figure 8-1) case, bundles of sensors and intermediate processing nodes distributed across different platforms must be considered. I plan to investigate the use of multiple auctions, one per platform, which act in an interrelated fashion, somewhat similar to [101]. Thus, platform/auction A might request some resource by forwarding a bid to platform/auction B. Clearly the details of these interactions, and the time constraints involved, will need to be carefully designed and tested.
2. Improvement of agent learning algorithms: The agent learning techniques developed in this work need not converge to globally optimal sub-task weights. The convergence properties of the agent learning techniques needs to be clearly analyzed using game-theoretic techniques, particularly when multiple, heterogeneous consumers participate in the market.

3. Extension to “smart dust” scenarios: Although A-MASM has good real-time capabilities, the proposed algorithms cannot be used in “smart dust” [102] type scenarios, where the number of sensors involved is on the order of a few thousand. Markets involving a large number of sensors should be decentralized where the complete decision making, including the price of sensor schedules and choosing which tasks to undertake, takes place at node level. A market algorithm, SORA, based on node-level decision making has been proposed by Mainland et al. [103]. However, SORA does not provide cost-benefit analysis of undertaking a task, a key
advantage of using MASM. Our initial results from extending MASM to smart dust scenarios are very promising.

4. Incentive compatibility in CCA: An investigation into the effects of strategic bidding has been conducted (see Section Error! Not a valid link. for details). Incentive compatibility of CCA protocol needs clearer examination. Particularly, an analysis of the Bayes-Nash equilibria associated with the payment mechanisms used in CCA protocol would be valuable.

5. Enforcement of task deadlines: No straightforward way to enforce task deadlines using the current CCA protocol exists. Current research is focusing on designing prioritization techniques that give greater weight to tasks with earlier deadlines during resource allocation.

6. Testing with real-world sensor data: In addition, I am currently working on implementing the developed market algorithms on real-world sensor data.

For supply chain research work, a collaborative multi-chain supply chain simulation should be developed to test the effectiveness of CAST-based information sharing framework, described in the Appendix. Walsh’s supply chain model [80] is being extended to use task dependency networks to create a collaborative supply chain environment. Extensions to the model should be made to consider supply-chain operation in addition to supply chain formation issues.
Bibliography


Appendix A

A Team-Based Multi-Agent Architecture for SCM

A.1 Problem Background

Modeling and simulation tools are often used as a low cost means for exploring different strategies and for learning to manage the unexpected. However, analytical models are limited in their ability to model complex, multi-firm, multi-dimensional relationships. New simulation tools, including multi-agent systems, are starting to be investigated [13-16]. Multi-agent system design meshes well with modeling supply chain networks, as it inherently assumes that agents have their own goals, which may be anywhere from pure self-interest to cooperative, thus allowing more freedom of analysis compared to traditional simulation or analytical tools. While supply chain agent-based modeling is not yet a standard approach, Proctor & Gamble has already realized $300 million in annual savings from an under $3 million dollar investment [104]. However, research on using agent technologies to support collaborative sense-making demands more studies.

A.1.1 Multi-Agent Systems for Supply Chain Management.

Multi-agent approaches for tactical and operational management of supply chains have received some attention in recent years [13-16]. Cohen and Stathis [13] argue that
multi-agent architectures for supply chains will shift the balance of power completely to consumers, make consumers perfectly rational, consistent and tireless, create lowest price markets and necessitate build-bulk distribution infrastructure and informative marketing techniques. Fox et al. [16] have developed a generic and reusable agent building shell (ABS), a collection of software components and interfaces that provide support for application-independent agent services for implementing supply chain agents. Based on an analysis of several different supply chains, Swaminathan et al. [15] have created a library of supply chain modeling components. These components representing types of supply chain agents (vendors, manufacturers etc), the control policy of these agents (inventory policy etc) and inter-agent interaction protocols are used to model supply chains. Nissen [14] used the Agent Development Environment (ADE), a research application to develop either centralized/distributed multi-agent applications, to formulate an easy-to-use agent implementation methodology for web-based supply chains.

A.1.2 Team-based Agents

Teamwork requires flexible coordination that cannot be modeled using simple coordination plans. Multi-agent theory has been proposed as a comprehensive framework for teamwork. The theory of Shared Plans [105] is based on Pollack's mental state model of plans [106], and models teamwork using intentions to do certain steps together. Based on the theory of Joint Intentions, proposed by Cohen et al. [107] and Jennings [108], Tambe [109] has proposed a multi-agent team architecture STEAM which uses specialized communication protocols to establish team goals. In addition to the
individual operators that an individual agent’s individual activities, STEAM uses specialized operators called “Team operators” which represent a team’s joint activities. Rules that represent how and when team operators can be added or modified form the core of the team functioning. These multi-agent team architectures have been used in a wide range of agent only settings [110]. However, supply chain teams should have human in the loop capacity since some of the participating team members can be humans. To provide this capability, Collaborative Agent for Simulated Team-work (CAST) [111] focuses on reducing communication requirements by exchanging only the most important information required for coordination.

CAST uses Petri Nets to model agent mental states and also to monitor execution of team plans. Petri Nets have control nodes to represent the belief an agent has about the current goals and other agent activities and belief nodes to represent its belief about the world. Petri-Nets are generated using: a) Team structures, i.e. the roles and responsibilities of different agents in the team and b) teamwork process knowledge (e.g. individual plans, team plans. CAST uses the principles of Pro-active information exchange to minimize communication overhead. The principles of pro-active information exchange allow an agent A to communicate some information I to another agent B if and only if A is sure about I’s truth value, and A believes that I is relevant to B’s goal and that B is not aware of I. CAST uses MALLET, a LISP like language to encode knowledge about operators and plans of individuals and the team as well as definition of roles and responsibilities on the team. Role of an agent in a team is defined by the particular steps that should be performed by the agent in the team plan. CAST uses the concepts of
a.) Operators which are the names of atomic actions that can be taken in the environment. Operators can be either individual or team operators, depending on the number of agents required for that particular operator.

b.) Processes which are invocations of atomic actions

c.) Plans which are essentially designed to describe processes. Plans have pre-conditions and effects and a process description. Team plans have additional constructs for role assignment to the various agents.

CAST agents execute a standard sense/decide/act loop. The sense phase involves querying the simulation server for updating their knowledge of the state of the world and checking the message queue for any relevant messages. The decide phase involves examination of the Petri-Net representation to see if the agent is responsible for any actions. Before acting, the agent attempts to determine if it needs to initiate any interactions, including pro-active information exchange. A specialized algorithm DIARG, (Dynamic Inter-Agent Rule Generator) is used for determining opportunities for proactive Information exchange. DIARG is based on information tuples generated by a knowledge compiler which is queried by the individual agents to generate either tell or ask queries.

A.2 Team-based Agents for SCM

We developed a new agent-based approach to support collaborative sense-making in supply chain management. The approach comes from combining two research initiatives: PSUTAC and CAST. PSUTAC is an agent that models a PC manufacture and
competes with other agents in a simulative supply-chain trading competition [112]. In order to make decisions in an information rich environment, PSUTAC implements a sense-making framework, which includes the steps of information scanning, interpretation, and action. Sense-making approach is different from decision-making approach since the emphasis is not exclusively on the action taken but also on the process of obtaining and interpreting information [113]. In addition, PSUTAC focuses on knowledge representation and acquisition. CAST is a team-based agent architecture that is capable of proactive communication for collaborative interpretation by exchanging of information and knowledge. Team-based agents differ from many multi-agent systems in that team members have shared mental models (SMM) where mutual understanding goes deeper than shared communication protocols and standards [114]. The SMMs include an overlapping knowledge regarding roles and responsibilities.

The approach provides four main benefits. (1) What-if scenarios do not have to be custom built. Simply changing a team member’s mental model such as goals, or roles, or knowledge, suffices. (2) Agents have the potential to reason about the supply chain in a meaningful way and transform the massive amounts of data into compact knowledge. Given that analytic tools are limited in their ability to model complex, multi-firm, multi-dimensional environments, the ability to interact with agents at the knowledge level [112] allows human managers to create and test strategies in a meaningful way. (3) Direct communication regarding time-critical information improves not only the responsiveness of supply chains but also the reliability of communication. (4) Instead of involving multiple parties when implementing or maintaining communication links, the new
approach just needs each member to maintain its own knowledge bases: both a private one used to interpret information and public ones that are shared with others.

In the next section a collaborative sense-making framework is described. In Section 3.2.2, the methodology in which PSUTAC captures and uses knowledge under a sense-making framework is described. Section 3.2.3 explains the idea of using teamwork architecture to support collaborative information scanning and interpretation in a supply chain environment.

A.2.1 Framework of Collaborative Sense-making

A.2.1.1 Sense-Making

Within an organization, collective understanding and action can be promoted through mechanisms such as shared culture and shared meaning [17]. Supply chain members do not share the same culture, or even all of the same goals, so shared meaning becomes even more important in directing supply chain business partners towards achieving common goals [113].

Sense making has been successfully employed to study organizational patterns of information usage [115]. There are three main interrelated aspects: scanning, interpretation, action and its relationship to performance. Scanning involves information gathering and interpretation involves making sense of the information within the context of the organization and its environment. Actions are changes that result from interpretation, and may involve developing new products or services or other strategic considerations. Performance is the means to evaluate the effectiveness of previous steps.
A.2.1.2 Collaborative Sense-making

Interpretation of data by individual business partners in the supply chain may not be consistent and may vary according to individual organizational culture and goals. This can lead to inefficient supply chain performance, as different organizations will respond to a given situation based on their interpretation of how it effects their organization. To counter this, a supply-chain framework should support collaborative interpretation where different participating agents develop uniform interpretations of the supply-chain situation. Collaborative interpretation occurs when a group interprets and transforms a diverse set of information fragments into a coherent set of meaningful descriptions. This activity is characterized by emergence, where the participants’ shared understanding develops gradually as they interact with each other.

We propose supporting collaborative interpretation by modeling supply chains as multi-agent networks, with a shared mental model of the supply chain process and goals. Software agents are a natural means of representing and encapsulating various organizations and intra-organizational processes that participate in the supply chain. Shared mental model [114] refers to an overlapping understanding among members of the team regarding their objectives, their structure, their process, and so on. A shared mental model of the supply chain among the various entities enables more meaningful team communication. It also acts to prevent information overload, as individual members of a supply chain do not need to communicate the entire information set available to them to all other members. Instead each agent has a model of what information is useful or relevant to other members (i.e., a shared mental model).
It is important to note that our proposed agent based collaborative interpretation framework focuses on a different problem from traditional decision support systems. Decision support systems provide the pros and cons of various actions that a manager can take in response to a given situation, so that the primary focus is at the action level of the sense-making paradigm. Collaborative interpretation focuses on problem formulation, a critical but poorly understood aspect of organizational decision making [116]. This process is made even more difficult given a supply chain’s inherently distributed nature.

A.2.2 PSUTAC Agent

The recent Trading Agent Competition on Supply Chain Management (TAC/SCM) game grew out of substantial previous work in agent-based negotiation and coordination, see [117] for more background. In each TAC/SCM game, six participants provide their PC assembly agents, which interact with a customer and eight suppliers created by the game developers. The agents assemble a total of 16 different types of PCs, depending on different configurations for CPU, motherboard, memory, and hard disk. Each game takes place over a simulated year. The assemblers must respond to daily Requests For Quotes (RFQs) from customers with bids, and negotiate with suppliers for PC components by sending out RFQs. Furthermore, they need to decide how to manage the demand and supply of components and products including how to arrange production according to a fixed amount of capacity, how to optimize delivery according to the customer’s order, and so on.
We see the purpose of supply chain games such as TAC/SCM being two-fold. The first is to raise interest and discussion in the area by providing a relatively simple game that still captures some compelling aspects of the overall problem. The second is to raise research issues that can best be explored outside of the game constraints. Thus games like TAC/SCM allow us to do “wind tunnel” experiments, singling out the contribution of individual components, and suggest directions for new theories or practices. However, agent games do not capture many aspects of the real world, including real-world data and interacting with human decision-makers, expert and novice alike. We see the need for simulations involving real-world data and company decision-makers. We also see a role for team-based agents to provide training for human decision-makers given rapidly changing business conditions.

With supply chains becoming more dynamic, information flow in the supply chain needs to be improved for better responsiveness, not only to dynamics of customers’ demand, but also to various unexpected market events—opportunities or negative impacts. When designing a supply chain, however, people often face a dilemma on what information to communicate. On one hand, missing information can cause deleterious effects. In TAC’03, for example, our agent PSUTAC’s failure to respond to the preemption strategy [118] employed by Deep Maize in the semi final rounds proved fatal. The unexpected event (the breakdown of supplies) was not easily identified by the designers. On the other hand, the information rich nature of the SCM domain can cause serious problems of information overflow: useful information will be buried in a huge volume of data. From published reports on TAC’03 teams, few of them could fully use
all of the information because the amount of data that an agent needs to process (under time-pressure) is huge [119].

To demonstrate the methodology through which supply chain entities can be represented as knowledge-based software agents, we present the design of our software agent, PSUTAC [120] that participated in the annual trading agent competition, TAC’03. Specifically, we illustrate how we captured the domain specific knowledge required by the supply chain agent in the form of a formal knowledge base.

A. Knowledge-base driven agent design

Our initial agent design was based on heuristics that were coded within the various agent modules. This design leads to a series of problems with respect to knowledge encoding, knowledge visibility and performance evaluation [120]. To address these issues, we have shifted the agent design to a formal knowledge-base driven software agent. The new approach of the PSUTAC for TAC’04 employs an expert system for decision making. The approach is based on the premise that being able to express market strategies and knowledge in human understandable form allows humans to both have more confidence in the agent’s behavior, and also to contribute to the decision making based on experiences learned from market scenarios like the one presented in SCM. Logic based languages like Jess can be used to capture human’s knowledge about the game domain will be captured in the form of rules [120].

B. Knowledge representation

Although representing knowledge as part of heuristic code or as rules seem to be equivalent, the two methods have fundamental differences in their purposes. A heuristic is a rule of thumb, simplification, or educated guess that reduces or limits the search for
solutions in domains that are difficult and poorly understood [121]. Heuristics are mainly used to simplify an algorithm. In contrast, knowledge is used for making decisions or suggesting solutions. Knowledge can be classified into two categories: (a) declarative knowledge that describes facts and relationships and (b) procedure knowledge that describes how to take actions.

The agent uses declarative knowledge to assess situation. For example, the rule in Figure A-1 indicates that current demand is high if the agent knows the current number of RFQ (from the customer) and the number is larger than 180.

![Figure A-1: Rule for assessing the customer’s demand](image)

The agent uses procedural knowledge to select actions. In Jess, a procedural rule has two parts separated by the "=>" symbol. The first part consists of the Left Hand Side (LHS) patterns; the second part consists of the Right Hand Side (RHS) actions. The patterns are used to match the current situation (represented as facts) in the knowledge base, while the RHS contains function calls. For example, the agent uses the following rule in Table 3-2 to select the price setting functionalities (RHS actions) on the basis of LHS patterns: the customer’s demand and the present inventory level.
C. Knowledge acquisition

Knowledge acquisition, or the process of acquiring domain knowledge, has two different sources: human designers and the agent’s own perceptive functions. In a complex environment like TAC where the range of situations that an agent may encounter is very large, it is essential for the agent designers to add or update agent’s knowledge directly. The TAC game stipulation that an agent cannot be changed in the middle of the game forces knowledge acquisition from human input to be completely offline. Human designers get useful knowledge by analyzing historical data, which is recorded in a database. For example, the average selling price of a product in a low demand game can be set as the base price of that product in the games with similar situations. Human designers may also update the agent’s knowledge by evaluating the agent’s performance. In addition, the agent can update its knowledge online by the perceptive inputs. For example, after receiving a RFQ message, the agent may update its knowledge about the number of RFQs as (number_of_RFQ 230).

The new architecture (Figure A-2) is based on the needs for sense-making including scanning, interpretation, and action. The agent has two functional modules: a database that keeps track of all transactional data and a knowledge base centered kernel. The database records daily transactions and decisions. Human designers may analyze data and code knowledge for the agent offline, or monitor the agent’s online performance through a situation awareness panel. In addition to the knowledge base (implemented in Jess), the kernel also contains four functional modules on scanning, interpretation, decision making about actions, and operational methods. The scanning module processes the incoming messages and extracts and combines data into facts that are useful for
making decisions. The agent needs to process a large volume of daily information. For example, as many as over 300 RFQs can be sent to the agent in a day, and each RFQ contains information about product, quantity, and delivery date. Therefore, the information content of the messages is large in volume. To keep the information as raw data is time consuming for later decision-making functions because, each day, the agent only has limited time to process the data and make decisions. The scanning module can greatly improve the efficiency by extracting and summarizing facts that are needed for making further inferences. After the knowledge base receives the scanned updates, the interpretation module makes inferences and provides the results that the agent uses to make decisions. Interpretation can be done quickly because the process is based on the abstracted trends, not raw data. The result can be used directly to match with patterns specified in the procedure knowledge to make decisions on operations. Although the agent can make inferences directly based on the facts, this step is necessary because (a) it forces designers to be precise when defining the decision variables; (b) since the result will be recorded in the database, it helps designers in location of problems; (c) the results will be displayed at a situation awareness panel, where the designers can evaluate the results by comparing them with observations.
Each decision, or a fired procedure rule, triggers an operational method. For example, if the agent decides to set prices high, a high-price-setting function sets the mean of selling prices to high values. Of course, the operation-method module defines a complete set of operators that are needed for the domain. It should be noted that the knowledge base does not need to keep knowledge that should be defined as an operator. For example, the procedure of the high-price-setting function should not be captured as knowledge.

The knowledge based design makes it easy to evaluate supply chains with various what-it scenarios because the knowledge base is independent of the agent design, thus enabling incremental knowledge addition and modification. The effects of individual pieces of knowledge can be evaluated online by adding or removing rules. In addition, simulation of supply chains at the knowledge level allows human managers to create and
test strategies in a meaningful way. Knowledge is organized as business rules instead of procedurally encoded “if…then…” statements, thus allowing an easy understanding of an agent’s mental model in various situations. Designers can also test the knowledge offline with predefined commands.

A.2.3 Teamwork in SCM

Although PSUTAC design adopted an approach that is consistent with the sense-making steps, it failed to capture the collaboration nature of real supply chains because of the limitation of the TAC/SCM game itself. Since it is a competition, TAC agents focus on locally optimum decisions (i.e., what is best for my company) not global ones (i.e., what is best for the supply chain). By the rules of the competition, they are not allowed to communicate directly with supply chain partners except through the market place. Stepping outside the necessarily restricted TAC domain allows us to conduct research in more real-world situations. As part of this process, we adopted the CAST agent architecture that enables proactive communications based on shared mental models within teams. Our objectives in applying this approach are three-fold. First, we hope the approach can provide a decision support tool to help supply chain designers identify specific information and knowledge sharing needs. Second, we expect the approach can help supply chains be more responsive to unexpected market events. Lastly, the approach can shed some light on how to adopt industry standards to enable stronger coordination in supply chains.
A. Proactive communication in CAST

Research on human teams has repeatedly pointed out that members of high performance human teams can often anticipate the needs of other teammates, and proactively help them regarding their needs [123]. One of the team cognition theories that attempts to explain these teamwork behaviors introduces the notion of “shared mental model” [114], which refers to an overlapping understanding among members of the team regarding their objectives, their structure, their process, and so on. Along this direction, Yen et al. implemented a team-oriented agent architecture called CAST (Collaborative Agents for Simulating Teamwork), which realized a computational shared mental model and allows agents in a team to anticipate the potential information needs of teammates and help them proactively.

The main distinguishing feature of CAST is proactive team behavior enabled by the fact that agents within CAST architecture share the same declarative specification of team structure and process. Therefore, every agent can reason about what other teammates are working on, what the preconditions of the ‘teammates’ actions are, whether the teammates can observe the information required evaluating a precondition, and hence what information might be potentially useful to the teammates. As such, agents can determine what information to proactively deliver to teammates, and use a decision theoretic cost/benefit analysis of the proactive information delivery before actually communicating. The proactive inform behavior has been demonstrated to be useful for enhancing teamwork in the various domains including battle space situation awareness and information fusion [124], army logistic coordination [125], and anti-terrorist information analysis [125].
B. Use of CAST for supply chain interpretation support

A typical supply chain can be viewed as two levels of teamwork: intra-organizational teams (sub-teams) and inter organizational teams (top-teams). A sub-team is composed of various functional areas within an organization such as purchase, sales, and inventory management (see the vendor in Fig. 2). Team members communicate with each other directly. A top team is composed of all the business partners (i.e., sub teams) in the supply chain such as vendors, manufacturers, and retailers. Typically, each sub-team communicates with its business partners through the appropriate business-relationship contact point. In Fig.2, for example, communication between the vendor (A) and the manufacturer (B) is realized as communication between the business contact points—Sales_A and Purchase_B or the logistic contact points—Inventory_A and Inventory_B.

We use Figure A-3, a typical supply chain, to illustrate how CAST agents can reason about information needs of business partners and communicate information and knowledge proactively.

Figure A-3: Proactive communication in a supply chain
Suppose

1) a supply chain includes three business partners A (the vendor), B (the manufacture), and C (the retailer), where B’s organizational structure includes purchase, inventory, production, and sales department;

2) A knows from B’s general processes that it needs a certain raw material X for its production;

3) A knows that the final product Y is made from X;

4) A observed an event—the supply of a raw material X was interrupted by an aggressive competitor.

In this simple case, naturally, A should inform B about this event that X is unavailable now. We describe how CAST agents realize this feature by shared business processes and organizational structures.

In CAST, processes and team structures are represented as MALLET (Multi-Agent Logic Language for Encoding Teamwork) [126] that is a logic-based language for specifying the structures and processes of agent teams. The process describes the procedure of how a team will accomplish their task. To be expressive, MALLET provides a rich set of constructs to define such procedures. A process consists of invocations of operators or plans, or arbitrary combinations using various constructs such as sequential, parallel, conditional, or iterative, blocks, etc. For instance, Figure A-4 describes the two levels of teams in the supply chain and business processes of B’s production and C’s sales.
Figure A-4: An example of MALLET

The precondition of the production process includes the availability of material X, i.e., (available X). By matching the Manufacturing_B’s needs with a Dynamic Inter-Agent Rule Generator (DIARG) algorithm [73], A will proactively deliver the information about unavailability of X to B’s production department. With similar process sharing, A can also inform B’s sales, purchase, and inventory department.

Now, the question is whether A should inform C about the event. Generally A does not want to overwhelm its business partners with irrelevant information. However, given the knowledge that C may soon be considering a sales promotion of Y, and that the
shortage of X will cause a shortage of Y, it is clear that timely dissemination of this information is crucial. Although the information can be send to C by conventional communication means, directly informing C can greatly reduce the delay and improve the responsiveness of the whole supply chain. Thus by sharing some elements of their business processes, and incorporating the knowledge that Y is made from X, Purchase_A can reason that the information is also relevant to Sales_C. However, suppose Sales_C does not know that Y is made from X. Then Sales_C will not understand why Purchase_A sent this information. In that case, Purchase_A can send both the information and the relevant knowledge, which allows the receiver to make sense of the information.

Using team-based agents to facilitated communication has two benefits for supply chains. First, direct communication regarding time-critical information improves not only the responsiveness of supply chains but also the reliability of communication. For example, from Figure 3-5, we can compare the difference between proactive information exchange and traditional method by examining the communication between Purchase_A and Sales_C. The proactive approach enables information delivery in one step. By contrast, without this approach, the message must be relayed at least four times. The delay of Sale_C’s responding to the critical information from Supply_A may cause serious problems such as losing a business opportunity or failure of preventing an imminent risk. In addition, reducing communication links may also eliminate risks of communication breakdown that have been seen in many supply chains. Second, instead of involving multiple parties when implementing or maintaining communication links, the new approach just need each member to maintain its own knowledge bases: private one that are used for interpret information and public ones that are shared with others.
Therefore, simplifying the process to establish and maintain communication links makes supply chains more flexible and dynamic. On one hand, adding or dropping a member is relatively easy. On the other hand, adjusting an agent’s public mental model may automatically affect its interaction with other business partners. To some degree, an agent’s public mental model serves as a company’s public identity and the managers may concisely maintain its identity and enables tighter collaboration.
VITA

Viswanath Avasarala
Pennsylvania State University, College of IST, University Park, PA 16802
Tel: (814) 8658950; FAX (814) 865-6426; email: avasarala@psu.edu

Education
2006  PhD – PSU, College of Information Sciences and Technology, Minor in Statistics
2001    B.Tech – IIT-Madras, Dept of Chemical Engineering

Research Interests
Data-Fusion, Data-Mining, Multi-agent theory.

Publications
• Shuang Sun, Viswanath Avasarala, Tracy Mullen, and John Yen. PSUTAC: A Trading Agent designed from Heuristics to Knowledge. Workshop on Trading Agent Design and Analysis (Co-located with the Trading Agent Competition) at AAMAS'04, July 19-23, 2004, New York City, NY.
• Tracy Mullen, Viswanath Avasarala and David Hall. Market-based Sensor Management. Accepted to IEEE Intelligent Systems, Special Issue on Self-Management through Self-Organization in Information Systems, 2006.

Work Experience
• Research Intern, GE Global Research, Jun’05- Sept’05: Worked on sensor validation, feature extraction and classification algorithms for Non Destructive Evaluation of defects in remote solid structures.
• Software Developer, May-Aug’00 Worked in Tachyon technologies, Madras India on implementation of SMTP and IMAP services for a mail server

Teaching Experience
• Teaching Assistant, IST420, Jan’04-May’04: Assisted in devising and teaching a problem based learning version of the course “Fundamentals of Systems and Enterprise Integration”.
• Teaching Assistant, IST220, Aug’01-May’02 Devised and conducted lab exercises on web-design tools for a class of hundred students

Professional Organizations
IEEE, AAAI