Table of Contents

Abstract..... ii.

Acknowledgments .....iv.

List of Figures.....vi.

List of Tables .....vii.

1 Introduction..... 1

1.1 Advanced Planning and Scheduling.....1

1.2 Multi-Agent Systems.....3

2 Problem Statement..... 12

3 Literature Review 14

3.1 Advanced Planning and Scheduling Systems 14

3.2 Enterprise Integration using Multi-Agent Systems 15

3.3 Auction Frameworks in Manufacturing 17

3.4 Previous Work in Multi-Facility Production Planning Problem 19

4 Solution Approach 21

4.1 Generic Contract Net Based Multi-Agent System..... 21

4.2 Hierarchical Multi-Agent Model for Planning and Scheduling 26

4.3 Mathematical Model of the Multi-Facility Planning Problem.....28

4.4 The Iterative Combinatorial Auction Procedure……..... 31

4.5 Accommodation of Changes in Demand Forecast during Planning Cycle..... 44

5 Results and Discussion..... 48

5.1 Validation of the Auction Mechanism..... 48

5.2 Validation of the Production Planning Approach 53

6 Conclusions and Future Work..... 59

7 Summary and Recommendations..... 61

Appendix A: Software Requirements and Properties Files….... 62

Appendix B: Structure of *MultiPlanner* Production Planning Software..... 65

References…..... 67

Vita …..... 76

List of Figures

4.1 Activity Diagram for Assignment Agent 23
4.2 Activity Short-listing Algorithm..... 23
4.3 Activity Diagram for Resource Agent..... 24
4.4 Activity Diagram for Facilitator Agent 24
4.5 Activity Diagram for Regulator Agent 25
4.6 System Architecture…… 25
4.7 Agent Model of Production Planning and Manufacturing Scheduling……..…. 28
4.8 Precedence Relations between Components……...…… 30
4.9 BOM for Product A…… 36
4.10 Cost Curve for Component i……..……...…… 39
4.11 The Multi-Agent System with Message Flows……..…….…….. 44
4.12 Software Architecture of the Multi-Agent System…….…….. 47
5.1 Sample Output Screen (1)……...……..…….. 56
5.2 Sample Output Screen (2)……...……..…….. 56

List of Tables

4.1 Auction Fees.....32
4.2 Maximum Slack Algorithm.....41
5.1 Auction Framework Simulation Test Results..... 53
5.2 Production Planning Simulation Test Results …….. 58

1 Introduction

This section provides a brief overview of the concepts behind this research study and motivation for the research problem we are trying to solve.

1.1 Advanced Planning and Scheduling

According to Amstel [1] an Advanced Planning and Scheduling (APS) system is an umbrella technology consisting of a full spectrum of solutions, to integrate both enterprise and inter-enterprise planning and scheduling systems. It helps to gather real-time information from different points in the chain, to calculate a feasible schedule, resulting in a fast and reliable response to the customer.

According to Eck [19] the main features of APS which makes it relevant for today's manufacturing enterprises are as follows:

(1) More efficient and dynamic approach than MRP I/II and ERP: MRP I/II (Material Requirement Planning / Manufacturing Resources Planning) functionally carry out planning separately from other links in the chain and the planning is carried out sequentially. This leads to different parts of the supply chain making independent decisions about inventories which leads to higher and unbalanced stocks over the whole chain, also known as Forrester-effect Forrester[23] .

An ERP (Enterprise Resource Planning) system works well in a manufacturing environment which is fairly static. The basic paradigm in a conventional ERP system is to seek '*total visibility*' of system operations in a top-down hierarchical manner. This requires consolidation and frequent updation of all perceivable

aspects of the organization using sophisticated information and database management systems (Ertogral and Wu [20]).This clearly is a very hard task for highly distributed and global organizations.

APS can help in the integration of different aspects of decision-making in a supply chain environment. It is especially helpful in dynamic environments because it is connected to real-time systems and it facilitates frequent change in plans according to market and resource availability conditions. It is also beneficial in that it facilitates the combination of information at multiple sites and calculates an optimal plan across the entire supply chain (Eck [19]).

(2) Geared towards business problems of today and future: APS solutions envision a globally competitive market place, with short manufacturing lead times and tightly controlled inventories coupled with highly uncertain demand environment. The current manufacturing environment is facing many of these problems and it is imperative that these situations would be more common in the future. APS systems are supposed to function efficiently in these environments and hence are geared for changes in manufacturing paradigms in future.

(3) Concurrent demand, material and capacity planning: Instead of using the 'waterfall approach', i.e., sequentially going through demand, material and capacity planning, the three planning variables are considered simultaneously. This results in an integrated, synchronized and cohesive plan for the chain as a whole (Eck [19]).

(4) Total Order Management (TOM): APS solutions can be successfully used for TOM. TOM stands for planning for its fulfillment from the time an order enters the enterprise logging system to its eventual delivery to the customer. According to Eck [19] in order to collect all the needed information to optimize planning for an order, the APS system uses Intelligent Client

Processes (ICP). These processes act as intelligent agents, which collect all the information needed for the planning engine to make decisions. ICPs check availability of components, prepare delivery schedule for components and associated costs. Using this information and capacity information a delivery schedule for the order is produced.

(5) Better integration of strategic, tactical and operational planning levels: The scope of APS is not just limited to factory planning and scheduling. It helps enterprises to evaluate current practices and plan for future operations at tactical and strategic levels too.

According to Eck [19] APS systems are particularly suited for distributed enterprises. The organizations such as semiconductor and automotive manufacturers, where products are developed over a number of production stages, possibly carried out at different facilities which can be distributed through-out the world. The optimization of flow of goods/material and capacity over all manufacturing facilities, is an important decision making problem for these organizations. Given their emphasis on integration of distributed decision-making at different functional levels as well as at the same level in different facilities, APS systems can be effectively applied for planning in such organizations.

1.2 Multi - Agent Systems

According to Woolridge [68] , "an agent is a computer system which is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives". To be classified as an agent the key requirement for a computer system is autonomy, but definition of autonomy is vague and varies across domains. In some environments learning from past behaviors of self and other agents is important but in some other domains it may be unimportant, or even undesirable (in the case of highly reactive systems) (Wiess [67]). To reinforce the authenticity of agents as a new concept in software system design, they have been compared with the popular

object-oriented programming, expert systems and classical artificial intelligence in Woolridge [68]. The main advantages offered by agent-based modeling as opposed to other contemporary modeling techniques are:

(1) **Agents are autonomous**: Though it is true that uninhibited autonomy can lead to chaos, carefully designed autonomous systems can tremendously reduce complexity in modeling of a big and highly inter-connected system. Most real-world systems faced by model designers are highly complicated networks of hierarchically distributed entities which operate on their individual control logic. To use a centralized modeling scheme which tries to combine functioning of all these entities in one model, would be an impossible task in many cases and inefficient in most. Agents can be used to model separate entities using simple rules and then rules of interaction can be defined between these entities to create a model very similar to the real-life system. Use of mathematical programming and other classical techniques in such cases will lead to oversimplification and subsequently ineffective modeling.

(2) **Agents are flexible and socially aware**: Expert systems were very popular in 1980s for modeling knowledge based systems. They acted like consultants which could suggest possible solutions given a problem they were structured to solve. The main drawback of these systems was that they were not a part of the environment which they were modeling, a human operator would feed in the perceived state of environment and the expert system was expected to provide a knowledgeable response. Also, rarely expert systems were developed to interact with other expert systems to try to improve their solution quality. On the other hand, agents are normally supposed to be situated in the environment they model, and they would try to solve the problems which they are not originally structured to solve by negotiating with other agents in the environment.

The computational and physical environment in which agents operate and interact with each other is called a Multi - Agent System (MAS). In the literature multi-agent systems have been used to model complex, dynamic situations where problems can be best solved by an inter-connected network of distributed intelligent problem solvers rather than a central monolithic solver. Several multi – agent systems have been proposed in the literature to address different problems related to manufacturing enterprise integration (Woolridge [68]; Clearwater [17]; Weiss [67]).

According to Gasser[24], Distributed Artificial Intelligence (DAI) is concerned with study and construction of autonomous automated systems which interact with each other and their environment. It considers the social aspect of the coordination problem in multi-agent systems. The basic problems in DAI systems are as follows: (*Adapted from* Gasser[24])

(1) Description, decomposition, distribution and allocation of tasks: Any problem which needs to be distributed among various problem-solver needs to be described in a comprehensive way to aid in its decomposition into separate sub-problems and identification of interesting sub-problems by solving entities. This is accomplished through abstraction and standardization of problems. Once a decomposed version of the original problem is available, the next problem is to select an appropriate way to distribute these tasks between competing (if more than one) entities. An efficient task allocation scheme has to be designed which can meet required design parameters.

(2) Communication, interaction languages and protocols: Once resources have been allocated to tasks, an interaction language which can facilitate coordination between various solving entities is desired. To support this language, we need to define a set of standard protocols through which agents having different internal logic as well as implementation environments can communicate with each other.

(3) Achieving coherent collective behavior: Unchecked autonomy can easily degenerate into chaos, and a major challenge for DAI systems is to exhibit coherent collective behavior. This is generally achieved by defining a minimal

set of common rules for all the entities or major groups of entities and making one of them in-charge of enforcing these rules. The rules for individual behavior and interactions are designed such as to maintain a global objective while having minimal impact on autonomy of entities.

(4) **Modeling other agents and organized activity**: Agents must be able to reason and interpret behavior of other agents present in the system so as to form an agent model to aid in maintaining individual objective while negotiating with other agents(s) having conflicting goals or objectives. This can lead to very complex internal logic in case of a highly dynamic system as well as an environment having a large number of interacting agents.

(5) **Recognizing and resolving inter-agent disparities**: In dynamic and complex systems with a large number of intelligent agents, it becomes very difficult to maintain consistency between knowledge bases of different agents. This inconsistency can be at different levels and can be resolved by a periodic updating of local knowledge bases with a central repository in case of smaller systems or an elaborate conflict resolution system in case of bigger systems.

(6) **Implementation languages, frameworks and environments**: As implementation of intelligent agents is a complex task most conventional development environment are not geared for this task. Object-oriented programming environments with some extensions such as multi-threading are popular means of developing agents and agent systems.

(7) **Methodologies to address practical engineering problems for DAI systems**: Some of the serious practical problems encountered while development, testing and operation of multi-agent systems are heavy bandwidth usage and choking of underlying networking structure in face of enormous amounts of messages exchanged between agents, complex system response to seemingly simple inputs and hard to characterize responses, and lack of elaborate testing schemes to assess the system performance as a whole. We need to develop mechanisms for addressing these problems such as protocols enabling shorter or less number of messages, and devising new and complex testing environments.

According to Baker [6], factory control architectures were traditionally centralized and later hierarchical systems became popular. The popular view among research community until recently was that hierarchical control structures best suit the domain of factory control. The concept of heterarchical control structure, where a number of autonomous entities allocate resources among themselves was first proposed by Hartvany [27]. Coupled with the market-driven contract net protocol (Smith and Davis [59]) it became the founding theory of agent-based applications in manufacturing control. The contract net is a conceptual design for a method of allocating tasks to nodes which can perform these tasks in a distributed environment (Tilley [61]). The basic concept of contract net protocol is straightforward. A manager node which has a single task that can be decomposed into a number of sub-tasks or a number of tasks to be performed, communicates or broadcasts a description of these tasks to contractor nodes which can perform these tasks. The contractor nodes based on their capability, availability and desire to perform the offered tasks submit bids to the manager node. The manager node after considering the bids, awards the individual tasks to the most preferred contractor node. Even though the general mechanism is same, the methods used for task description, communication between nodes, bid formulation and evaluation are part of the detailed design of the system, and vary markedly with the task domain in which the system is applied (Tilley [61]). The authors in Tilley [61] also site examples of some early contract-net based applications in shop-floor control and their drawbacks.

According to (Shen, et al. [57]) agent technology is now recognized as enabling paradigm of next generation of manufacturing design and control systems. It has been applied to a wide variety of problems in concurrent engineering, collaborative engineering design, manufacturing enterprise integration, scheduling and control, as well as material handling and holonic manufacturing systems. It is a multi-disciplinary field integrating inputs from computer science and engineering, electrical engineering and manufacturing and industrial engineering.

Benefits of using MAS in modeling of manufacturing systems can be summarized as follows:

(1) **Ability to model more complex systems realistically**: Because we use a collection of inter-connected intelligent as well as autonomous entities to model the environment, the resulting model is a more realistic approximation of the underlying environment.

(2) **Achieve increased flexibility and adaptability without losing efficiency or productivity**: Because of their learning behavior as well as the ability to negotiate with other agents, multi-agent system based models can be used in a Component of scenarios while giving near-optimal results in these varied scenarios.

(3) **Attain Lean and agile enterprise operations**: Because of more real-time handling of information and software based coordination approach, multi-agent systems can lead to more flexible organizations which can efficiently handle change on various levels as well as cut slack in operations.

(4) **Achieve better integration of enterprise functions**: With better social interaction between agents representing various functionalities of an organization, operational response time can be drastically cut back and operational coordination improved. This leads to better informed as well as integrated enterprise functions.

(5) **Results in a learning system with improved quality of decision making**: Use of multi-agent systems as the backbone of decision-making process improves overall quality of decisions because of their holistic approach.

Enterprise activities involve making decisions related to allocation of resources as well as planning and scheduling of activities at different levels of the organization. At the

shop-floor this may pertain to a supervisor's need to effectively execute production plans so as to complete the jobs on time. A purchasing manager on the other hand, may be concerned with allocation of demand distribution among various vendors. The higher management may be interested in finding out the best way to allocate production between various processing facilities and deciding on which market is served by whom. Essentially, the generic problem is to choose from among several alternatives, the one that optimizes the objective at hand; only the scope and parameters involved in the problem change as we move from one domain to the other. A generic system which encompasses all the common features of a problem solving domain and, then, can be applied to various levels of decision-making with minor modifications will enable an efficient, robust and a quick way to build an enterprise-wide integration framework. The added advantage of using multi-agent system as the back-bone of such architecture is that these systems are not susceptible to the size of the problem and they can still be applicable even if details of the problem dynamically change with time; where the classical mathematical modeling fails. It is hard to imagine that in todays global and competitive operating environment, the problems faced by an enterprise would remain unchanged over time. Agility is the key to survival in today's markets and the markets of tomorrow, and, thus a multi-agent system based architecture is a good approach for building an enterprise integration solution. Also, it would be beneficial from both design and implementation viewpoints to have a generic system, which can later be applied at various levels of an enterprise with minor modifications. Through this research study, we first present a generic multi-agent system which can be used at various levels of an enterprise and then customize this approach to the specific problem of planning over multiple production facilities in a dynamic environment using a combinatorial auction mechanism.

Multi-agent systems are particularly suited for implementation of Advanced Planning and Scheduling systems as the latter consists of a highly inter-connected network of different enterprise functions. MAS provide a "collective" view required for solving multi-lateral integrated planning problem as faced in APS systems involving many entities and multiple objectives. Also, within a function, the advanced design

attributes provided by multi-agent systems in terms of modeling of autonomy and intelligence, help designers to create more complex systems easily. Production planning is one of important functions that are part of APS systems, as it guides production through the shop-floor to the distributors while maintaining profitability and due dates. The planning problem is complicated by demand variations and also uncertainties on the shop-floor. In this research study we focus on developing a solution for the planning problem in an Advanced Planning and Scheduling system which can later be integrated with other modules of the APS.

Today, one of the major planning problems faced by medium and large manufacturing enterprises producing discrete parts is distribution of demand to various production facilities. The problem is further exacerbated by shorter product life cycles, tighter inventories, increased customer expectations and complex supply chains which are all too common for manufacturers operating in a global market and embracing lean manufacturing concepts. An important dimension of the problem is decision regarding which parts to outsource and which to manufacture in-house as well as their respective quantities. According to Wu and Golbasi [69] the need for cross-facility capacity management is most evident in high-tech industries having capital - intensive equipment and short technology life cycle such as the automotive, electronics and semiconductor manufacturing.

Even the single product sub-problem is known to be NP-complete. There have been solutions in the literature based on the lagrangian decomposition method which separates the overall multiple product problem into a number of single product problems. There is still no solution procedure which solves the entire problem holistically. We believe that multi-agent systems, given their distributed problem solving approach can be used to solve this complex problem in its entirety. According to Ertogral and Wu [20] auction theoretic mechanisms are a good way to solve complex production planning problems. We intend to develop a multi-agent system negotiation protocol based on an auction theoretic approach to solve the given problem.

This research study is one part of a bigger research study that we are pursuing at Center for High Performance Manufacturing (CHPM). Our overall objective is the development of an integrated solution for development of an advanced planning and scheduling application suit for our industry partners. Through this work we focus on planning function of the overall system which is assumed to be operating in a highly uncertain market with multiple in-house as well as outsourcing facilities in a multiple product environment. The solution methodology developed for the planning function would be later integrated with a scheduling methodology being developed for individual production facilities. Thus, our research problem is two-faceted. Not only that we develop a novel methodology for planning in a known NP-hard domain but also develop the methodology such that it can later be seamlessly integrated with scheduling methodology being developed concurrently. Later, we introduce the generic architecture on which both methodologies are based, which in turn facilitates consistency in their design and later integration as one solution.

We are specifically concerned with coming up with an agent-based model of the multi-product, multi-facility capacity allocation problem and then suggest an auction - mechanism which can be used by individual agents to solve the periodic planning problem in real time, operating in a dynamic environment. To validate the proposed methodology we create a simulation model of the multi-agent system using the JADE agent environment (JADE [28]) which can be used to model various enterprises and different scenarios typically faced in such environment. The simulation environment is used to compile response of the proposed mechanism in different operating as well as production scenarios. The simulation model is used to first show the effectiveness of the auction mechanism as well as to determine optimal values of auction parameters for specific instances of the problem.

2 Problem Statement

The problem to be solved is described as follows:

Given a multi-facility manufacturing enterprise manufacturing discrete parts in an Make-To-Order (MTO) production environment, and given the customer demand for a particular time horizon, to find a plan of allocating production of various components required according to the exploded BOM for each product, such that the planning function is integrated with the scheduling function of various production facilities, which operate to optimize their individual objectives, while meeting customer deadlines. The production facilities considered can be either a part of the enterprise itself or sister facilities to whom production can be out-sourced. The main characteristic of this scenario is that exact resource capacity at each of these facilities may not be known to the enterprise. The facilities act as autonomous entities with finite production capacity, and carry out production targets set by the planning function of the enterprise to meet their own objectives which can change with time. The planning function should be agile enough to take care of unforeseen disturbances such as order cancellations, order updating, material and resource unavailability at production facilities etc., so that plans can be changed dynamically to solve current demand problem using the current production resources. It is also preferable to incorporate the concept of priority between customer orders in order to expedite processing of rush orders.

Given demand for a particular period for all the end products, we need to allocate this demand in terms of production of constituent components to competing production facilities. The production of these components has to follow the rigid precedence order as indicated in the exploded BOM for the particular product. These production facilities have overlapping capacities, i.e., the same facility can produce more than one component as well as the same component can be produced at more than one facility which compete with one another to secure production of these components so as to maximize the

utilization of their available capacities and also profit. The objective of the solution approach is to minimize the total cost of production, incurred by the system, while at the same time satisfying all of the projected demand in a given planning period.

The solution methodology is supposed to have a two-pass approach. In the first pass, it determines the feasibility of fulfilling the projected demand given the current level of limited production capacity and due dates. If the production can be feasibly scheduled on the production facilities, the next step is to create a master production plan to allocate entire production (fabrication/assembly/inspection etc.) over the given production facilities while trying to optimize the total cost of production given individual objectives of various entities (orders and production facilities).

3 Literature Review

This chapter discusses previous research efforts and state of the art in various domains related to this research study.

3.1 Advanced Planning and Scheduling Systems

Turbide[63], offers an overview of APS capabilities. A detailed description of APS functionalities is provided in Eck[17]. It also provides a list of commercially available APS solutions and a brief comparison of their features. Lee, et al. [37] considers the APS problem in a production environment with finite capacity resources, multiple products with precedence constraints and the option of outsourcing. The main objective of the research study is to produce an APS model that minimizes make-span by considering alternative machines and operation sequences with precedence constraints and outsourcing. The authors develop a GA-based approach to obtain near-optimal solutions in this environment for an integrated process plan and production schedule. The model does not consider multiple in-house production facilities. Even though precedence constraints between operations are considered, the quantity relationships between components are not addressed. This limits the applicability of the solution procedure presented in view of a real-world production system. In our work, we increase the applicability of APS solution procedure by including these extensions. Lee and Kim [38] consider a similar problem of multi-period, multi-product and multi-shop production and distribution problem in a supply chain to satisfy retailer demand. The authors propose a hybrid analytical and simulation based approach to find near-optimal solutions to minimize overall cost of production, distribution, inventory holding and shortages while maintaining due date and capacity constraints. The main drawback of this model is that it does not consider overlapping production capacity between different facilities which limits its application in modern industrial environments. Also, the emphasis on analytical model makes it inapplicable for complex production and distribution networks.

3.2 Enterprise Integration using Multi-Agent Systems

With an increase in global competition and uncertainty, agility has become an important requirement of success for enterprises. New models are needed to capture the complex and dynamic operating environment which is beyond the capability of the mathematical programming and simulation based approaches to model and solve effectively. Multi-agent systems are a branch of Distributed Artificial Intelligence (DAI) which are a logical framework for developing distributed applications to support production network modeling and management (Nigro, et al. [43]). Noori and Mavaddat [44] discussed some of the contemporary issues and solutions in enterprise integration. Similarly, Vernadat [64] and Lim et al. [39] presented good reviews of integration methodologies and their significance. A good reference for current standards for enterprise integration as well as a comprehensive comparison is given in Chen and Vernadat [16]. The reviewers note the lack of satisfactory use of available standards in industry and suggest applicability of these standards only to upper levels of system life-cycle as the disincentive for their restricted use. Use of application specific multi-agent systems which are themselves based on standards prevalent in distributed artificial intelligence community as well as heterarchical manufacturing system design, seem to offer a solution.

Multi-agent system based solutions for enterprise integration was first proposed by Pan and Tenenbaum [45]. According to Jain, et al. [29] autonomous agents provide a good way to coordinate the activities of various entities in a supply chain network. To avoid unrestricted autonomy degenerating into chaos, the authors propose to restrict autonomy of individual agents through the concepts of flexible commitments in a SoCom (Sphere of Commitments). The case of leveled commitments as an approach to coherence in contract net based multi-agent systems for iterative task allocation is discussed in Sandholm and Lesser[56]. The Integrated Supply Chain Management System (ISCM) project (Barbuceanu and Fox[10]) considered manufacturing enterprise as a network of operational nodes, enabling decentralization of control using agent technology. A significant result of ISCM was the development of a generic Agent Building Shell (ABS)

to support agent construction by providing several layers of reusable services and languages (Barbuceanu and Fox[10]). According to Sandholm [55] virtual enterprises which are formed in real-time to take advantage of economies of scale and complementary expertise can use multi-agent systems for negotiations at operative decision making level. In Sandholm [55] the authors discuss different types of contracts possible between agents in a contract net based framework. According to authors in Pancerella and Berry[46] by incorporating agents with their inherently distributed characteristics of autonomy, reasoning and goal-driven behavior, existing EI frameworks can be enhanced to support the paradigm of adaptive virtual enterprises.

Integration of manufacturing systems control is an important part of any enterprise integration effort. To that end, Authors in Lin[40] and Lin and Solberg[41] show that multi-agent systems based on the contract net protocol with monetary transactions can be successfully implemented in a shop-floor environment. AARIA Baker, et al. [7] and Parunak, et al. [48] investigated large-scale resource allocation and system simulation using autonomous agents, in the context of factory scheduling. ADDYMS (Butle and Ohtsubo [14]) was one of the earliest applications of agent technology to distributed dynamic manufacturing scheduling. In this framework agents represented physical resources and a dynamic local resource allocation mechanism for dynamic scheduling was used. Lee and Lau [36] discussed a multi-agent model to enhance the performance of a dispersed manufacturing network, involving companies with different core competencies. The multi-agent model enables monitoring the information flow and task allocation among the network companies. One interesting agent-based framework for intelligent enterprise integration called CIIMPLEX was introduced by Peng et al. [51]. The system was geared towards only higher level interactions between entities, and ignored lower level activities. Chalmeta et al. [15] discussed reference architectures proposed for carrying out enterprise integration. Methods proposed to date are not mature enough and are still improving. A new reference architecture ARDIN was also introduced in Chalmeta et al. [15]. Authors also argue that a parameterized standard solution is better than a totally customized solution for a particular enterprise. It helps in future extensions and adaptation to decision

frameworks of other enterprises (this counters the claim of customized solutions for every enterprise Patankar and Adiga [49]). We have developed a generic architecture which can be applied to various manufacturing enterprises for intra-enterprise integration with minor modifications. The architecture would be discussed in the next chapter.

Virtual Enterprises (VEs) are a new paradigm in enterprise integration where production networks are created in real-time to service a particular customer request between different enterprises. According to Fischer, et al. [22], a virtual enterprise is a temporary, cooperative network that is formed by independent, autonomous companies to exploit a particular market opportunity. As the problems faced in formation and coordination of virtual enterprises are very similar to the ones faced in the formation of a production network between multiple production facilities and outsourcing destinations to meet customer demand, we are interested in solutions proposed in this domain. In Nigro, et al. [43] authors state that an enterprise giving autonomy to each plant in decision-making process in the planning domain, acts as a Virtual Enterprise. Petersen, et. al. [52] provide an agent-based modeling procedure for modeling virtual enterprises using the AGORA multi-agent architecture. Zhou, et al. [71] present an object-oriented technology to support production planning and control in virtual enterprises. Gjerdrum et. al. [25] apply multi-agent modeling techniques to simulate and control demand-driven supply chain network. Authors in Sadeh, et al. [53] introduce the MASCOT agent architecture for modeling and coordination of virtual enterprises which they call dynamic supply chains. Similarly, Wagner, et al. [65] describe the TAEMS agent framework for creation and management of dynamic supply chains. Zhou [70] introduce a learning-based approach for agent-oriented supply chain management. Walsh, et. al. [66] describe a combinatorial auction framework for creation of a virtual enterprise.

3.3 Auction Frameworks in Manufacturing

In a free-market based economic environment scarce resources are allocated to entities who value them the most. The mechanism which decides pricing of goods and actual allocations is called the market mechanism. In a free-market mechanism prices of

goods are supposed to rise with increase in demand and fall with a decrease in demand. Most complex decision-making problems encountered in the manufacturing planning and scheduling as well as other domains involve formulation of an efficient mechanism that can be used to allocate limited resources to tasks which usually have due dates or deadlines. Free-market style mechanisms can help us in coming up simple methodologies to solve such problems (Clearwater [17]). provides a good reference to market-based systems used for resource allocation in manufacturing and allied fields. Kaihara [31] [32] introduce a virtual market-based system which can be used for supply chain management in a dynamic environment. Auctions are the most frequently used allocation mechanisms in market-based control systems. The contract-net protocol, as described in the first chapter does not stipulate a way for the manager nodes to select a particular contractor node or group of contractor nodes to accomplish tasks. Auctions provide a means to achieve that allocation in a simple and efficient manner. A very lucid description of various auction formats and their formal analysis is given in Krishna [34]. In its simplest form an auction is a mechanism in which interested parties submit bids for an item of interest which is offered for sale by a seller. The seller at the end of the bidding process selects the best bid based on a weighing criteria and the object is sold to the bidder who submitted that bid. The price at which the item is sold to the winning bidder is also decided as a part of the auction mechanism.

A new type of auction mechanism known as combinatorial auctions has been recently introduced Parkes [47]. Combinatorial auctions allow bidders to consider more than one parameter of interest as well as multiple items simultaneously. Iterative combinatorial auctions allow bidders to submit multiple bids during the course of an auction. The auction in this case, takes place in multiple rounds, where bidders are allowed to submit one bid for each round based on their eligibility to take part in that round as decided by the rules of the auction. The problem of allocating multiple goods through a single auction is covered in (Ausubel [3]; Ausubel and Cramton[4]). The focus of our research study is to design an auction mechanism which can be used to allocate multiple components to different competing production facilities in a single efficient auction. The presence of product due dates and component precedence relations further

complicate the process. Thus, combinatorial auctions provide a good way to design our auction framework. (Klemperer[33]; Krishna [34]) provide very good references on how to design a new auction mechanism for a particular application.

Authors in Siwamogsatham and Saygin [58], present an auction-based methodology for real-time scheduling of a Flexible Manufacturing System (FMS) with alternative routings of parts. The authors extended the cost function used in the multi-agent scheduling system proposed in Macchiaroli and Riemma [42], to include processing time as a primary criteria.

3.4 Previous Work in Multi-Facility Production Planning Problem

The multi-facility production planning problem was first considered in Bhatnagar [11]. The authors consider a simple case of two production facilities and developed a mathematical model which combined the objectives of the two plants, while maintaining the relevant constraints. They, then, formulated a Langrangian relaxation based heuristic procedure to find near-optimal solutions. Again, Bhatnagar and Chandra [12] provided an exhaustive survey of multi-plant coordination techniques prevalent at the time. They divided coordination into two categories: (i) Coordination among different functional areas; and (ii) coordination of the same function across different facilities. Our research focuses on coordination of the latter type. Tharumarajah [60] presents an extensive survey of resource allocation methods in the distributed manufacturing environment.

Wu and Golbasi [69], present a Langrangian decomposition based heuristic to solve the multi-facility problem as a collection of single-product, multi-facility sub-problems and a resource allocation sub-problem (Dhaenens-Flipo and Finke [18]), introduce a mixed-integer program based solution for the network model of the multi-facility, multi-product problem. The authors admit that the optimal method cannot be used for larger problems encountered in bigger markets (Timpe and Kallrath [62]), also present a mixed-integer programming model to solve the multi-facility production

planning problem. The authors discuss ways to define the capacity of a multi-site, multi-product production network. The authors in Nigro, et al. [43] stress the importance of an agent-based approach in solving the multi-facility production planning problem. They also develop a competing as well as coordination based multi-agent model for the single product, continuous planning environment. The problem which we are trying to solve considers multiple products with sub-component requirements and, as such, the situation that we are considering is much more complex and closer to the real-life scenario. Authors in Pátkai [50], present results of the ModNet project which involved creation of an agent-based toolkit for creation of simulation models of virtual production networks. Brandolese [13], introduce a new approach for allocating common resources to demanding entities using a multi-agent system. The methodology developed by them can support distributed decision-making required in overlapping resource requirement in multi-product environment with uncertain demand.

Recently auction-based approaches to solve the multi-facility production planning problem have been proposed. The auction based systems use the concept of market equilibrium for allocation of resources. Authors in Ertogral and Wu [20], propose a auction-based methodology to allocate production of products to various competing production facilities. Through the use of a competitive market, the decision-making agents are coordinated based on a much simpler set of policies while their long-term behavior can be predicted and modeled using equilibrium conditions. The overall problem is solved as a mutually acceptable negotiation between individual facilities which start from locally optimal plans. Authors in Kutanoglu and Wu [35], present a combinatorial auction framework used for job-shop scheduling. The most interesting contribution of this paper is in suggesting similarities between combinatorial auctions and lagrangean decomposition.

4 Solution Approach

According to Nigro, et al. [43], two approaches are available for managing complex distributed production networks as encountered in the multi-facility production problem: (1) a centralized approach where a central planning entity has access to all the necessary information to make planning decisions for the entire network; and (2) a decentralized approach where each entity in the network has the necessary information and knowledge to make autonomous planning decisions, while the common goal is achieved through cooperation between the network entities. Ertogral and Wu [20], state the several drawbacks in terms of implementation complexity, operational costs, reliability, reactiveness and maintenance costs which centralized approach faces as compared to the distributed decentralized approach. Even though centralized approach is suggested by several popular ERP vendors in their APS tools, considering their drawbacks, we focus on a decentralized approach using a multi-agent system based methodology.

4.1 Generic Contract-Net Based Multi-Agent System

We would first introduce a contract-net based generic multi-agent framework which can be applied in various problem domains and then use an extended version of this framework to solve our particular problem. This section is adapted from Goel, et al. [26].

The general contract net protocol as described in Smith and Davis [59] with some modifications has been shown to work efficiently for solution of problems in diverse domains (Lin [40];Sandholm and Lesser [56]). The generic multi-agent framework we will use has an extended version of the general contract net protocol as its core. The additions made to the general contract-net protocol are:

1. The concept of competing agents rather than cooperating ones as envisioned in the original contract net protocol.

2. The concept of monetary transactions between agents for services provided, which was absent in the original contract net protocol.

These additions increase the applicability of the contract net protocol to enterprise integration by making it more robust and flexible.

The global objective of the environment is to conduct a set of activities, whose execution is temporally bounded (i.e., they have due-dates) while optimizing a particular enterprise objective function (maximize profit or output, minimize cost or defects, etc.). This objective can be a single function of the system variables or a combination of many such functions. The activities constitute a group of tasks; which may be independent, or dependent through precedence constraints. The objective of executing an activity is to get its component tasks processed before the activity due-date, while consuming the resources available in the system such that the overall cost of performing the activity, in terms of the fictitious currency used for fund transfers, is minimized or is not more than the budget allocated to the activity. The tasks are carried out by resource stations (or simply resources) which have an objective of executing only those tasks maximizing their return on investment. By keeping the objectives of various entities in the system vague, we can insert the particular objectives of choice at various levels, while keeping the underlying structure more or less intact.

The generic architecture is composed of the following agents:

The Assignment Agents: This agent interacts with system users through a user interface and accepts problems to be solved using the multi-agent system. It converts problems in native format to a system understandable standard format based on XML. It interacts with other agents in the system for the solution of these problems and communicates them to the user. Each agent in the system has a particular *activity diagram* associated with it, which depicts its actions and behaviors for different stimuli. The activity diagram for an assignment agent is shown in figure 4.1.

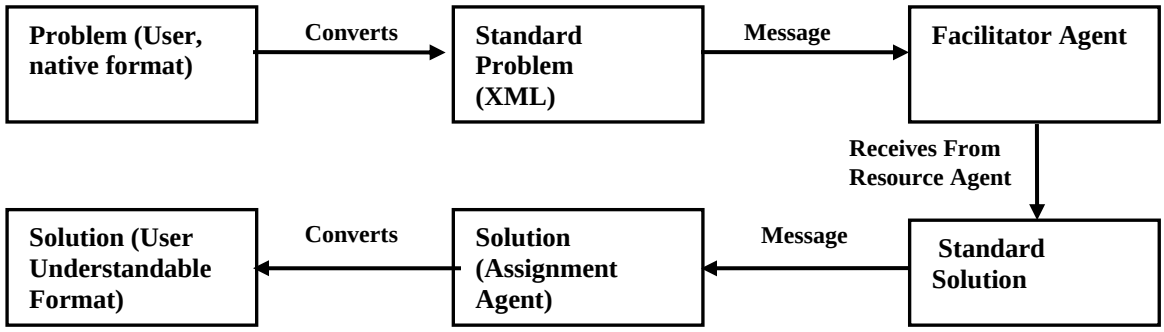

Figure 4.1: Activity Diagram for Assignment Agent

The Resource Agents: This agent models the various resources available to the multi-agent system. The types of resources available would be different for different levels of the enterprise. These agents bid for the problems that are posted by the Assignment agents. The agent has the knowledge about the problems it can handle as well as their payoffs to it. The agent can also delegate its work to other agents. The *activity selection* algorithm used by a Resource agent is explained in Figure 4.2. The activity diagram for a Resource agent is shown in figure 4.3.

- Poll all the blackboards; select the set S of available activities from the activity template which can be feasibly done by this resource.
- Read parameter value requirements for each activity in this set. If parameter values lie beyond the range possible on this resource, then remove particular activity from the set S.
- Calculate current scores for all the activities based on their ranks in the activity template of the resource agent. Adjust scores for any preferred parameter values.
- Rank the set S based on the scores obtained in the last step, with an activity with the higher score being ranked higher. In case of a tie in scores, break the tie in favor of the task having a higher rank in the ranked list of activity template of the resource agent.
- Based on the range in which cardinality of S lies and short-listing range criteria set by the user, pick the top *n* activities from the set S and call them *ready set,* R.
- Prepare bids for all activities in set R.

Figure 4.2: Activity Short-Listing Algorithm

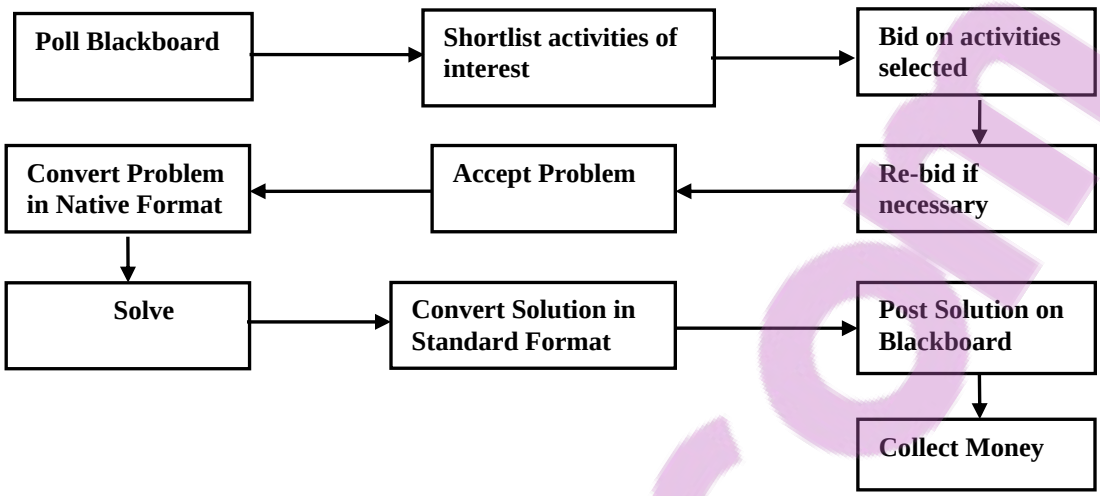

Figure 4.3 : Activity Diagram for Resource Agent

The Facilitator Agent: This agent accepts problems in standard format from the Assignment agents and uses a multi-criteria bid selection rule to distribute tasks to appropriate Resource agents. It creates a Blackboard object to monitor activity status. The activity diagram for the Facilitator agent is shown in figure 4.4.

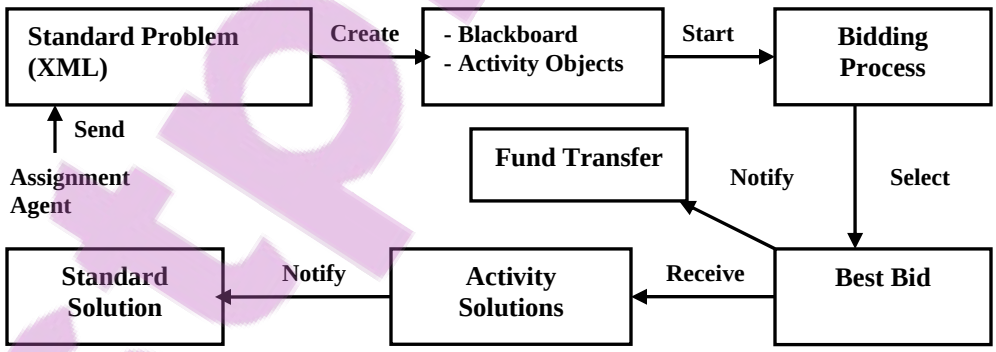

Figure 4.4: Activity Diagram for Facilitator Agent

Regulator Agent: This agent carries out requested monetary transactions for agents. It provides a transparent and trusted environment for transfer of funds between agents. The activity diagram for the Regulator agent is shown in figure 4.5.

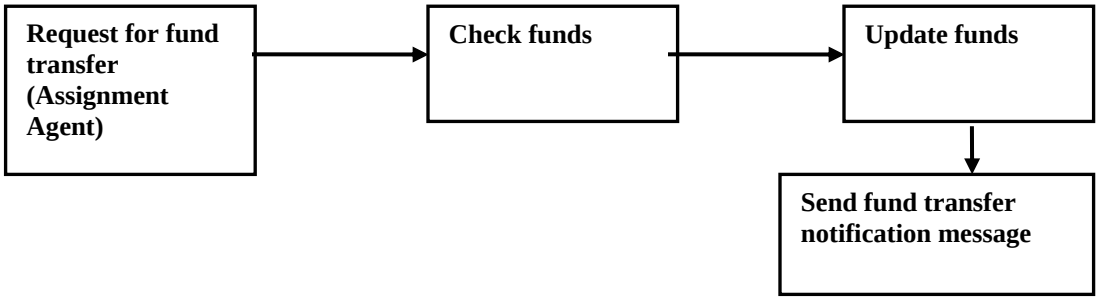

Figure 4.5: Activity Diagram for Regulator Agent

The overall structure of the proposed architecture is shown in Figure 4.6. The message flow between the agents is indicated by the arrows. The messages follow the standards for an agent communication language (ACL) as defined in FIPA-ACL (FIPA [21]). Also, the message content is XML based and follows the particular ontology of the level at which the agent is situated.

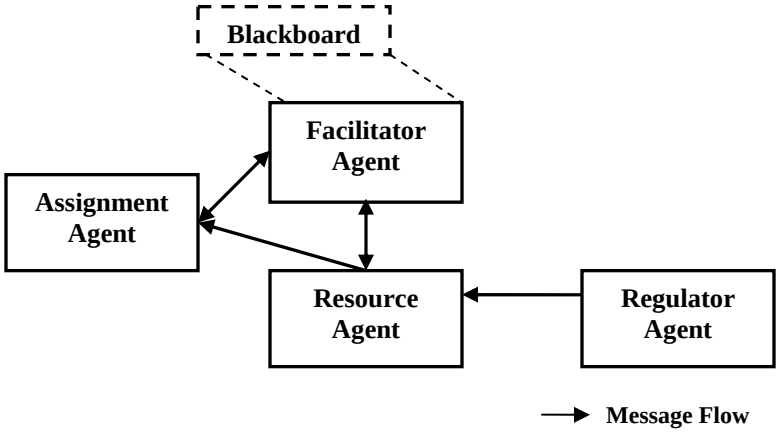

Figure 4.6: System architecture

The operation of the framework is explained as follows:

The Assignment agent accepts the problem from the user in a native format (user-understandable format). It decomposes the problem into precedence-related activities which can be performed by individual Resource agents. These activities are encoded into the standard format as would be understood by the Resource agents. It sends a message to the Facilitator agent containing the problem definition in standard format and parameters related to its execution such as deadline, etc. The Facilitator agent then creates a blackboard object for this problem. The Resource agents poll the blackboards and select those problems (activities) that are of interest to them. They select activities on which they would submit their bids based on the activity selection algorithm as discussed earlier. After creating bids, they send their bid messages to the Facilitator agent. Resource agents are notified by the Facilitator agent when they get outbid so that they may submit further bids. Bids are allowed till a pre-determined auction termination time is reached, when the auction is said to be over. When the auction is over, the activity is awarded to the "winning bidder" (selected using the particular bid-selection rule). The Assignment agent that proposed the problem is also informed as to which Resource agent was awarded the particular activity. The winning Resource agent then executes the activity and posts the solution on the blackboard. The Assignment agent is notified to start the fund transfer process, which sends a message to the Regulator agent requesting transfer of appropriate funds from its account to the corresponding Resource agent. Once solutions to all the activities of a particular problem are posted by corresponding Resource agents, the Facilitator agent sends the solutions in standard format to the Assignment agent. The Assignment agent integrates these solutions and builds the solution to the original problem, and displays the solution to the user after converting it into the native format.

4.2 Hierarchical Multi–Agent Model for Planning and Scheduling

We create a hierarchical multi–agent model for planning and scheduling functional modules of the APS. This model is described in Figure 4.7. The Planning Agent is in-charge of solving the multi-facility multi-product planning problem at the start of each planning period. It receives the demand for the various products from the

Demand Agent. It has more than one Facility Agent representing the various production facilities of an enterprise interacting with it to arrive at a feasible and "good" solution to the planning problem. The Facility Agent has several Shop Floor Agents interacting with it for the component level planning. The Facility Agent also interacts with the Inventory Agent to gain knowledge about current inventory levels of various products. Using the plan for a specific Shop Floor Agent, the Scheduling Agent builds the schedule for production of the jobs as a result of the interaction of Machine and Job Agents. The Flow Shop, Job Shop and Cell Agents are the lower level agents depending on the configuration of the machines present in a particular shop floor. Each of these agents is comprised of various Machine Agents. The Material Handling Agent represents the material handling equipment available in the shop floor and, depending on the configuration they may or may not interact with Machine and Job Agents for developing a production schedule of the jobs. The Product Agent represents the finished goods and the Job Agent is the components that the product is made up of as determined by the bill of material. For each resource (machine or material handling equipment), the operator associated with it is represented by the Operator Agent, which in some cases might not be required for fully automated machines like FMS, AGV etc.

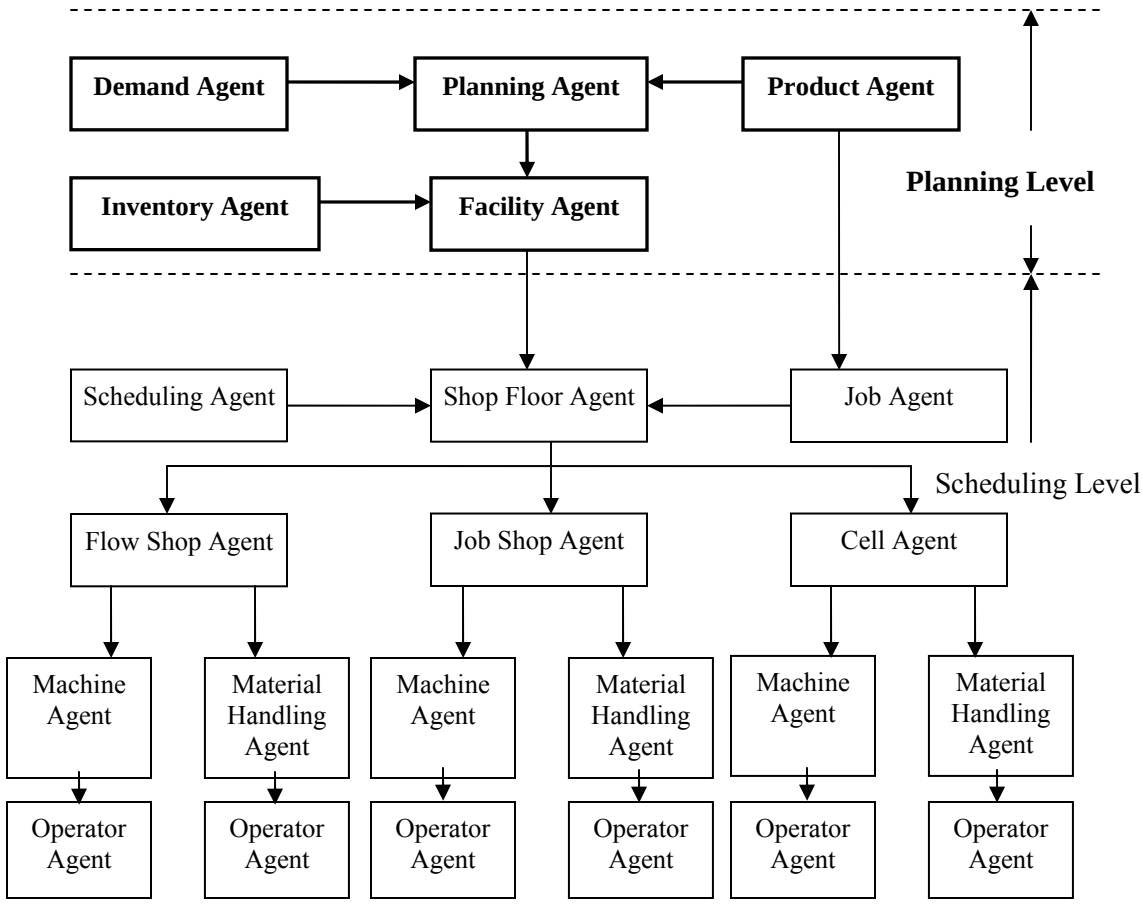

Figure 4.7: Agent Model of Production Planning and Manufacturing Scheduling

4.3 Mathematical Model of the Multi-Facility Planning Problem

We first build a mathematical programming model of the problem we are trying to solve. This serves the following purposes:

(1) We form a formal model of the problem to be solved.

(2) We obtain a framework against which to compare the performance as well as validity of our agent model.

(3) We get an idea of the complexity of the problem we are trying to solve.

Terminology:

i : index of components

i^e : index of finished products

i, i^e $\in$ N, the set of components/products

f : index of facilities, f $\in$ F, the set of facilities

t : index of time periods, t $\in$ T, the length of planning period

d_i^e : demand for end product $_i^e$ over the planning period

C_t^f : aggregated production resource capacity available at facility r during period t

S_i^f : setup cost for component i if produced at facility r

P_i^f : processing cost for component i if produced at facility r

H_i^f : inventory holding cost/period for component i at facility r

L_i^f : Lower bound on batch size for production of component i at facility r

U_i^f : Upper bound on batch size for production of component i at facility r

B_i^f : Upper bound on inventory level for component i at facility r

u_i^f : aggregate production resource needed per unit of component i at facility r

N_{pi} : the set of predecessor components for component/product i

(The Level $_{n-1}$ components needed for production of a Level $_n$ component i)

N_{si} : the set of successor components/products for component/product i

(The Level $_{n+1}$ components which need Level $_n$ component i)

a_{ij} : number of component j needed for starting processing of one unit of component i

$x_{i\,t}^f$: quantity of component i produced at facility r in period t

$y_{i\,t}^f$: 1, if $x_{i\,t}^r > 0$

0, otherwise

$b_{i\,t}^f$: quantity of component i carried in inventory at facility r in period t

Objective: Minimize Total Setup cost + Production cost + Inventory holding cost at all facilities across the planning period

$$\text{Min.} \sum_{i}^{N} \sum_{f}^{F} \sum_{t}^{T} (S_i^f * y_i^f{}_t + P_i^f * x_i^f{}_t + H_i^f * b_i^f{}_t) \quad \ldots (4.1)$$

s.t. :

<batch size constraints>

$$L_i^f \le x_{i\,t}^f \le U_i^f \qquad \forall f, i, t \qquad ...(4.2)$$

<inventory size constraints>

$$b_{i\,t}^f \le B_i^f \qquad \forall f, i, t \qquad ...(4.3)$$

<capacity constraints>

$$\sum_i^N x_{i\,t}^f * u_i^f \le C^f{}_t \quad \forall f \qquad ...(4.4)$$

<due date constraints>

$$\sum_r^R (x_i^{ef}{}_T + b_i^{ef}{}_{T-1}) \ge d_i^e \qquad ...(4.5)$$

<precedence constraints>

$$a_{ij} \sum_r^R x_i^f{}_t + \sum_r^R b_j^f{}_t = \sum_r^R (x_j^f{}_t + b_j^f{}_{t-1}) \quad \forall j \in N_{pi}, \forall i \in N$$

…(4.6)

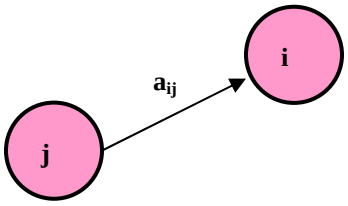

Figure 4.8: Precedence Relation between Components

<Batch size constraints>

$$x_i^f - L_i^f * Y^f \ge 0 \quad \forall f \in F$$
$$x_i^f - U_i^f * Y^f \le 0 \quad \forall f \in F \qquad ...(4.7)$$

This is an NP-hard problem and to optimally solve a reasonably sized problem is impossible. We need a heuristic procedure to solve this problem in real-time and we propose an agent-based combinatorial auction procedure as a good way to solve even bigger problems with reasonable results.

4.4 The Iterative Combinatorial Auction Procedure

Combinatorial auctions allow bidders to simultaneously bid on bundles of items, rather than just on a single item. Although complex to set up and understand, this type of auction nevertheless has been successfully used in multiple problem domains (Ausubel and Cramton [4];Parkes [47]). Whereas conventional auctions typically involve only a single parameter of interest, such as the price offered for a given item, combinatorial auctions instead involve multiple such parameters. For example, in order to determine a winning bid, one might need to consider both the unit price associated with supplying a given type of item and the corresponding quantity to be supplied. This use of multiple parameters adds significantly to the complexity of the auction.

We assume we are given demands for individual end products for the entire planning period in terms of a demand forecast. Each product has a unique BOM (Bill of Materials) which is a rigid precedence structure between constituent components. An example of BOM is given in Figure 4.8. The components can be manufactured at various in-house as well outsourcing facilities. These facilities have overlapping capacities, i.e., the same facility can produce more than one component as well as the same component can be produced at more than one facility which compete with one another to secure production of these components so as to maximize the utilization of their available capacities and also profit. The objective of the system is to minimize the total cost of production while, at the same time, satisfying all of the projected demand in a given planning period. The system is also flexible enough to efficiently take care of changes in actual demand during the planning period and change the initial production plan between various facilities.

We first describe a general iterative combinatorial auction mechanism which we have developed for multi-lateral negotiations between the buyer and the sellers. The auction framework is applicable for a single buyer and multiple sellers. The buyer order may be

very large and may consist of multiple units of different commodities. The sellers might be required to form a *virtual cooperative* to satisfy the entire demand of the buyer. The seller posts the requirement for the season on a bulletin board system which can be accessed by sellers who may be interested in forming a virtual coalition to satisfy that demand. To prevent abuse of the bulletin board service and to maintain wholesale level of transactions, the buyer is charged two types of fees to use its services. The *Attendance fee* is a one-time fee charged for each requirement posted by the buyer. The *Round fee* is charged from the buyer for each round of auction conducted for a particular requirement. The presence of this fee optimizes auction length as explained in Table 4.1. The use of these fees as a deterrent to prevent auction participants to deviate from desired behavior is one way of encouraging such behavior. The main drawback of this approach is that fee values chosen play an important role in performance of the auction procedure as they determine frequency and length of auctions. But, this is a good way of maintaining order in the system without restricting autonomy of auction participants. This is important because we are going to use software agents in the bidding process, where autonomy is a key requirement.

Table 4.1: Auction fees

Attendance fee (Demand Agent)	One-time fee charged for conducting a set of auctions for a demand agent. It helps prevent demand agent to obtain free estimates of bidding behavior of bidding agents.
Round fee (Demand Agent)	Per-round fee the demand agent must pay to extend the auction. It helps restrict the length of auctions. Its value is inversely proportional to number of components involved in the particular demand requisition.

The buyer has a set of prices called the *reserve prices* for each component, above which it would be infeasible for it to carry out the transaction with involved sellers.

Thus the buyer objective function is:

$$\text{Max.} \sum_{i}^{V}(p_i^{r} * Q_i) \quad - \quad \sum_{j}^{N}\left(\sum_{i}^{V} p_{ij}(q_{ij})\right) \quad - \quad n_R * F_R \quad - \quad F_A \qquad \ldots(4.8)$$

Where,

p_i^{r} : Reserve price for component i

Q_i : Quantity of component i desired by the buyer

$p_{ij}(q_{ij})$: Price of component i for seller j. It is a function of quantity allocated q_{ij}

n_R : Number of rounds in auction

F_R : Round fee (See Table 4.1 for details)

F_A : Attendance fee (See Table 4.1 for details)

The first round of the auction is started following the initial exchange of information (requirements and fees). Each round is of pre-specified time duration and bids are accepted at any point before the end of the round. Seller agents create bids based on a utility function that incorporates the fee values for the current auction with their private cost functions for each of the varieties. Each bid covers all commodities that the seller is interested in and consists of fixed and variable prices and minimum and maximum quantities that can be supplied for each of those commodities. The total amount bid cannot exceed the capacity available with the seller. Once the round is closed, the bulletin board service evaluates the submitted bids to select a winning coalition of sellers for the round.

The following bidding rules are enforced:

- The sum of the maximum amount bid for all the commodities by a seller cannot exceed total production capacity available to the seller.
- A bid declared as winning in a round cannot be retracted in the next round. It can be either made better by reducing the asking price by at least a *minimum bid decrement,* or by increasing quantity at the same price, or kept the same as the *freezing bid*, which cannot be altered in subsequent rounds. A Freezing bid gives the option to sellers to stop improving bids once they reach their minimum allowable profit level.

The auction continues until the buyer is ready to pay the fee for a subsequent round. The round at which buyer signals the bulletin board to discontinue the auction becomes the last round and the allocations at the end of this round become the final allocations to the winning producer agents.

The seller objective function is:

$$\text{Max.} \sum_{i}^{V} \left(p_{ij}(q_{ij}) \; - \; c_{ij}(q_{ij}) \right) \qquad \ldots(4.9)$$

Where,

$c_{ij}(q_{ij})$: is the cost function of seller j for quantity q_{ij} of component i

As the goal of the auction is to maximize the social welfare, i.e., to maximize the value obtained by all participating agents, the overall objective function for the auction is given by the sum of the buyer's and the seller's objective functions (for all sellers participating in the auction):

$$\text{Max.} \sum_{i}^{V} \left(p_{ij}(q_{ij}) - c_{ij}(q_{ij}) + p_i^{\,r} * Q_i \right) \; - \; \sum_{j}^{N} \left(\sum_{i}^{V} p_{ij}(q_{ij}) \right) \; - \; n_R * F_R \; - \; F_A \qquad \ldots(4.10)$$

Now using the general auction framework, as described above, we create an auction-based negotiation framework for the multi-facility production planning problem. The multi-agent system introduced here consists of Demand, Planning, Product, Inventory and Facility Agents, as shown in figure 4.7 for the production planning level. The agents interact with each other through XML-based messages. The agents are able to handle complex bid calculations and their submissions in a timely manner so that auction length can be minimized. The planning agent carries out auctions for various available components once it receives a demand signal from the demand agent, and the facility agents send bid messages to the planning agent to participate in the auction, as explained in Figure 4.8. The *blackboard* is a bulletin board system which is used by the facilitator to initially advertise the auction to interested producer agents and later to gather their bids and store results of auction rounds.

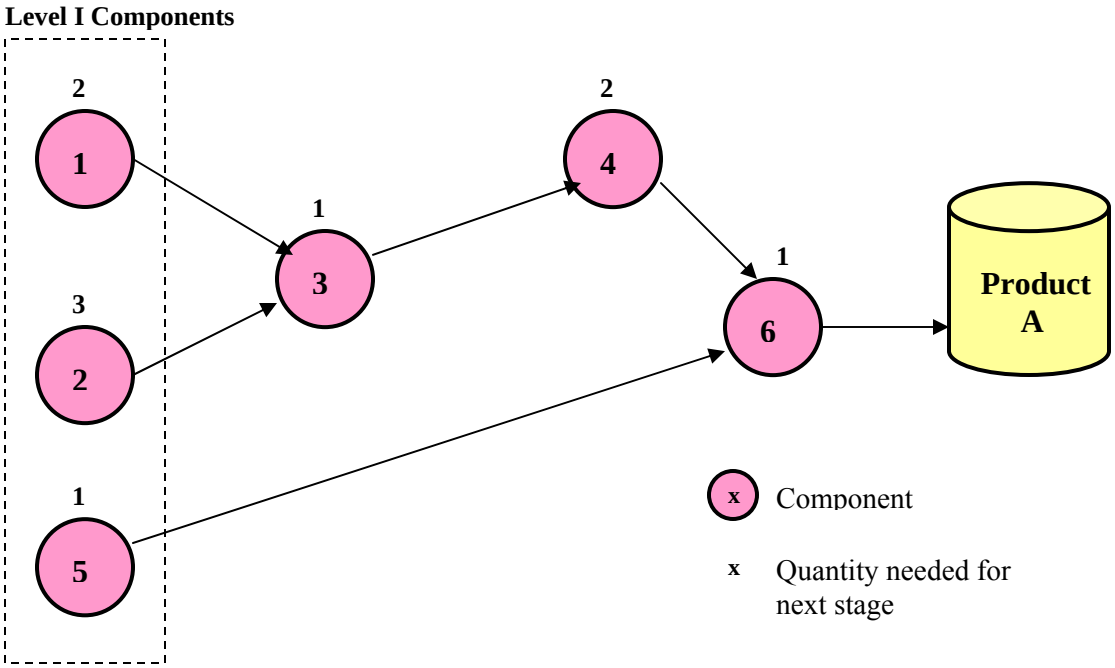

Figure 4.9: BOM for Product A

The *Demand Agent*, based on forecasted or actual demand sends a *Demand Message* to the Planning Agent, requesting the demand to be planned over available facilities. The structure of Demand Message is as follows:

```
<requirement agent_id="agent_name" demand_id="demand_identifier">
        <products>
                <product name="prod_name" quantity="num" reserve="num" start="num" />
                .....
                .....
                </products>
</requirement>
```

The *Planning Agent*, which is in-charge of carrying out the auction calculates actual requirement for each of the components for satisfying given demand of the end products. It then posts the requirements for Level I components for all products to the *Facility Agents*, which represent individual production facilities as well as outsourcing destinations. Let the demand for individual components as calculated be d_i^t. The Planning

Agent creates a Blackboard object to store and process information regarding the auction. The Planning Agent notifies Facility Agents about the first auction through a *Blackboard Created Message* whose structure is as follows:

```
<blackboard agent_id=" agent_name " demand_id="demand_identifier ">
        <components>
                <component name="comp_name" quantity=" num " start=" num " />
                        .....
                        .....
        </components>
</blackboard>
```

The Planning Agent notifies the Demand Agent about fee values (Table 4.1) through a *Fee Message*. The structure of Fee Message is as follows:

```
<fee>
        <attendence_fee> num </attendence_fee>
        <round_fee> num </round_fee>
</fee>
```

Through the Created Blackboard Message, the Planning Agent calls for bids from Facility Agents. The Facility Agents respond with *Bid Messages*, which carry the following information

For each component on which the facility bids

<The minimum and maximum batch size for the component>

<The price/unit for the component (includes setup/production/inventory cost)>

<The time required to complete the entire batch>

End For

The structure of *Bid Message* is as follows:

```
<bid demand_id=" demand_identifier ">
        <components>
```

```
            <component name=" comp_name " min=" num " max=" num " unit_cost=" num "
fixed_cost=" num " time=" num"/>
            .....
            .....
        </components>
</bid>
```

As we include setup cost as one dimension of the cost, it is clearly non-linear, and as such per unit cost is not a constant but marginally decreases from minimum to maximum batch size. An example cost curve is shown in Figure 4.10.

Referring back to the Mathematical model of the multi-facility problem, The per unit cost at a given value of x_i^f is given by: (t is omitted from this discussion, as we are analyzing bids at a single point of time)

$$\frac{x_i^f = S_i^f + v\,(x_i^f - L_i^f)}{x_i^f}$$

price p_i^f as a function of x_i^f is defined as $p_i^f(x_i^f) = S_i^f + v\,(x_i^f - L_i^f)$

The bid message consists of the cost curve parameters (S_i^f, L_i^f, U_i^f, v)

The Planning Agent receives this information from all facilities for all the components as first round bids. The Facilitator Agent runs an LP based algorithm to fit these bids to the demand such that the overall cost is minimized.

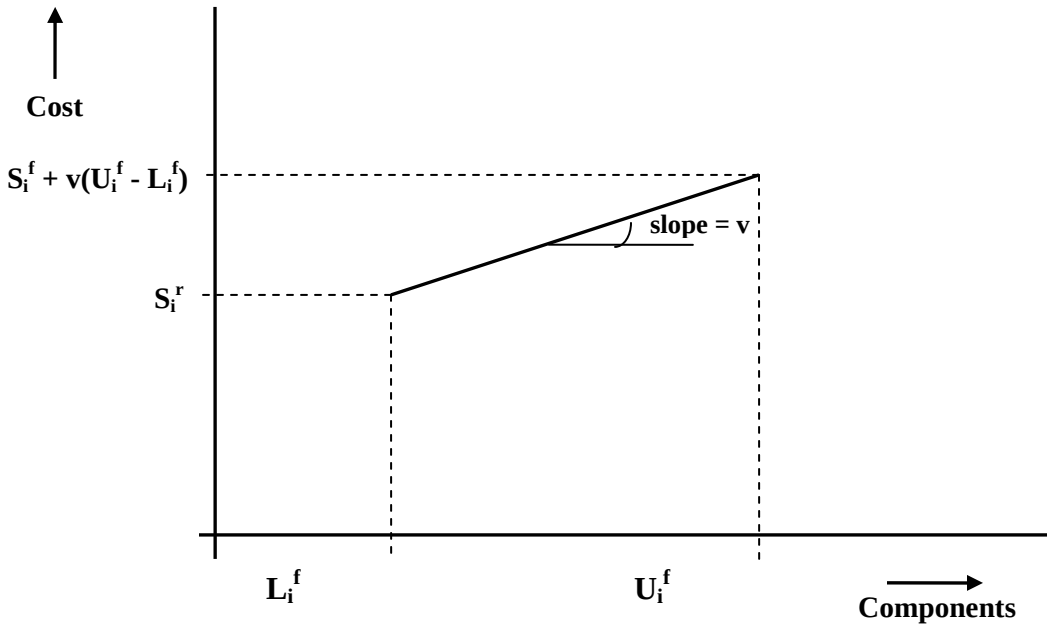

Figure 4.10: Cost Curve for Component *i*, Facility f

The winner determination algorithm is as follows:

For each component i in the auction, find optimal solution to the following LP,

$$\text{Min.} \sum_{f}^{F} S_i^f * Y^f + v\,(x_i^f - L_i^f * Y^f) + p_{\max}(x_i^{F+1}) \qquad ...(4.8)$$

s.t.:

<demand constraint>

$$\sum_{f}^{F} x_i^f + x_i^{F+1} = d_i \qquad ...(4.9)$$

<Batch size constraints>

$$\begin{aligned} x_i^r - L_i^f * Y^f &\geq 0 \quad \forall f \in F \\ x_i^r - U_i^f * Y^f &\leq 0 \quad \forall f \in F \end{aligned} \qquad ...(4.10)$$

F+1 stands for a dummy facility which is used to fill the demand which is not satisfied by bids submitted. It is assumed that this demand is filled at a very

high price p_{max} by the dummy facility. Thus, Facility Agents interested in the component can bid with a lower value in the next round. The value of x_i^{F+1} decreases as the auction proceeds.

Using this algorithm the Planning Agent allocates the winning production amounts among the *winning coalition* of Facility Agents. Winning coalition is the set of Facility Agents which win allocation of atleast some production. Each round is of pre-specified time duration and bids are accepted at any point before the end of the round. The Facilitator agent collects all bids from the Facility Agents and then evaluates them after the round has closed. The above winner determination algorithm is used to select winners and their contributions to the production of each component.

The Planning agent then informs the Facility Agents of the winning prices and quantities through a *Round Winners Message*. The format of this message is as follows:

```
<round_winners agent_id=" agent_name " demand_id="demand_identifier " round_num="1">
        <components>
                <component name=" comp_name " quantity="num">
                        <facility price=" num " quantity=" num ">
                        <facility price=" num " quantity=" num ">
                </component>
                .............
                .............
        </components>
        <total_cost> num </total_cost>
</round_winners>
```

At the end of the first round and each subsequent round, the Planning Agent decides to continue the auction given the tradeoff between *round fee* (See Table 4.1) and the reduction in overall cost obtained using the previous auction round. If after completion of a round, the Planning Agent decides that not much can be gained by calling another round of auction, the final round of the auction is started. The duration and bid selection rule are the same for the last round as for other rounds, and the Facility Agents are notified about the winning bids as before. If after completion of the final

round a Facility Agent has won a production contract for some component then it is notified about its allocation.

The following bidding rules are enforced between rounds:

- Agents cannot bid for components which were not specified in their Interest message.

- From round two onwards, a bidding agent has to specify which winning bid it wishes to replace. There are two ways of specifying a better bid - by quoting a lower price with the same quantity or by quoting the same price with a greater quantity. Winners of the previous round are allowed to bid the same amount again, but it then becomes their *freezing bid* and can't be changed in subsequent rounds.

The Facility Agents follow a particular bidding strategy to maximize their individual utility function. This strategy, which we call the *Maximum Slack Algorithm,* is explained in Figure 4.9. It is based on the heuristic rule that the probability of a facility winning is always higher for a component in which the difference between current price and producer cost (slack) is greater, since the facility has more price flexibility. The experimental simulation results presented in Section 5 demonstrate that the maximum slack algorithm is a dominant equilibrium bidding strategy for producers. It is only weakly dominant, however, in that it is not the only equilibrium strategy which can be used within an implementation of facility agents.

Table 4.2: Maximum Slack Algorithm

- Initialize
  Available_Size = Capacity
  For all varieties Component.Allocated = 0
  For all varieties Component.Available = true
  Set hasWon flag and Size_Won for Varieties won in last round
  Set hasBids flag for varieties with no bids
  Reset Bid_Freeze flag for all Varieties not winning now

- For all won varieties
  - If (HighestSlackNonWinningComponent(available_size) > Component.slack)
    - Freeze Bid
  - End If
  - Component.allocated = Component.Size_Won//cannot retract a won bid
  - Update Available_Size
- End For
- For all varieties with no bids
  - Set Component.Attack_Price = Component.Start_Price
- End For
- For all varieties not won but with bids
  - If(cannot bid more on this Component given its maximum bid price)
    - Component.Available = false
  - End If
- End For
- While(true)
  - if(!isAnyComponentAvailable()) Quit While
  - if(isInfeasible(Available_size)) Quit While
  - calculateSlacks(available_size)//Calculate Slacks for all available varieties
  - //Slack = Attack_Price - (unit_cost + fixed_cost/max_allowed)
  - //max_allowed = maximum amount which can be bid for the Component given bounds and Available_size
  - Let max_slack_index = Index of Component with maximum slack
  - Let max_permit = Maximum allocation to maximum slack Component given bounds, Available_Size and Component requirement
  - Update Available_size = Available_size - max_permit
  - Component[max_slack_index].available = false
- End While
- For all varieties
  - updateAttackPrice() //based on agent type
  - calculateBidPrice() //based on the Component.Attack_Price
- End For

Special Case: First Round

- Initialize
  - Avaialable_Size = Field_Size
  - For all varieties Component.Allocated = 0
  - For all varieties Component.Available = true
- While(true)
  - if(!isAnyComponentAvailable()) Quit While
  - if(isInfeasible(Available_size)) Quit While
  - calculateSlacks(available_size)//Calculate Slacks for all available varieties
  - //Slack = Attack_Price - (unit_cost + fixed_cost/max_allowed)
  - //max_allowed = maximum amount which can be bid for the Component given bounds and Available_size
  - Let max_slack_index = Index of Component with maximum slack
  - Let max_permit = Maximum allocation to maximum slack Component given bounds, Available_Size and Component requirement
  - Update Available_size = Available_size - max_permit
  - Component[max_slack_index].available = false
- End While
- For all varieties
  - updateAttackPrice() //based on agent type
  - calculateBidPrice() //based on the Component.Attack_Price
- End For

Once the Planning Agent decides that continuing the auction is not feasible due to lesser returns in terms of reduced costs as compared to the round fee, the auction is called off at the end of the current round, and the winners at the end this round are deemed the final winners. Once the auction is over, the Planning Agent sends an *Auction Winners Message* to Facility Agents to notify them about the final allocations after completion of the auction. The format of the message is as follows:

<auction_winners agent_id=" *agent_name* " demand_id="*demand_identifier* " >
    <Components>
        <Component name="*comp_name*" quantity="*num*">
            <facility name="*fac_name*" price=" *num* " quantity=" *num* ">
            <facility name=" *fac_name* " price=" *num* " quantity=" *num* ">
        </Component>
        .....
        .....
    </Components>
    total_cost>*num*</total_cost>
</auction_winners>

The Facility Agents use this message to update values of their objective functions as well as available capacities. Based on the amount of components won, Facility Agents push a *Production Event* in the global *Event Queue* as shown in figure 4.10. The Event stores information regarding which component would be finished, batch size and the time of the event.

Once the auction is over for Level I components and production plan is communicated to all facilities, the simulation clock is advanced to the next production completion event by the Planning Agent, which is defined by the lowest time at which any allocated production would be finished by a facility. If production of any Level II components is possible at this time, their appropriate quantity is 'opened' for auction and the similar auction procedure as above is carried out to find allocations across different

facilities. This is done by the Planning Agent by pulling an event out of the global Event Queue.

Once all auctions are over and the clock reaches the end of the planning period, we have a master production plan ready to be used to coordinate production across multiple facilities.

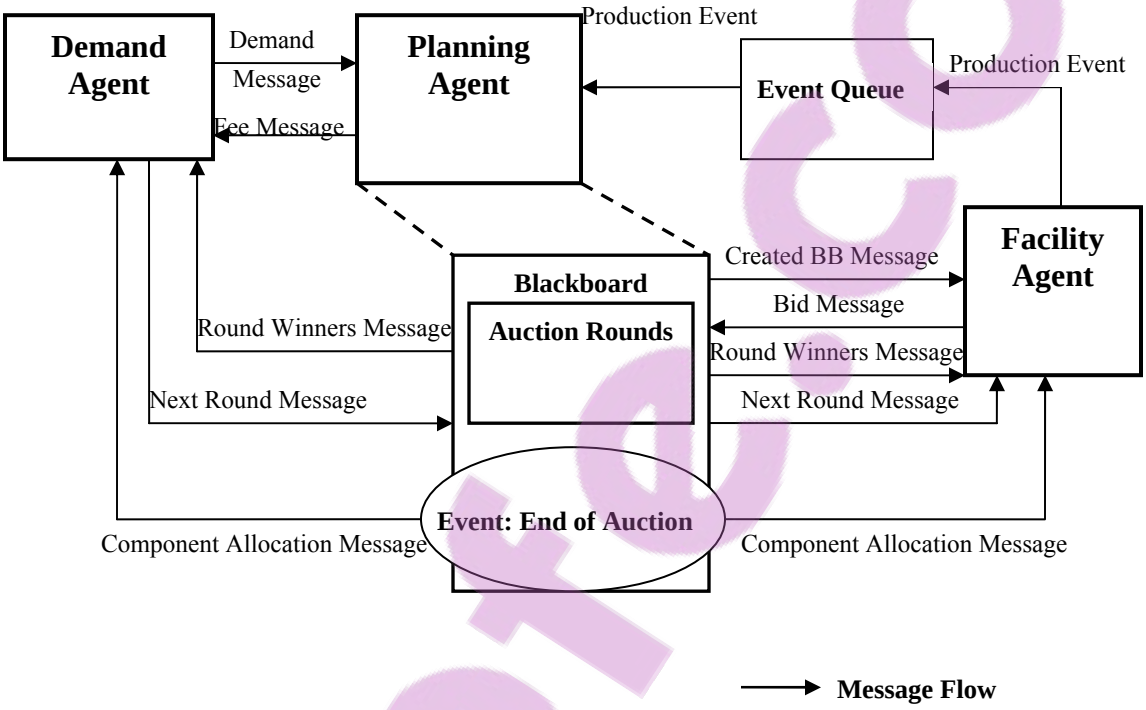

Figure 4.11: The Multi-Agent System with Message Flows

4.5 Accommodation of Changes in Demand Forecast during Planning Cycle

Assuming that at any time period before completion of the current planning period, we get a message from the Demand Agent that demand for a particular product has changed (decreased). The Facilitator Agent informs Facility Agents that demand has changed and they are requested to follow the original production plan only till the end of their next *production cycle*. The facilities know the status of their produced quantities and inventories at the end of their production cycles. The Facilitator revises the production targets of components related to the particular end product and calls for bids from Facility

Agents based on the new demand projection. The Facility Agents bid for the reduced production target as before and a new plan is thus formed. The new plan is communicated to the Facility Agents, so that they can schedule their production accordingly from the start of their next production cycle.

Priority based planning for multiple products:

We will assume three levels of priority between orders

- Low
- Normal
- High

Facilities can pre-empt production of lower priority jobs for carrying out production of higher priority jobs. Using this mechanism, the capacity reporting by a facility becomes dynamic and depends on the priority of the order of which the auctioned components are a part of. The higher the priority, it is expected, the higher would be the production capacity reported. Also, the facilities expect a higher pay-off for higher priority jobs carried out by them. When we carry out auction for a collection of different components, they are segregated based on their priority. The High priority jobs are auctioned first, then the Normal and lastly the Low priority jobs. It is assumed by doing this; higher priority jobs get capacity preference over lower priority jobs. To consider the case of demand change in this scenario, if priority of incoming demand is higher then currently executing plan on a facility, a facility can report available capacity at a higher price to accommodate the higher priority jobs. Though, if the priorities of incoming jobs is the same or lower than the currently executing jobs, there is no incentive to the facilities to report extra available capacity. So, it is theoretically possible that low priority jobs can be starved for resources in the system, if there are no breaks in the planned schedule.

This auction process is run carried out to determine production quantities at various facilities. The auction process quickly runs through the entire planning period (weeks/months) in a matter of minutes just to create a master production plan for the

entire planning period. We assume the demand function updates current demand daily to have an impact on the master production plan. As such, it is not possible for the demand to change during the time this auction process is being run.

To summarize, we model the problem as a multi-agent system having the three main agent types. The *Demand Agent* which is in-charge of periodic demand. The *Facilitator(Planning) Agent* which carries out the auction on behalf of the Demand Agent and makes sure that all auction rules are satisfied by all the parties. There is a collection of *Facility Agents* which submit proposals for carrying out production activities in response to auction supervised by the Facilitator Agent.

The Demand Agent submits the overall demand for various products in the particular period to the Facilitator Agent. The Facilitator Agent requests interest messages from various Facility Agents. Based on response of the Facility Agents the Facilitator Agents determines the feasibility of the auction. If the auction is feasible the Facilitator Agent carries out the auction based on auction rules and desired parameter values supplied by the Demand Agent and Facility Agents. Once the auction is finished, the winning agents as well as the Demand Agent are notified and the plan thus generated becomes the production plan for the involved production facilities for that particular period.

Level I auction starts at the start of the planning period. In Level I auction bids are invited for all the items which can be produced independently, i.e., they have no predecessor items, as specified in product BOMs. The quantities of various components desired at this level are according to the forecasted demand of various products.

In Figure 4.12 we present the high-level system architecture in terms of software component functionalities as well as how they are linked with each other. We also show the components of system which are not yet developed but are needed for integration with other modules of the APS system such as the scheduling module. (dashed outline)

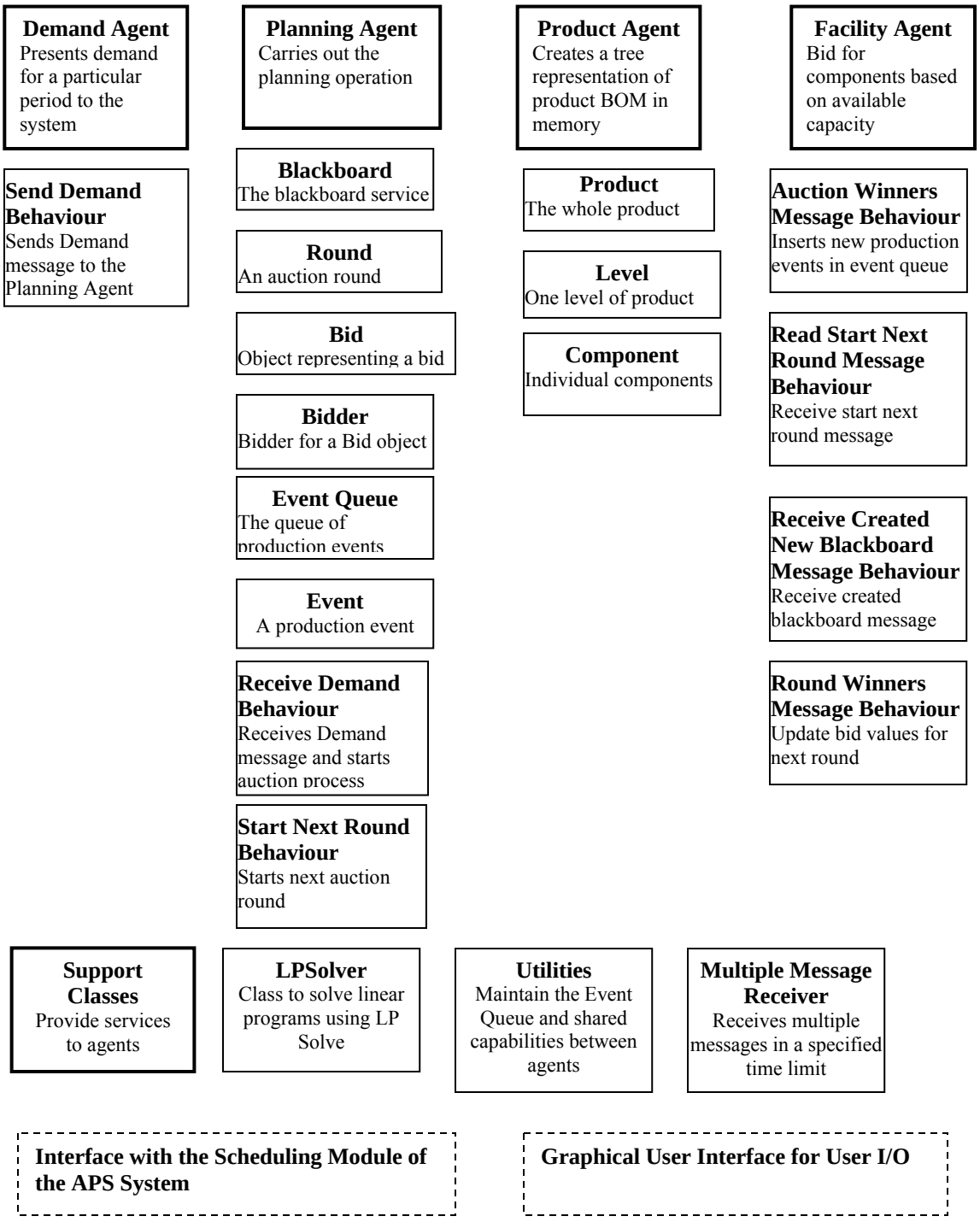

Figure 4.12: Software Architecture of the Multi-Agent System

5 Results and Discussion

The proposed auction based production planning methodology was validated in a two step process. First, we validated the proposed iterative combinatorial auction framework to be an efficient way to allocate multiple commodities to a collection of sellers given a multi-commodity demand presented by a seller. Once the auction mechanism was shown to be an efficient as well as near optimal method of allocation, we extended the simulation environment constructed earlier to validate the efficiency of the bidding mechanism in the multi-facility, multi-product production planning environment.

5.1 Validation of the Auction Mechanism

To validate our auction mechanism, we tested two main characteristics of any auction mechanism: efficiency and optimality. An auction is said to be efficient if the buyer always buys the object from the bidder (seller) who has the best valuation (lowest cost) for it, as long as that price is better then seller's reserve price (if any) (adapted from Krishna [34]). Reserve price is the highest price at which the buyer is ready to buy the object. An optimal auction is one which maximizes the overall objective function for that auction. As given by equation (4.10), this incorporates the value obtained by all the actors involved and not just that of the sellers.

It is very easy to determine the efficiency of an auction in the case of single object auctions. In the multiple commodity, multiple quantities case that we are dealing with, however, overlapping demand functions among the various sellers can make it difficult to find the auction efficiency. Our efficiency measure is similar to the one used in (Jones and Koehler [30]), which is comparable to the efficiency measures used in other

combinatorial auction studies. This efficiency measure is formally defined as a percentage that represents the closeness of the results to the optimal allocation (100% efficient). This optimal allocation is achieved by optimizing the linear model given by equations (5.1) - (5.4), which in turn is a simplified version (BOM has a single component, no precedence) of the multi-facility, multi-product production planning presented in chapter 4.

$$\text{Max.} \sum_{i}^{N} (p_i^{\,r} * Q_i) \quad - \quad \sum_{j}^{F} \left(\sum_{i}^{N} C_{ij}(q_{ij}) \right) \qquad \ldots(5.1)$$

s.t.

$$\sum_{j}^{F} q_{ij} \quad = \quad Q_i \quad \forall i = 1,2,..N \quad \text{<Quantity required is satisfied for each component>} \quad \ldots(5.2)$$

$$\sum_{i}^{N} q_{ij} \quad \leq \quad A_j \quad \forall j = 1,2,..F \quad \text{<Production capacity is satisfied for each seller>} \quad \ldots(5.3)$$

$$L_{ij} \quad \leq \quad q_{ij} \quad \leq \quad U_{ij} \quad \forall i = 1,2,..V \; ; \; j = 1,2,..N \quad \text{<Batch sizes are satisfied>} \qquad \ldots(5.4)$$

Where, again C_{ij} is the cost function for facility j for component i, based on quantity q_{ij}. As it is not possible for a centralized optimizer to know actual values of cost functions of autonomous sellers, the solution to this linear program provides us an upper bound for auction performance.

If we assume that $q_{ij}(A)$ is the allocation obtained by our auction framework, while q_{ij}^* is the allocation associated with the optimal solution, then the auction efficiency is given by:

$$E = 1 - \frac{\sum | q_{ij}(A) - q_{ij}^{*} |)}{\sum Q_i} * 100 \qquad \ldots(5.5)$$

and optimality is defined as:

$$O = \frac{\text{(Value of objective function for auction mechanism)}}{\text{Optimal value of objective function}} \qquad \ldots(5.6)$$

Jones and Koehler [30] discuss three specific types of bidders, as identified by (Bapna et. al. [8]): Participators, Evaluators and Opportunists. Participators bid actively throughout the auction while Evaluators place a single bid representing their best price early in the auction and Opportunists enter just before the auction's close to seek bargains. In our simulation studies we have also used three different types of bidding agents: Slow, Random and Fast. The relationship between these two taxonomies will be explained below.

Our Slow bidding agents reduce their bids by the lowest allowable amount between auction rounds. They are similar to risk-averse agents, as given in (Attri et. al. 2004), whose objective is to win the bid at highest price possible, even if there is a possibility that they might not win the bid. The Random bidders pick a random fraction of their slack as the bid amount (see Section 4 for definition of slack). Finally, the Fast bidding agents present their best bid offer in the second round. They are similar to the risk-neutral agents described in (Attri et. al. [2]), whose objective is to win the bid at any cost greater than or equal to their minimum acceptable bid amount.

Our Random and Slow Producer agents are examples of Participator bidders, and the Fast Producers act as Evaluators. Due to lack of pre-knowledge about auction completion, having Opportunist bidders in our auction is not possible. All our agents are

AllAdjustors as defined in (Jones and Koehler [30]) because they are free to adjust both price and quantity across auction rounds.

Our preliminary testing began with running a series of simulated auctions, within which we varied the mix of different agent types in order to identify the most appropriate combination of agents with respect to both optimality and efficiency. Using the most effective combination of agents, as discussed below, we then ran a series of additional tests to examine the impact of different auction parameters on system performance. The results of this testing are provided below.

The multi-agent simulator used to validate the proposed auction mechanism was developed using the JADE toolkit (JADE [28]) for creating agents with the Java programming language. The system allows characteristics of the agents, and of the auction it self, to be stored in separate properties files, and thus provides a great deal of flexibility with respect to the number and type of agents being simulated, as well as the nature of their interactions. In order to consider different auction scenarios, with different fee structures and different numbers and types of producers, only these properties files need be changed.

For our preliminary testing we used a single buyer, with a fixed set of requirements, and generated nine seller agents that compete to satisfy the stated demand. The sellers had adequate capacity to meet whole of the buyer's demand but they do so through some combination of seller contributions. For the first part of the testing, we ran the auction under three different combinations of Random, Slow, and Fast Producers: (6 Random, 3 Slow, 0 Fast); (9 Random, 0 Slow, 0 Fast); and (6 Random, 0 Slow, 3 Fast).

All producers were identical across all scenarios except for their bidding type. As shown in Table 5.1, the scenarios with only Random agents gave the best performance.

We then undertook a series of tests to determine good values for three of the primary auction parameters: minimum decrement price, round fee, and attendance fee. As shown in Table 5.1, it is clear that higher values for the minimum decrement price are better than lower values, as at lower values the prices decrease less rapidly across auction rounds and the buyer stops the auction well before the equilibrium prices are reached. If the decrement amount is too high, however, then the agents may stop bidding well before the equilibrium prices are achieved. Based on this behavior we determined that a good value for the minimum decrement is between 5% and 10% of the buyer's reserve price. We should add an important note here that, these values obtained are highly application specific and we should re-calibrate the system for every new application we put it in.

Lower values for the round fee will lead to more auction rounds and increased opportunity for optimality, since the resulting prices will be closer to equilibrium values. Too small of a round fee, however, may lead to extremely long auctions. In contrast, higher round fees will tend to cause the buyer to stop the auction before equilibrium prices are achieved, and thus auction performance will be sub-optimal. We therefore suggest fixing a round fee value as small as practically possible.

The results given in Table 5.1 also indicate that increasing the attendance fee actually improves performance. This behavior is surprising as one would expect similar results to those associated with an increasing round fee. Also, the number of rounds first increased which is counter-intuitive and then decreased. A possible explanation for this behavior is that at very low values, the attendance fee has very little effect and the round

fee is therefore the only driving factor. As the attendance fee is increased, the impact of the round fee diminishes and the buyer attempts a higher number of rounds, but any impact on performance is absorbed by the higher attendance fee buyer is now paying. As the attendance fee is increased further, the buyer offsets its increasing value by decreasing the number of rounds to save on round fee. Additional testing is necessary in order to further clarify this behavior.

Table 5.1: Auction Framework Simulation Test Results

Agent Type	Efficiency (%)	Optimality (%)	# Rounds
6 Random, 3 Slow	66.11	84.25	19.6
6 Random, 3 Fast	56.94	92.326	8
9 Random	89.8	92.145	9.5

Minimum Decrement	Efficiency (%)	Optimality (%)	# Rounds
1 % of Reserve Price	72.2	90.293	8.75
5 % of Reserve Price	89.8	92.326	9.5
10 % of Reserve Price	90.97	91.198	10.25

Round Fee	Efficiency (%)	Optimality (%)	# Rounds
5 % of Reserve Price	88.9	94.03	10
10 % of Reserve Price	89.8	92.145	9.5
20 % of Reserve Price	75.9	87.18	8

Attendance Fee	Efficiency (%)	Optimality (%)	# Rounds
10 % of Reserve Price	87.9	95.43	10.33
25 % of Reserve Price	86.1	93.639	12.33
50 % of Reserve Price	89.8	92.145	9.5

5.2 Validation of the Production Planning Approach

The agent model, as described in previous section was extended to conform to the agent model for the planning level as shown in Figure 4.7. The agent model consists of a Demand agent, a Planning (Facilitator) agent, a Product agent, a number of Facility agents and an Inventory agent associated with each facility agent. The agent model is constructed as an extension of the JADE agent API, so that a developer familiar with the

JADE API can later use the code libraries we developed to create a fully functioning planning module of an APS software system.

The planning software takes its inputs from various properties files as explained in the Appendix A. Below we explain the working of the software through an example.

To understand the operation of the software application, we would explain how to input a sample planning problem to the software and the output generated by the planning software. Consider the product, P1 which has exactly the same BOM precedence structure as Product A in fig 4.8. To keep our example simple so that we can just focus on the operation of the planning software, we would consider demand based on only one multi-level product and one production facility.

Assuming we require ten units of P1 over our planning period. The *demand.properties* file would have the following contents

```
Products = P1
Quantities = 10
```

The P1.properties file would have the following contents

```
Levels = 4
Level1 = T1,T2,T5
Level2 = T3
Level3 = T4
Level4 = T6
# Component = Quantity required for next stage, Moving Average of price, sub-component list
T1 = 2, 1000
T2 = 3, 990
T3 = 1, 980, T1, T2
T4 = 2, 1000,T3
T5 = 1, 1100
T6 = 1, 880, T4, T5
```

A sample facility, Facility1 which can carry out this production activity feasibly can be modeled through the file Facility1.properties

```
Components = T1, T2, T3, T4, T5, T6
```

```
#Component = Min. batch size, Max. batch size, Variable Cost, Fixed Cost, Capacity
Consumption/unit, Setup time for batch, per unit time needed
T1 = 5, 40, 900, 100,0.8, 2, 1
T2 = 5, 70, 950, 100,1.3, 5, 2
T3 = 5, 20, 850, 100,0.7, 3, 1
T4 = 5, 20, 850, 100,0.7, 3, 1
T5 = 5, 15, 850, 100,0.8, 3, 1
T6 = 5, 10, 850, 100,0.7, 3, 1
#Facility Type, S:Slow, R:Random, F:Fast
Type=R
#Total Capacity in aggregate units
Capacity = 153
```

We don't need to change the default properties file for planning agent. To run the agents and create the master production plan, we start the JADE agent container and create instances of Demand, Planning and Facility agents through the following command on DOS prompt:

```
java jade.Boot -nomtp DemandAgent:MultiPlanner.DemandAgent
PlanningAgent:MultiPlanner.PlanningAgent Facility1:MultiPlanner.FacilityAgent(Facility1)
```

Other support agents such as Product and Inventory agents are automatically created by the application and we don't need to specify them at the DOS prompt. If we have configured the properties files correctly we should get the following output.

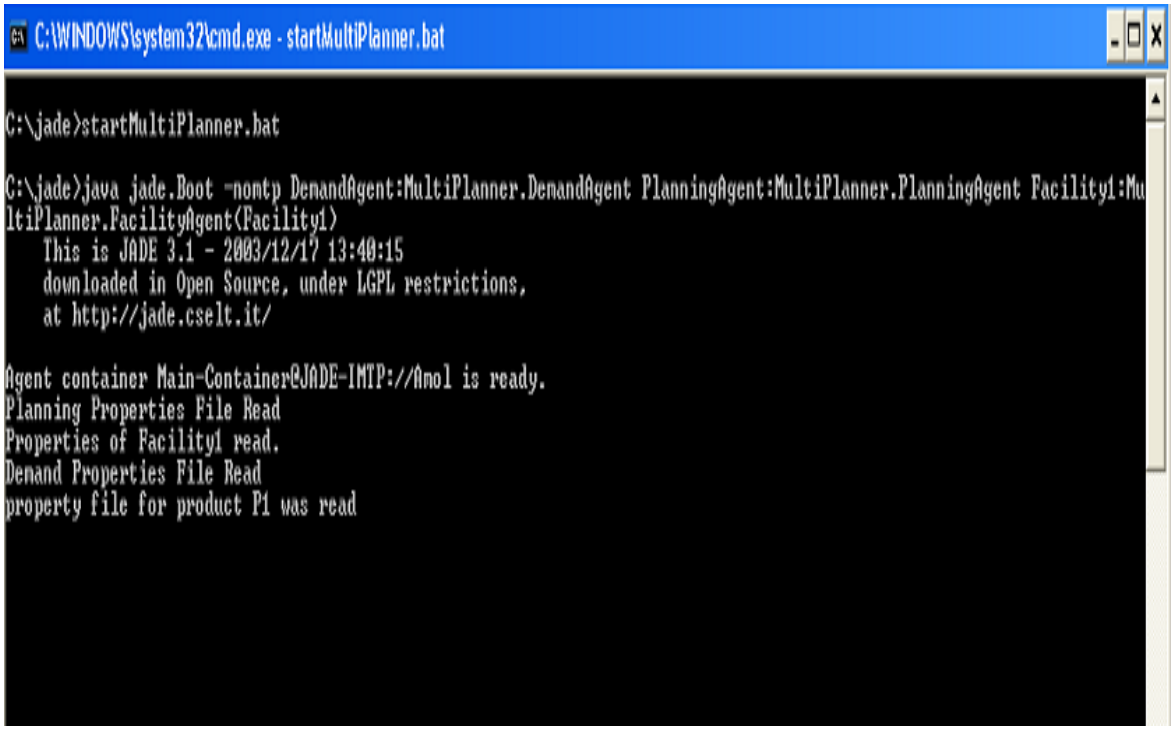

Figure 5.1: Sample Output Screen (1)

The screen pauses for sometime at this point as auction process is started by the system. The screen scrolls for some time as the results of winner determination LP program are displayed after each auction round.

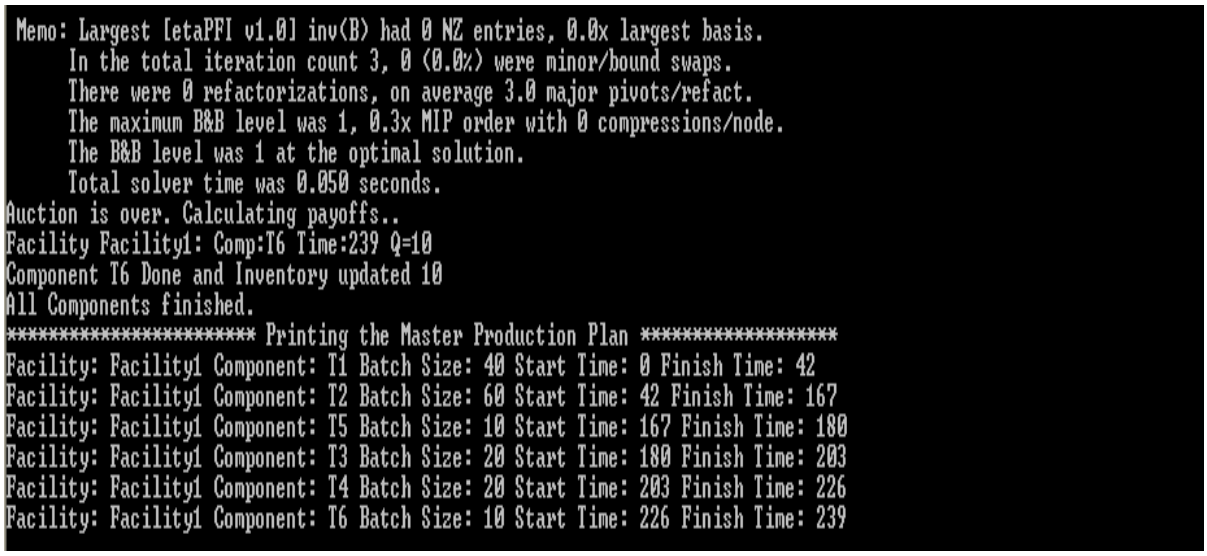

Figure 5.2: Sample Output Screen (2)

Once all auctions are over and all components have been allocated to various facilities (as we have only one facility, all components are allocated to it), the software prints out the overall master production plan on the DOS screen as well as the log.txt file in log directory of the software (See Appendix B for directory details).

Same output is also saved in the log.txt file for facilitating integration with other software applications such as the scheduling module of the APS suite.

To validate the production plan generated by our agent-based model of the multi-facility problem we use the similar approach to the one used to validate the iterative auction mechanism. We would again measure efficiency and optimality of our planning approach as compared to an optimal planning approach based on the mathematical model of the multi-facility production planning problem as discussed in section 4.3.

If we assume that $q_{ij}(A)$ is the allocation to a specific production facility obtained by our agent-based planning approach, while q_{ij}^* is the allocation to the same facility associated with the optimal solution, then the plan efficiency is given by:

$$E = 1 - \frac{\sum | q_{ij}(A) - q_{ij}^* |)}{\sum Q_i} * 100 \qquad \ldots(5.7)$$

and plan optimality is defined as:

$$O = \frac{\text{(Value of objective function for agent - based approach)}}{\text{Optimal value of objective function}} \qquad \ldots(5.8)$$

Where, objective function is minimization of total cost as explained in section 4.3.

We use the same auction parameter values we determined during the validation process in Section 5.1. We would validate the agent-based methodology against the optimal approach using five different production scenarios with increasing level of complexity in scheduling of production with the intent of showing that the performance

of the proposed methodology does not suffer from increase in complexity of the problem. The results of the validation process are presented in Table 5.2.

Table 5.2: Agent-Based Production Planning Simulation Test Results

Production Scenario	Efficiency (%)	Optimality (%)
One Product, Multiple Levels, One Production Facility	100.00	89.46
One Product, Multiple Levels, Three Production Facilities	91.56	88.43
Three Products, Multiple Levels, Three Production Facilities	84.32	87.53
Five Products, Multiple Levels, Three Production Facilities	81.56	86.69
Five Products, Multiple Levels, Five Production Facilities	79.45	84.2

6 Conclusions and Future Work

An integrated approach is required to solve production planning problem in medium and large scale manufacturing enterprises having distributed production facilities. The need for cross-facility capacity management is more evident in high-tech industries having capital-intensive equipment and short technology life cycle. There have been solutions proposed in the literature for solving complex multi-facility, multi-product planning problems by decomposing and approximating them as either multiple single product or single facility problems. The results obtained by this research study demonstrate that novel distributed computing concepts such as autonomous intelligent agents and negotiation mechanisms such as iterative combinatorial auctions can be used to develop solution methodologies which can solve the integrated problem in real-time.

A generic architecture of an agent-based modeling approach is presented which can be used to efficiently solve complex distributed problems. This generic architecture is adapted to production planning and scheduling domain to create an agent-based integrated framework to implement an Advanced Planning and Scheduling (APS) system geared towards enterprises having multiple production facilities. An iterative combinatorial auction mechanism is introduced and discussed as a negotiation protocol suited for this environment. These concepts are then used to develop a software library and a planning tool which can be used to create master production plans for all production facilities and which can be seamlessly integrated with other software solutions such as a scheduling tool.

The validation process carried out on both the iterative combinatorial framework and the agent-based production planning methodology, demonstrate that the proposed solution strategies can be used for integrated decision making in the multi-product, multi-facility production planning domain. Also, the software tool developed as part of this research is a robust, platform independent tool which can be used by manufacturing enterprises to make production planning decisions.

We list below further work that will help to increase the applicability of the proposed production planning approach includes the following:

- **Extending the auction mechanism to consider a greater number of criteria**: Currently, the auction mechanism considers cost of production, batch size and processing time as the criteria for making allocation decisions. Simultaneous consideration of these multi-lateral decision criteria leads to a better modeling of the negotiation process, though it makes the solution mechanism more complex with addition of each new criterion. It is desirable to add quality of components produced by a facility as a criterion in future as it is also an important decision making parameter which is used frequently.
- **Extending the planning mechanism to consider transportation costs**: In the current solution architecture transportation costs between production facilities are neglected and are assumed to be the same. In the real-world, transportation costs of semi-finished components and material from one facility to another can have a major impact on production allocation decision for various facilities. Thus, it is desirable to extend the solution architecture by considering transportation costs. This can be done by extending the agent model by adding a Transportation agent which becomes part of the negotiation process by adding transportation constraints to the process.
- **Adding some degree of look-ahead to agents:** The way agents carry out the bidding process right now is to bid to the maximum level possible for currently available components. By adding look-ahead to the agents, they can save their capacities for some important components which are still to come and improve the system efficiency. The drawback of adding look-ahead is that system loses its reactive capability to accommodate sudden changes. So, a trade-off between look-ahead and reactive capability will have to be determined.
- **Development of theoretical support of the methodology used:** In this thesis we have developed and demonstrated experimentally the effectiveness of the agent-based methodology. However, it would be worthwhile to model the entire agent-based decision-making process proposed here and develop a theoretical support of its effectiveness.

7 Summary and Recommendations

The output of this research are an agent-based framework for creation an Advanced Planning and Scheduling (APS) system, an iterative combinatorial auction mechanism which can be used for multi-lateral negotiations in real-time, and a software library which implements these new concepts to create an agent-based production planning tool. The developed software tool has the capability to accept production data for various production facilities part of a manufacturing enterprise, BOM precedence data for various products, adjusting of auction parameters and provides output in form of a master production plan which can be both displayed on the screen to the user and stored in a file so that it can be easily integrated with other software applications. The library is created using open-source components ands can be extended by adding more classes to it.

This work is part of an initiative to develop a robust and exhaustive suite of software applications which can be used to seamlessly integrate decision-making processes of a manufacturing enterprise and implement an APS system. The production planning solution developed through this research can be used as a standalone tool though it is expected to be integrated with other modules of the APS system later to create the overall APS application.

Appendix A: Software Requirements and Properties Files

Software requirements:

1. JRE 1.4.2 or higher, http://java.sun.com/j2se/1.4.2/download.html
2. JADE 3.1 or higher, http://jade.cselt.it/
3. Log4j 1.2 or higher, http://logging.apache.org/log4j/docs/download.html
4. JDOM 1.0 or higher, http://www.jdom.org/downloads/
5. LP Solve 5.0 or higher, http://groups.yahoo.com/group/lp_solve/files/ (registration required, free)
6. RngPack 1.1 or higher, http://www.honeylocust.com/RngPack/

It is assumed that the user will be using the API over a Microsoft Windows 9.x (or higher) environment. The above libraries are needed to be installed before *MultiPlanner* API can be installed. To install the API, copy API files provided into MultiPlanner directory inside class directory of JADE. You will have to create a new MultiPlanner directory if this is a fresh install. If the CLASSPATH variable of Windows is set correctly, the API should be usable after this step.

Sample Properties Files:

Properties files are needed to be included for Demand agent, Planning agent, one each for each Facility agent and each product in the system. A few sample files are provided below to help user build their own properties files.

ProductA.properties (Corresponding to the BOM structure of Product A in figure 4.8)

```
Levels = 4
Level1 = T1,T2,T5
Level2 = T3
Level3 = T4
Level4 = T6
# Component = Quantity required for next stage, Moving Average of price, sub-component list
```

T1 = 2, 1000

T2 = 3, 990

T3 = 1, 980, T1, T2

T4 = 2, 1000,T3

T5 = 1, 1100

T6 = 1, 880, T4, T5

Log.properties (Needed to configure the logging system)

```
# An example log4j configuration file that outputs both to System.out
# and a file named 'test'.
# For the general syntax of property based configuration files see the
# documenation of org.apache.log4j.PropertyConfigurator.
# WARNING: Location information can be useful but is very costly in
# terms of computation.
# The root logger uses the appender called A1,A2.
log4j.rootLogger=, A1, A2
# A1 is set to be ConsoleAppender sending its output to System.out
log4j.appender.A1=org.apache.log4j.ConsoleAppender
# A1 uses PatternLayout.
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
# The conversion pattern consists of date in ISO8601 format, level,
# thread name, logger name truncated to its rightmost two components
# and left justified to 17 characters, location information consisting
# of file name (padded to 13 characters) and line number, nested
# diagnostic context, the and the application supplied message
#log4j.appender.A1.layout.ConversionPattern=%d %-5p [%t] %-17c{2} (%13F:%L) %3x - %m%n
log4j.appender.A1.layout.ConversionPattern=[%t] %-17c{2} (%13F:%L) %3x - %m%n
# Appender A2 writes to the file "log.txt".
log.home=C:/jade/classes/MultiPlanner/log
log4j.appender.A2=org.apache.log4j.FileAppender
log4j.appender.A2.File=${log.home}/log.txt
# Truncate 'test' if it aleady exists.
log4j.appender.A2.Append=true
# Appender A2 uses the PatternLayout.
log4j.appender.A2.layout=org.apache.log4j.PatternLayout
log4j.appender.A2.layout.ConversionPattern=[%t] %c{2} - %m%n
```

```
# In this example, we are not interested in INNER loop or SWAP
# messages.  You might try to set INNER and SWAP to DEBUG for more
# verbose output.
#log4j.logger.org.apache.log4j.examples.SortAlgo.INNER=INFO
#log4j.logger.org.apache.log4j.examples.SortAlgo.SWAP=INFO
```

Demand.properties (To set demand for a planning period)

```
Products = P1, P2, P3, P4,P5
Quantities = 20, 20, 20, 15,10
Reserve_Price = 1000, 1000, 1000, 1000, 1000,1000
Start_Price = 1000, 1200, 1300, 1400, 1500,1200
```

Planner.properties (To configure Planning agent)

```
Attendence_Fee = 500
Round_Fee = 100
# The percentage of average price start price is
Start_Price = 150
```

Facilityx.properties (To configure facility x)

```
Components = T1, T2, T3
#Component = Min. batch size, Max. batch size, Variable Cost, Fixed Cost, Capacity
Consumption/unit, Setup time for batch, per unit time needed
T1 = 5, 20, 900, 100,0.8, 2, 1
T2 = 5, 20, 950, 100,1.3, 5, 2
T3 = 5, 20, 850, 100,0.7, 3, 1
#Facility Type, S:Slow, R:Random, F:Fast
Type=R
#Total Capacity in aggregate units
Capacity = 20
```

Appendix B: Structure of *MultiPlanner* Production Planning Software

The structure of the software application is as follows:

jade_class is assumed to be the location of the classes directory of JADE kit on the user machine.

jade_class /MultiPlanner

AuctionWinnersMessageBehaviour.class
Bid.class
Bidder.class
Blackboard.class
Component.class
DemandAgent.class
Event.class
EventQueue.class
FacilityAgent.class
LPSolver.class
Level.class
MultipleMessageReceiver.class
PlanningAgent.class
Product.class
ProductAgent.class
ReadStartNextRoundMessageBehaviour.class
ReceiveBidMessage.class
ReceiveCreatedNewBlackboardMessageBehaviour.class
ReceiveDemandBehaviour.class
ReceiveFeeBehaviour.class
ReceiveWinnersMessageBehaviour.class
Round.class
RoundWinnersMessageBehaviour.class
SendDemandBehaviour.class
StartNextRoundBehaviour.class
Utilities.class
WaitBehaviour.class

WaitPlanningBehaviour.class

jade_class /MultiPlanner/Properties

Demand.properties

Facilityx.properties [set of user defined files]

Productx.properties [set of user defined files]

Planner.properties

log.properties

jade_class /MultiPlanner/log

log.txt

The startMultiPlanner.bat file can be used to start the MultiPlanner software application. The actual contents of the file depend on number and names of facilities being used in planning by the user. The general structure of the file is

java jade.Boot -nomtp DemandAgent:MultiPlanner.DemandAgent

PlanningAgent:MultiPlanner.PlanningAgent

FacilityName:MultiPlanner.FacilityAgent(FacilityName)

where, *FacilityName* is the unique name of the facility, this line should be repeated for each production facility. It is also assumed that the properties file of the facility would have the same name.

References

[1] Amstel, P. v. (1998). Snel, Sneller, Snelst, APS-systeem schiet logistiek manager te hulp. *Tijdscrift voor Inkoop & Logistiek, 5*, 18 - 23.

[2] Attri, H.; Goel, A., Chen, F.F., and Sarin, S.C. (2004). Proxy-Bidding Strategies for Intelligent Agent Negotiations, In, *Advanced Simulation Techniques, Business and Industry Symposium*, Arlington, VA, 187 - 192.

[3] Ausubel, L. M. (2002). An Efficient Dynamic Auction for Heterogeneous Commodities (Working Paper), University of Maryland.

[4] Ausubel, L. M., & Cramton, P. (2004). Auctioning Many Divisible Goods. *Journal of the European Economic Association (Forthcoming), 1*.

[5] Ausubel, L. M., & Cramton, P. (2004). Auctioning Many Divisible Goods. *Journal of the European Economic Association, 2*, 480 - 493.

[6] Baker, A. D. (1996). *Metaphor or Reality: A Case Study where Agents bid with Actual Costs to Schedule a Factory*, Edition. Market-Based Control: A Paradigm for Distributed Resource Allocation. Chapter Volume Clearwater, S. H.: World Scientific. 184 - 223.

[7] Baker, A. D., Parunak, H. V. D., & Ero, K. (1999). Agents and the Internet: Infrastructure for Mass Customization. *IEEE Internet Computing, 3*(5), 62 - 69.

[8] Bapna, R., Goes, P., and Gupta, A (2000). A Theoretical and Empirical investigation of Multi-Item On-Line Auctions. *Information Technology and Management*, 1, 1, 1 - 23.

[9] Barbuceanu, M., & Fox, M. S. (1995). The Architecture of an Agent Based Infrastructure for Agile Manufacturing. Proceedings of IJCAI'95 Workshop on Intelligent Manufacturing. Montreal, Canada.

[10] Barbuceanu, M., & Fox, M. S. (1996). *The Architecture of an Agent Building Shell*, Edition. Intelligent Agents II, LNAI 1037. Chapter Volume Woolridge, M., Muller, J. P., & Tambe, M.: Springer. 235 - 250.

[11] Bhatnagar, R. (1995). Multi-Plant Coordination: Towards Improved Manufacturing Performance. Engineering Management Conference**:** 396 - 401.

[12] Bhatnagar, R., & Chandra, P. (1993). Models for Multi-Plant Coordination. *European Journal of Production Research, 67*, 141 - 160.

[13] Brandolese, A., Brun, A., & Portioli-Staudacher, A. (2000). A Multi-Agent Approach for the Capacity Allocation Problem. *International Journal of Production Economics, 66*, 269 - 285.

[14] Butle, J., & Ohtsubo, H. (1992). *ADDYMS: Architecture for Distributed Dynamic Manufacturing Scheduling*, Edition. Artificial Intelligence Applications in Manufacturing. Chapter Volume Famili, A., Nau, D. S., & Kim, S. H.: The AAAI Press. 199 - 214.

[15] Chalmeta, R., Campos, C., & Grangel, R. (2001). References architectures for enterprise integration. *The Journal of Systems and Software, 57*, 175 - 191.

[16] Chen, D., & Vernadat, F. (2004). Standards on Enterprise Integration and Engineering - State of the Art. *Int. Journal of Computer Integrated Manufacturing, 17*(3), 235 - 253.

[17] Clearwater, S. H. (1996). *Market-Based Control: A Paradigm for Distributed Resource Allocation*: World Scientific.

[18] Dhaenens-Flipo, C., & Finke, G. (2001). An Integrated Model for An Industrial Production-Distribution Problem. *IIE Transactions, 33*, 705 - 715.

[19] Eck, M. v. (2003). Is Logistics Everything? A Research on the Use of Advanced Planning and Scheduling Systems. PhD Thesis, Universiteit Amsterdam,

[20] Ertogral, K., & Wu, S. D. (2000). Auction-Theoretic Coordination of Production Planning in the Supply Chain. *IIE Transactions,*(32), 931 - 940.

[21] FIPA. (2004). Retrieved from http://www.fipa.org.

[22] Fischer, K., Muller, J. P., Heiming, I., & Scheer, A. (1996). Intelligent Agents in Virtual Enterprises. In Proceedings of the First International Conference and Exhibition on the Practical Applications of Intelligent Agents and Multi-Agent Technology. London, U.K.**:** 205 - 224.

[23] Forrester, J. W. (1958). *Industrial Dynamic*: M.I.T. Press, Cambridge, MA.

[24] Gasser, L. (1992). *An Overview of DAI*, Edition. Distributed Artificial Intelligence: Theory and Praxis. Chapter Volume Avouris, N. M., & Gasser, L.: Kluwer Academic Publishers. 9 - 30.

[25] Gjerdrum, J., Shah, N., & Papageorgiou, L. G. (2001). A Combined Optimization and Agent-Based Approach to Supply Chain Modelling and Performance Assessment. *Production Planning and Control, 12*(1), 81 - 88.

[26] Goel, A., Chen, F. F., Attri, H., & Sarin, S. C. (2004). A Generic Multi-Agent System for Manufacturing Enterprise Integration. Proceedings of the 14th

International Conference on Flexible Automation and Intelligent Manufacturing. Toronto, Canada.

[27] Hartvany, J. (1985). intelligence and Cooperation in Heterarchic Manufacturing Systems. *Robotics and Computer Integrated Manufacturing, 2*(2), 101 - 104.

[28] JADE Agent Toolkit. (2004).

[29] Jain, A., Aparicio, M., & Singh, M. P. (1999). Agents for Process Coherence in Virtual Enterprises. *Communications of the ACM, 42*(3), 62 - 69.

[30] Jones, J.L. and Koehler (2002), G.J. Combinatorial Auctions Using Rule-Based Bids. *Decision Support Systems*, 32, 1, 59 - 74.

[31] Kaihara, T. (2001). Supply Chain Management with Market Economics. *International Journal of Production Economics, 73*, 5 - 14.

[32] Kaihara, T. (2003). Multi-Agent Based Supply Chain Modelling with Dynamic Environment. *International Journal of Production Economics, 85*, 263 - 269.

[33] Klemperer, P. (2002). What Really Matters in Auction Design. *Journal of Economic Perspectives, 16*(1), 169 - 189.

[34] Krishna, V. (2002). *Auction Theory*: Elsevier Science (USA).

[35] Kutanoglu, E., & Wu, S. D. (1999). On Combinatorial Auction and Lagrangean Relaxation for Distributed Resource Scheduling. *IIE Transactions, 31*(9), 813 - 826.

[36] Lee, W. B., & Lau, H. C. W. (1999). Multi-agent modeling of dispersed manufacturing network. *Expert Systems with Applications, 16*, 297 - 306.

[37] Lee, Y. H., Jeong, C. S., & Moon, C. (2002). Advanced Planning and Sceduling with Outsourcing in Manufacturing Supply Chain. *Computers and Industrial Engineering, 43*, 351 - 374.

[38] Lee, Y. H., & Kim, S. H. (2002). Production-distribution planning in supply chain considering capacity constraints. *Computers and Industrial Engineering, 43*(1-2), 169 - 190.

[39] Lim, S. H., Juster, N., & Pennington, A. d. (1997). Enterprise modeling and integration: a taxonomy of seven key aspects. *Computers in Industry, 34*, 339 - 359.

[40] Lin, G. Y. (1993). A Distributed Production Control for Intelligent Manufacturing Systems. PhD Thesis, Industrial Engineering, Perdue University,

[41] Lin, G. Y., & Solberg, J. J. (1992). Integrated Shop Floor Control Using Autonomous Agents. *IIE Transactions: Design and Manufacturing, 24*(3), 57 - 71.

[42] Macchiaroli, R., & Riemma, S. (2002). A Negotiation Scheme for Autonomous Agents in Job Shop Scheduling. *International Journal of Computer Integrated Manufacturing, 15*(3), 222 - 232.

[43] Nigro, G. L., Diega, S. N. L., Perrone, G., & Renna, P. (2003). Coordination Policies to Support Decision Making in Distributed Production Planning. *Robotics and Computer Integrated Manufacturing, 19*, 521 - 531.

[44] Noori, H., & Mavaddat, F. (1998). Enterprise integration: issues and methods. *International Journal of Production Research, 36*(8), 2083 - 2097.

[45] Pan, J. Y. C., & Tenenbaum, J. M. (1991). An Intelligent Agent Framework for Enterprise Integration. *IEEE Transactions on Systems, Man and Cybernetics, 21*(6), 1391 - 1408.

[46] Pancerella, C. M., & Berry, N. M. (1999). Adding Intelligent Agents to Existing EI Frameworks. *IEEE Internet Computing*.

[47] Parkes, D. C. (2004). *Iterative Combinatorial Auctions*, Edition. Combinatorial Auctions (Forthcoming). Chapter Volume Cramton, P., Shoham, Y., & Steinberg, R.

[48] Parunak, H. V. D., Baker, A. D., & Clark, S. J. (1998). The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design. In Working Notes of the Agent-Based Manufacturing Workshop. Minneapolis, MN**:** 136 - 145.

[49] Patankar, A. K., & Adiga, S. (1995). Enterprise integration modeling: a review of theory and practice. *Computer Integrated Manufacturing Systems, 8*(1), 21 - 34.

[50] Pátkai, B., Keskinarkaus, J., Szaniszló, Z., & Torvinen, S. (2002). Agent-Based Simulation and Optimization of Production Networks. Proceedings of Mechanical Engineering Research – New Possibilities by Co-operation Seminar**:** 143-153.

[51] Peng, Y., Finin, T., Labrou, Y., Chu, B., Long, J., Tolone, W. J., & Boughannam, A. (1998). A Multi-Agent System for Enterprise Integration. Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-98).

[52] Petersen, S. A., Divitini, M., & Matskin, M. (2001). An Agent-Based Approach to Modelling Virtual Enterprises. *Production Planning and Control, 12*(3), 224 - 233.

[53] Sadeh, N. M., Hildum, D. W., Kjenstad, D., & Tseng, A. (2001). MASCOT: An Agent-Based Architecture for Dynamic Supply Chain Creation and Coordination in the Internet Economy. *Production Planning and Control, 12*(3), 212-223.

[54] Sandholm, T. (1998). Contract Types for Satisfying Task Allocation: Theoritical Results. Proceedings of AAAI 1998 Spring Symposium on Satisficing Models.

[55] Sandholm, T. (1999). *Distributed Rational Decision Making*, Edition. Multi-Agent System - A Modern Approach to Distributed Artificial Intelligence. Chapter Volume Weiss, E. G.: The MIT Press. 201 - 258.

[56] Sandholm, T., & Lesser, V. (1995). Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. Proceedings of the First International Conference on Multiagent Systems (ICMAS-95). San Francisco, CA**:** 328 - 335.

[57] Shen, W., Norrie, D. H., & Barthes, J.-P. A. (2000). *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*: Taylor and Francis, NY.

[58] Siwamogsatham, T., & Saygin, C. (2004). Auction-Based Distributed Scheduling and Control Scheme for Flexible manufacturing Systems. *International Journal of Production Research, 42*(3), 547 - 572.

[59] Smith, R. G., & Davis, R. (1981). Frameworks for Cooperation in Distributed Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics, 11*(1), 61 - 70.

[60] Tharumarajah, A. (2001). Survey of Resource Allocation Methods for Distributed Manufacturing Systems. *Production Planning & Control, 12*(1), 58 - 68.

[61] Tilley, K. J. (1996). *Machining Task Allocation in Discrete Manufacturing Systems*, Edition. Market-Based Control: A Paradigm for Distributed Resource Allocation. Chapter Volume Clearwater, S. H.: World Scientific. 224 - 252.

[62] Timpe, C. H., & Kallrath, J. (2000). Optimal Planning in Large Multi-Site Production Networks. *European Journal of Operational Research, 126*, 422 - 435.

[63] Turbide, D. (1998). Advanced Planning and Sceduling (APS) Systems. Midrange ERP Magazine. *Jan-Feb Issue*.

[64] Vernadat, F. B. (2002). Enterprise Modeling and Integration (EMI): Current status and perspectives. Annual Reviews in Control. *26***:** 15 - 25.

[65] Wagner, T., Guralnik, V., & Phelps, J. (2003). TAEMS Agents: Enabling Dynamic Distributed Supply Chain Management. *Electronic Commerce Research and Applications, 2*, 114 - 132.

[66] Walsh, W. E., Wellman, M. P., & Ygge, F. (2000). Combinatorial Auctions for Supply Chain Formation. 2nd ACM conference on Electronic commerce. Minneapolis, MS, ACM Press New York, NY**:** 260 - 269.

[67] Weiss, E. G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA: The MIT Press.

[68] Woolridge, M. J. (2001). *An Introduction to Multiagent Systems*. England, UK: John Wiley & Sons Ltd.

[69] Wu, S. D., & Golbasi, H. (1999). Manufacturing Planning over Alternative Facilities: Modeling, Analysis and Algorithms. Bethleham, PA, Manufacturing Logistics Institute, Dept. of IMSE, Lehigh University.

[70] Zhou, H. D. (2003). Design and Evaluation of a Co-Learning Multi-Agent System to Collaborative Supply Chain Management. Proceedings of 34th Annual Meetng of the Decision Sciences Institute. Washington, DC.

[71] Zhou, Q., Souben, P., & Besant, C. B. (1998). An Information System for Production Planning in Virtual Enterprises. *Computers and Industrial Engineering, 35*(1-2), 153 - 156.

Vita

The author (Amol Goel), son of J.K. Goel and Beena Goel, was born in Delhi, India. He attended schools in various Indian cities before joining Delhi Institute of Technology, Delhi University from where he received his B. Eng. in Manufacturing and Automation engineering in 2000. Amol won two university silver medals for ranking first in the department as well as best senior year project. After completion of his undergraduate studies, Amol worked with Mahindra-British Telecom Plc. (MBT) as a software engineer for two years, where he was involved in a couple of R&D projects in the field of mobile computing and internet applications. Amol left MBT for attending Virginia Tech to earn a Master of Science in Industrial and Systems Engineering. During his graduate studies at Virginia Tech, Amol was a Graduate Research Assistant for the Center for High Performance Manufacturing (CHPM, Grado Department of Industrial and Systems Engineering) as well as Dept. of Business Information Technology. Amol was also awarded the Ingersoll-Rand departmental scholarship for the year 2003-2004. Amol has both conference and journal publications in the field of multi-agent system applications in e-commerce and manufacturing management.