Multi-agent System Models for Distributed Services Scheduling

by

Farnaz Dargahi

A Thesis

in

The Department of Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

April 2014

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Farnaz Dargahi

Entitled: Multi-agent System Models for Distributed Services Scheduling

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Electrical & Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Adam Krzyzak	Chair
Dr. Jean-Marc Frayret	External Examiner
Dr. Onur Kuzgunkaya	External to Program
Dr. Shahin Hashtrudi Zad	Examiner
Dr. Wei-Ping Zhu	Examiner
Dr. Chun Wang	Thesis Co-Supervisor
	Thesis Co-Supervisor

Approved by

Chair of Department or Graduate Program Director

April, 2014

Abstract

Multi-agent System Models for Distributed Services Scheduling

Farnaz Dargahi, Ph.D.
Concordia University, 2014

This thesis investigates the computational and modeling issues involved with developing solutions for distributed service scheduling problems. Compared with traditional manufacturing scheduling, service scheduling poses additional challenges due to the significant customer involvement in service processes. The first challenge is that the service scheduling environment is a distributed environment in which scheduling-related information is scattered among individual identities, such as service providers and customers. The second challenge is that the service scheduling environment is a dynamic environment. Uncertainty in customer demand, customer cancellations and no-shows make the scheduling of services a complex dynamic process. Service scheduling has to be robust and prepared to accommodate any contingencies caused by customer involvement in service production. The third challenge concerns customers' private information. To compute optimal schedules, ideally, the scheduler should know the complete customer availability and preference information within the scheduling horizon. However, customers may act strategically to protect their private information. Therefore, service scheduling systems should be designed so that they are able to elicit enough of a customer's private information that will make it possible to compute high quality schedules. The fourth challenge is that in a service scheduling environment, the objectives are complicated and they may even be in opposition. The distributed service scheduling environment enables each agent to have their own scheduling objectives. The objectives of these agents can vary from one to another. In addition to multiple objectives, since agents are self-interested, they are likely to behave strategically to achieve their own objectives without considering the global objectives of the system. Existing approaches usually deal with only a part of the challenges in a specific service domain. There is a need for general problem formulations and solutions that address service scheduling challenges in a comprehensive framework.

In this thesis, I propose an integrated service scheduling framework for the general service scheduling problem. The proposed framework uses iterative auction as the base mechanism to tackle service scheduling challenges in distributed and dynamic environments. It accommodates customer's private information by providing appropriate incentives to customers and it has the potential to accommodate dynamic events. This framework integrates customers' preferences with the allocation of a provider's capacity through multilateral negotiation between the provider and its customers. The framework can accommodate both price-based commercial settings and non-commercial service settings. Theoretical and experimental results are developed to verify the effectiveness of the proposed framework. The application of the framework to the mass customization of services and to appointment scheduling are developed to demonstrate the applicability of the general framework to specific service domains. A web-based prototype is designed and implemented to evaluate the scalability of the approach in a distributed environment.

Acknowledgement

It is indeed a great pleasure for me to express my sincere appreciation and thanks to my respectable supervisor Dr. Chun Wang. I would like to thank him from the bottom of my heart for all his generous support. He has supported me not only by providing a research assistantship over almost four years, but also academically and emotionally through the rough road to finish this thesis. Thank you for all your contributions, remarkable/practical ideas, and encouragement.

I would also like to thank the members of my defence committee, Dr. Frayret, Dr. Hashtrudi Zad, Dr. Kuzgunkaya, and Dr. Zhu, for taking their valuable time to examine my thesis.

I owe my deepest gratitude to my husband Majid for his continuous support, his kind guidance and encouragement. Thank you for providing valuable input to my work and all your support.

My parents, Reza and Shahin, receive my deepest gratitude and love for their unconditional support through my entire life. This thesis is dedicated to them. I would like to thank my brothers Ali Reza and Hamid Reza for always supporting me during the various stages of my life.

Table of Content

List of Figures

List of Tables

Chapter 1 Introduction and Motivation

Service scheduling is a decision-making process which allocates limited service resources to service activities over time while satisfying certain constraints and optimizing one or more objectives. Service scheduling problems are common to many domains such as healthcare, transportation and computing. Compared to scheduling problems in manufacturing, service scheduling problems have unique characteristics. In manufacturing an activity usually transforms a physical component and adds value to it; resources are typically referred to as machines and the configuration of machines; objectives are typically a function of the completion times, due dates, and the deadlines of jobs (Pinedo, 2009). In service settings, an activity usually involves people. Examples include a meeting that has to be attended by certain individuals, a flight that transports passengers, an operation that has to be done by a surgeon on a given day. Services usually require both physical and human resources. In contrast to most manufacturing scheduling models, in service settings, additional factors such as personnel costs, customer waiting costs and customer preferences are often considered in the objective function.

The differences between manufacturing and service scheduling are mainly derived from the fundamental characteristic that defines service processes. A service significantly involves customer inputs (Sampson & Froehle, 2006). In other words, in order for a service to be produced, a customer has to personally be present, or he/she has to present his/her belongings or information. Compared to classical manufacturing scheduling models, this significant involvement of customer inputs presents additional challenges, including distributed and dynamic scheduling environments, the presence of customers' private information (e.g. the value they place on various scheduling alternatives and their availability), and often considerably more complicated scheduling objectives.

1.1 Example application domains

To motivate this research from a practical perspective, here are some examples of service scheduling:

1.1.1 Transportation service scheduling

The transportation industry comprises a variety of service scheduling problems, such as the routing and scheduling of airplanes, timetabling of trains and carrier scheduling. The carrier scheduling problem, which determines what shipping orders should be assigned to which carriers in a transportation network, is addressed in this work. Each order that has to be transported is characterized by its weight, load port, delivery port, and the time constraints on the loading and delivery times. The carriers and the orders usually belong to different organizations and economic entities, and the customers most likely do not want to reveal information about the highest shipping prices they are willing to pay. Therefore, carriers' schedules should be generated in a distributed environment where the information about carriers and the order information is scattered among multiple independent organizations. For a carrier, a schedule defines the sequence of ports that should be visited within the scheduling window, the time of entry at each port and the orders loaded or delivered at each port. The uncertainty of travel time affects the pickup/delivery times for carriers on congested urban roads, and so the generated schedule should be robust in dynamic environments.

The objective of carrier scheduling typically is to minimize the total cost of transporting all orders. This total cost consists of a number of elements, namely the carrier's operating costs, the fuel costs, and the port charges.

1.1.2 Appointment scheduling in healthcare

There are a variety of problems involved with healthcare systems' scheduling, such as patient scheduling, laboratory and bed allocation scheduling, ambulances and emergency room scheduling and hospital personnel (doctors, nurses, technicians) scheduling. Here the appointment scheduling of high-volume specialized diagnostic services, such as magnetic resonance imaging (MRI) scanning and computed tomography (CT) scanning is used as an example, as it interacts with the services' customers, deals directly with demand uncertainty and has a large influence on many other departments. In such an environment, the capacity of diagnostic resources is limited, is expensive to expand, the demand is highly unpredictable and the waiting lists are already substantial. Healthcare managers and policymakers are therefore under considerable political and community pressure to better manage healthcare resources in order to provide patients with high quality care. To this end, appointment scheduling plays a key role.

Typically there are three objectives in the appointment scheduling problems. The first is to maximize the utilization of the service resource given the patients' availabilities. The second one is to maximize the sum of the of the scheduled patients' priority levels. The third objective is to accommodate patients' preferences. Accommodating patients' preferences in appointment scheduling is important because matching patients with their preferred service provider and offering them a convenient appointment time can decrease the number of no-shows and thereby increase operational efficiency (Barron 1980). However, accommodating scheduling preferences across a large number of patients is particularly challenging due to three areas of complexities: collection complexity, allocation complexity and elicitation complexity. Collection complexity refers to the efforts needed to collect preferences information from patients, which is not an easy task. The vast majority of appointment-booking systems are not automated. They have to rely on human schedulers to collect preferences information, which incurs high administrative costs to the healthcare system. Allocation complexity refers to the computation needed to compute high-quality service time allocations. Accommodating preferences can easily make mathematical models of the appointment booking process intractable, which is perhaps one reason why the majority of mathematical models do not include preferences (Gupta and Denton 2008). These issues are further complicated by the fact that patients are reluctant to reveal their availability. They are actually motivated to protect their private information because revealing too much availability increases the patient's possibility of being assigned an undesirable time slot.

1.1.3 Scientific facility scheduling

The facilities at national science research laboratories are accessible to scientists and researchers so that they can perform their experiments. Researchers' proposals for using these facilities are evaluated each year. The research laboratories normally start by scheduling those experiments with higher priority, and try to schedule as many experiments as possible.

Canadian Light Sources (CLS: http://www.lightsource.ca/), is a national science research laboratory for the production of high-intensity synchrotron light from the infrared, visible, and ultraviolet to X-ray region of the electromagnetic spectrum, and is accessible to scientists and researchers from the academic, government and private sectors. Currently, CLS has about 3000 researchers in Canada and other parts of the world as its user community. CLS send out two calls for proposals each year, resulting in a six-month scheduling cycle. Proposals are evaluated by a scientific committee composed of researchers from universities and industries across the country. Each application is assigned a weight based on its potential contribution to the advancement of knowledge and impact on the scientific community. Approved proposals by the peer review procedure need to be scheduled in the next scheduling cycle. CLS needs to improve the utilization of its valuable synchrotron resources and, at the same time, maximize the overall scientific contributions of the experiments. CLS knows the weight (scientific contribution value) of each application. However, they do not have direct access to researchers'/customers' availability information, and customers are actually motivated not to reveal their availability because revealing too much availability increases the possibility of being assigned an undesirable time slot. The lack of complete availability information can be a major constraint that limits the quality of the schedules. High-quality schedules may be determined to be impossible, given the partial availability of customer information. The service providers are faced with a decentralized scheduling problem, in the sense that the true availability of the customers is their own private information and may not be known to the service provider.

1.1.4 Cloud computing services

"Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services" (Armbrust et al. 2009). With the growth of the cloud computing market, more and more companies start to provide their software and hardware products as services to their customers. Service resource management systems need to provide mechanisms and tools that allow resource consumers (end users) to express their requirements and their time constraints. For any request, a customer has preferences over its completion time and is willing to pay a premium to have it completed during the preferred time windows. Given time constraints of service requests, to maximize profits, the provider has to prioritize service requests based on their profitability and, at the same time, schedule as many profitable requests as possible. The customer's value of a schedule (i.e. the price that she is willing to pay for the request to be completed at a specific time) is her private information. Each customer is motivated to maximize her own payoff, not the system wide optimality. In this context, the scheduling problem is a distributed optimization problem in a strategic setting, which calls for game theoretic solutions.

In the next section, I first describe the Unified Services Theory (Sampson, 2001), which categorically defines services, and then analyze the challenges in service scheduling in light of that theory.

1.2 Unified Services Theory

Services have been commonly defined as intangible products (Pearce, 1981, p. 390; Bannock et al., 1982, p. 372; Harvey, 1998, p. 596). In other words, a service typically does not result in the ownership of anything (Kotler, 2006, p. 402). Intangibility is an important characteristic of services. However, as stated in Sampson and Froehle (2006), it does not serve as a sufficient condition which defines a production process as a service. For example, software development results in a product that is intangible (computer code), but the output can indeed be inventoried and used or sold later. Unified Services Theory, on the other hand, identifies a single commonality that comprises all services. It defines what services are and what they are not. To facilitate the analysis of service implications to scheduling, it is useful to first introduce the Unified Service Theory.

The Unified Services Theory (UST) is formally stated as follows (Sampson, 2001, p. 16):

"With service processes, the customer provides significant inputs into the production process. With manufacturing processes, groups of customers may contribute ideas to the design of the product,

but individual customers' only participation is to select and consume the output. All managerial themes unique to services are founded in this distinction."

The most important component in UST is customer inputs, which distinguish services from manufacturing processes and are the root cause of the unique issues and challenges of services management. The literature has typically identified three general types of customer inputs (Wemmerlov, 1990): the customer's self, his belongings or other tangible objects, and information. Customer-self inputs are common in services involving co-production (i.e., the employment of customer labor in the process) and in services involving the physical presence of the customer. Typical examples are health care clinics, buffet restaurants and taxi services. These service providers can prepare for production, but they cannot execute the actual service process until necessary customer-self inputs are present. Tangible belongings (or property) and physical objects make up another type of input a customer can provide to the service process. One's car is an essential input into the automobile repair service process and one's clothing is a necessary input to the dry cleaning service process. Providing tangible inputs often allows the service process to proceed even without the customer being physically present. Customer-provided information is a third type of input to the service process. For example, the tax return preparation process requires that customers provide financial information as the process inputs. The service production process cannot begin without the input of that information.

The UST reveals principles that are common to the wide range of services and provides a unifying foundation for various theories and models of service operations. As demonstrated in Sampson and Froehle (2006), the UST has significant operational corollaries pertaining to the services management process. Among them, capacity management and demand management significantly rely on the scheduling of service resources. The challenges in designing service scheduling systems are presented in the rest of this section.

1.3 Challenges of Service Scheduling

Scheduling plays an important role in service management due to the perishable nature of service provider's capacity. A service provider has to pay scheduled workers even though there are no customers currently needing services. In other words, the service provider's capacity to produce the service is time-sensitive and cannot be inventorized by producing to stock. This high "operating leverage" implies that many service operations will be much more cost-competitive if the service providers effectively manage variable demand (Hur et al., 2004; Jack & Powers, 2004), which gives them higher utilization levels (Sampson, 2001, p. 240) or, alternately, manage capacity, which increase their volumes.

The management of demand and capacity involves the allocation of service orders and resources over time, which is essentially a scheduling activity. On the demand management side, reservation systems schedule customer inputs into the production process such that waiting times are minimized. On the capacity management side, service managers schedule full- and part-time personnel to meet the expected workload for a future day. When the day of service arrives, if a significant gap is present between the experienced workload so far and the scheduled staff capacity, service managers will attempt to make an immediate adjustment to the staff schedule by changing station assignment, shifting breaks, or calling in additional workers (Hur et al., 2004). Compared with classical manufacturing scheduling, service scheduling presents different challenges attributable to significant customer inputs in service production processes. Three important service scheduling challenges, namely distributed and dynamic environments, complicated objectives and customers' private information are described below.

Distributed and dynamic environment: The customer input requirement in services leads to a distributed and dynamic scheduling environment. First, the information needed for computing schedules, e.g. customers' availability and preference information, is scattered among possibly a large number of customers. Collecting the information and keeping it up to date can be challenging tasks. For example, consider the appointment scheduling problem. As mentioned before, considering patients' preferences and their availability are both important because patients need to be present themselves as an input to the service process. However, information about patient's preferences is distributed among patients themselves, and patient's preferences may change over time because of changes in work schedule or marital status. Therefore, appointment scheduling problems should be generated in a distributed and dynamic environment.

Transportation scheduling is another example of a distributed environment in service scheduling. Transportation companies have to carry out transportation orders. These orders are customer inputs which are distributed geographically. Each order should be picked up from a location and delivered to a destination. Transportation companies are geographically distributed and have a set of trucks at their local disposal. Each company makes decisions about its local scheduling according to the local trucks and the actual solution to the global order scheduling emerges from the local decision-making of these companies. Modeling the companies as independent and autonomous units seems the only acceptable way to proceed, because the task of centrally maintaining knowledge about all of the shipping companies, their vehicles and their policies is very complex. Moreover, this information is often not even centrally available (real-life companies are not willing to share all their local information with other companies) (Fischer et al. 1995).

In addition to the complexity arising from the distributed environment, service scheduling has to be robust in accommodating the contingencies caused by customer involvement in service production. Uncertainties regarding customer demand, resource availability, service times, customer cancellations and no-shows make the scheduling of services a complex dynamic process. For example, in the appointment scheduling problem, patients who make an appointment and fail to keep it can lead to poor resource utilization and longer patient waiting times. Service durations are also subject to change in appointment scheduling. Patient attributes such as age, degree of disease progression, cultural background and language fluency (need for an interpreter) can affect service durations (Gupta & Denton 2008). Longer than expected service duration results in late starts for the rest of the services that day. Late starts leads to costs associated with overtime staffing.

Service organizations also face uncertainty in the numbers of consumers and their resource demand. Examples include mail processing facilities, airline reservation desks, hospitals, telephone operators, and so on. In each of these dynamic environments, personnel scheduling is a challenging problem, and the goal is to assign personnel with different skills to each shift in order to cover the predicted demand.

In a service setting, customers may be requested to include additional, unanticipated tasks, or to adapt to changes to several tasks, or to neglect certain tasks. For example, consider the appointment scheduling problem. Medical treatments in a primary care clinic are often not completely pre-determined before an examination. The examination results may invoke additional activities and/or make other medical actions unnecessary (Paulussen et al. 2003). The beginning time and the processing time of a task are also subject to variations. A task can take more or less time than anticipated, and the customer inputs can arrive early or late. An optimal schedule, generated after considerable effort, may rapidly become unacceptable because of unforeseen dynamic situations. Since service capacity cannot be inventoried by producing goods, customers that fail to present their inputs according to the schedule can contribute to poor resource utilization, lower revenues and longer waiting times. The time-sensitive nature of service capacities signifies the need for more robust dynamic scheduling approaches. In addition, unlike manufacturing environments where the amount of resources (which are typically machines) is usually fixed (at least for the short term), in services, the number of resources (e.g. people, rooms, and trucks) may vary over time. Certain resources may become unavailable, and additional resources may need to be introduced. This variable can even be a part of the objective function (Pinedo, 2009).

As an example of a dynamic service scheduling environment consider a transportation scheduling problem in which orders have to be picked up and delivered at specific customer locations by a limited number of trucks. The main challenge is that the orders are not all known in advance. New orders can be received and then must be incorporated into the scheduling process. Truck availability adds further dynamicity. Trucks may be delayed or temporarily unavailable due to traffic or other unforeseen problems. In addition, the actual sizes of orders are subject to change (Davidsson et al, 2005).

In another dynamic service scheduling example, consider service computing where the amount of resources varies over time. In this environment, resources need to be dynamically (re-)configured and bundled via virtualization to provide different service profiles for dynamic demands (Sim, 2012).

The service scheduling process is further complicated by the fact that customers' needs for services have varying degrees of urgency, and some decisions about non-urgent requests must be made before the complete information about urgent and emergency demands is known. Take the example of appointment scheduling in diagnostic services; the low-priority demand (outpatients) must be booked (often several weeks in advance) before knowing the highly unpredictable high-priority demand (inpatients). To accommodate the demand imposed by the highly dynamic high-priority inpatients, the hospital is forced to reserve a significant portion of the total capacity for this unknown high-priority demand, leaving little room for outpatients. This results in unused capacity on days when inpatient demand is lower than expected and thus longer waiting times for outpatients than there would if this unused capacity could be utilized. Moreover, a patient's priority may change along the treatment process.

Complicated objectives: Planning and scheduling objectives in service industries are often considerably more complicated than those in manufacturing. Scheduling objectives in manufacturing are typically a function of the completion times, the due dates, and the deadlines of the jobs. Objectives in services often have additional dimensions. In contrast to manufacturing, the number of resources in a service environment may be variable (e.g. the number of full-time and part-time people employed). Given this situation, there may very well be a different type of objective – one that minimizes the number of resources used and/or minimizes the cost associated with the use of these resources (Pinedo, 2009). This minimization is a typical objective of capacity management.

Customer preferences regarding the timing of delivering their inputs should also be considered in service scheduling, as they represent the value that customers attribute to a schedule. For example, in healthcare services, patients want more personalized care, which includes their involvement in selecting appointment times. Some patients prefer an appointment on same day they call, or soon thereafter, and the day of the week or the time of the appointment is not particularly important to them. Others prefer a particular day of the week and a convenient time. These patients do not mind waiting for convenience. In both private and public healthcare systems, healthcare managers are motivated to achieve high scores on patient satisfaction surveys. In addition, offering patients a convenient appointment time can decrease the number of no-shows and thereby increase operational efficiency (Wang and Gupta, 2011).

Transportation service scheduling is another example of complicated objectives in service scheduling. The objective typically is to minimize the total cost of transporting all orders. This total cost consists of a number of elements, namely the ships' operating cost, fuel costs and the port charges. There are two kinds of ships operating in this realm. One type is company-owned and the other type is chartered. The operating costs of a company-owned ship are different from those of a charter. Companies make decisions about using their own resources or using chartered ships in a way that minimize their total cost (Pinedo, 2009).

Customers' private information: Service processes involve significant customer inputs, which, in many cases, require that services are produced and consumed simultaneously. Scheduling systems are used to synchronize the timing of the use of the different types of resources and the presence of customer inputs. To compute optimal schedules, ideally the scheduler should know the complete customer availability information within the scheduling horizon. However, collecting availability information across a large number of customers requires a significant amount of communication between the scheduler and the customers. This amount of communication can incur high administrative costs if the collecting procedure is not automated, which is the case of most existing service scheduling systems. The issue is further complicated by the fact that customers are reluctant to reveal their complete availability because their personal schedule is their private information and revealing too much availability increases the possibility that a customer will be assigned an undesirable time slot.

Consider the scientific facility scheduling environment. The CLS has two calls for proposals each year resulting in a scheduling cycle of 6 months. Bob needs to conduct his experiment in the facility. He can be available anytime from January to August. However, he prefers the experiment to be scheduled as early as possible because there is a possibility that he may go on vacation sometime during the summer. Based on his previous experience and his knowledge of the profile of the current year's applications, he believes that experiments with similar weights to his are likely to be offered a service time slot two months after the originally requested dates. Therefore, statistically, if he reports January to April as his available time window, he will have a much higher chance of being assigned time June or even sometime earlier. Therefore, based on his calculation and his knowledge base, Bob may indicate only January to April, which is not his complete availability.

For another example, consider meeting scheduling in which participants' calendars are usually considered private objects and so their information as confidential. There are situations when participants do not want to make their available time public. For example, people usually will not hire a consultant or schedule an appointment with a dentist who indicates that he/she has plenty of free time. Because of the social attribution of " importance " to people with little free time, many people may not be willing to publish their actual availability (Wainer et al. 2007).

Cloud computing systems that allow users to acquire computing resources and pay for it on a short-term basis are another example of service scheduling where customers have private information. From the economics literature (Zaman and Grosu 2011), it is evident that a fixed-price mechanism cannot support efficient resource allocation and cannot guarantee that the user who values an item the most will get it. Achieving economic efficiency in resource allocation should thus be based on the perceived values of the users. User value is the user's private value; it is the highest price they are willing to pay for a given service. In many cases, users are reluctant to reveal their values and may fear that the provider will take advantage of their information and use it to charge higher prices for their required services.

For multiple reasons, customers are motivated to protect their private information. Therefore, service scheduling systems should be designed so that they are able to elicit the necessary customer's private information required to compute high-quality schedules. The computation spent on eliciting customers' availability information is referred to as a system's elicitation complexity.

1.4 Scope and Approach

The objective of this research is to develop theories and approaches to service scheduling problems. Economic-based models (auction-based in particular) are used as the anchor from which to tackle the challenges in commercial service scheduling and non-commercial service scheduling problems. The contributions can be summarized as follows: (1) presenting an integrated framework that addresses the scheduling challenges in general service scheduling problems, (2) developing economic-based approaches for both commercial and non-commercial service environments, (3) showing the effectiveness of the framework through its application to various application domains, and (4) designing and implementing a prototype to validate the computational properties of the proposed framework.

1.5 Thesis Organization

The rest of the thesis is structured as follows. Chapter 2 provides a brief overview of traditional scheduling approaches to service scheduling and reviews the literature on agent-based service scheduling. Chapter 3 describes the general service scheduling problem studied in this thesis. An iterative bidding framework for services scheduling is presented in chapter 4, followed by that same framework adapted for scheduling non-commercial services in chapter 5. In chapter 6 I demonstrate the applicability of the proposed framework in two different application domains. The design and implementation of a prototype of the proposed approach and the simulation results are presented in Chapter 7 and finally, chapter 8 summarizes the thesis.

Chapter 2 Literature Review

This chapter presents a review of service scheduling models. General scheduling problem definitions are introduced first, followed by an overview of traditional centralized service scheduling approaches. The focus is then narrowed to the literature on agent-based service scheduling. Finally, the related literature is summarized, with an emphasis on how each work addresses the challenges in service scheduling and identifying the presented research position in relation to the big picture of existing approaches.

2.1 Definitions of the scheduling problem

Scheduling is a decision-making process which deals with the allocation of resources to tasks over given time periods under imposed constraints; its goal is to optimize one or more objectives. Bowman (Bowman, 1959) presented a simple definition from operations research field perspective: The scheduling problem in its most simple form consists of a number of jobs to be done on a number of machines, each job having a number of operations to be performed by the various machines in a specified sequence; what feasible schedule covers the least total time?

Pinedo (2008) and Brucker (2004) provide more complete description of various scheduling problem models. Although there are a variety of definitions, most of the scheduling problems can fit into a four element: *resources*, *jobs*, *constraints*, and *objectives*. Figure 2-1 presents the relationships of these elements: resources are assigned to jobs over time, this assignment process is restricted by the constraints and guided by the objectives. Note that, in the context of scheduling, indicating that a resource is assigned to a job does not mean that that resource is dedicated to that job. A more accurate interpretation would refer to a period of processing time from a resource is assigned to a job. In Figure 2-1, the dotted arrow pointing from the Processing Times to the “Assigned” action reflects this interpretation (Wang, 2007).

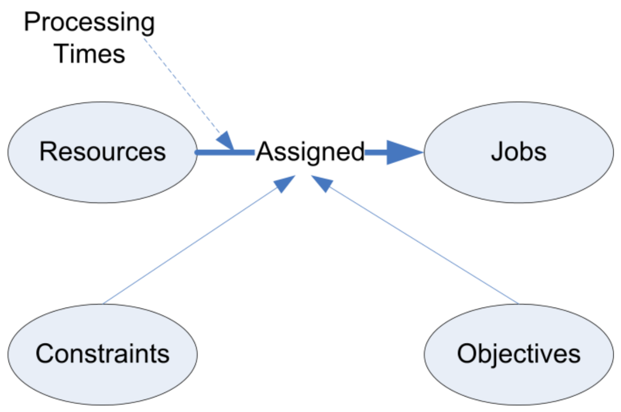


Figure 2-1 Four-element structure of scheduling problems (Wang, 2007)

In general, resources and jobs can take many forms. The resources may be a machine, operating room, airport gate, processing units in a computing environment, and so on. The jobs may be transporting a cargo, take-offs and landings at an airport, surgery, executions of computer programs, etc. *Constraints* are a set of conditions that must be satisfied, e.g. precedence constraints, time window constraints on release time, deadlines, or resource capacity constraints. *Objectives* may be the minimization of makespan, or of total cost, maximization of the resource utilization, or of the throughput. A solution to a scheduling problem is called *feasible schedule* if it satisfies all constraints of the problem. Otherwise, it is called an *infeasible schedule*.

2.2 Centralized Service Scheduling Approaches

Traditional service scheduling approaches usually assume a centralized environment in which a scheduler has all the information needed to compute the schedule. Various service scheduling models have been proposed, implemented, and evaluated, some for several decades. Generally speaking, the solution methods form two distinct classes: exact methods and heuristic methods. Exact methods are guaranteed to find a solution if it exists, and typically provide some indication if no solution can be found. However, given the NP-hard nature of service scheduling models, exact methods are not practical for non-trivial problem instances. Heuristic methods do not guarantee optimization, but typically assure, experimentally or analytically, some degree of optimality in their solutions. They are usually rapid and practical ways of solving larger-sized scheduling problems. In this section, we briefly review some general heuristic methods and their application to service scheduling problem.

2.2.1 Genetic algorithms

Genetic Algorithms (GAs) are a set of global search and optimization methods for solving complex optimization problems with a large search space. With the objective of reaching the "best" solution, GAs systematically evolve a population of candidate solutions by using evolutionary computational processes inspired by genetic variation and natural selection. One of the earliest GAs for scheduling was proposed by Davis (1985). Davis suggested an indirect representation which can be decoded to form the actual schedule of the scheduling problem. GAs have been applied to many service scheduling problems. For example, Ghaemi et al. (2007) proposed a co-evaluation algorithm for university timetabling problems. Paechter et al. (1995, 1996) apply a memetic algorithm for course timetabling. The memetic algorithm explores the neighbourhood of the solution obtained by GA and navigates the search towards the local optima. Graph colouring heuristics were used by Burke et al. (1995, 1996, & 1998) to improve and accelerate the search process in timetabling. Burke et al. (1995) also developed a hybrid GA to ensure that the most fundamental constraints are never violated in timetabling problem. They showed that the algorithm is guaranteed to produce a feasible solution by hard coding the constraints and using a hybrid crossover operator. In addition to timetabling, GAs have also been used to solve scheduling problems in healthcare, such as patient scheduling and nurse scheduling (Petrovic & Morshed, 2011; Aickelin & Dowsland, 2001).

2.2.2 Simulated annealing

Simulated Annealing (SA), is a neighbourhood search method. Rather than always choosing the direction of the best improvement, which gives the steepest-ascent, SA initially chooses random or semi-random directions, but over time comes to prefer the direction of the best improvement. The direction selection process is controlled by a sort of temporal parameter, which is usually called the 'temperature' by analogy with real annealing. SA approaches require a schedule representation as well as a neighbourhood operator for moving from the current solution to a candidate solution. Annealing methods allow jumps to worse solutions and thus often avoid local sub-optimal solutions (Kirkpatrick et al., 1983). The quality of solutions produced by an SA implementation depends on the correct choice of solution space and neighbourhood, as well as of the parameters that govern the cooling schedule. SA has been applied to service scheduling for several years. Gunawan et al. (2007) used a hybrid algorithm which consists of an integer programming, a greedy heuristic and a modified SA algorithm for solving large-scale timetabling problems. Bailey et al. (1997) solved a nurse scheduling problem using SA and compared its performance with integer programming and with a GA. They found that, for a given quality, their algorithm was faster than the GA and integer programming for the set of nurse scheduling testing problems.

2.2.3 Tabu search

Tabu search (TS) is similar to SA in that it also moves from one schedule to another, with the next schedule being possibly worse than the one before it. The difference is in the mechanism by which the moves to new schedules are accepted. A TS maintains a list of tabu moves, representing schedules which, having been visited recently, are forbidden in order to diversify the directions in which search proceeds. TS has been proposed to compute high-complexity large-sized health care service scheduling. Dowsland (1998) used tabu search with strategic oscillation for nurse scheduling. The objective is to ensure an adequate number of nurses are on duty at all times while incorporating individual preferences and requests for days off in a way that is seen to be fair to all nurses. The method uses a variant of TS which oscillates between solutions with feasible nurse coverage and then applies nurses' preferences to improve the solution. Demeester et al. (2010) proposed a hybrid TS algorithm for patient admission scheduling. It automatically assigns patients to beds in the appropriate departments by considering patients' medical needs as well as their preferences, while keeping the number of patients in the different departments balanced. The method uses a TS algorithm hybridized with a token-ring and a variable neighbourhood-descent algorithm. TS has also been applied to university course timetabling problems (Hertz, 1991; Hertz, 1992).

2.2.4 Constraint logic programming

Many service scheduling problems can be modelled as constraint satisfaction problems (CSP). In a CSP, values which satisfy a set of constraints must be found for a set of discrete variables with finite domains. Constraint satisfaction is a search procedure that operates in the space of constraint sets rather than in that of solution sets. Constraint Logic Programming (CLP) provides the ability to declare variables and their domains for CSPs. Examples of applying CLP to service scheduling problems can be found in Gueret et al. (1995), Henz and Wurtz (1995), and Abdennadher and Schlenker (1999).

2.2.5 Approaches considering customer preferences and dynamic environments

Due to the computational complexity involved in creating schedules that simultaneously consider customer preferences and scheduling objectives, there has been limited research in centralized service scheduling that considers customer preferences. Wang and Gupta (2011) proposed a heuristic approach for patient scheduling which captures customer preferences. Their method has two components. The first one dynamically learns a patient’s preferences and updates estimates of acceptance probabilities. The second one uses the acceptance probability information to make booking decisions. Jaumard et al. (1998) proposed an integer programming model accommodating nurses’ work preferences. The problem was solved using Dantzig-Wolfe decomposition. The objective was to minimize salary costs and maximize the nurses’ preferences. Azaiez and Sharif (2005) developed a 0-1 linear goal programming model for nurse scheduling. They obtained the nurses’ preferences for shift times from a survey consisting of 15 multiple choice questions. The nurses’ preferences were combined with hospital constraints to develop their linear goal programming model.

Centralized service scheduling usually deals with dynamic environments by using simulation-based approaches. A simulation is the imitation of the operation of a real-world process or system over time (Groothuis & Merode, 2001). An advantage of simulation studies over heuristic approaches is the ability to model complex systems and represent environmental variables. Hancock and Walter (1984) conducted a simulation study based on historical data of patient arrivals. The simulation was used to determine the number of procedures that would be performed each day of the week. Groothuis and Merode (2001) applied the discrete event simulation technique to optimize the use of catheterization capacity in a hospital. Ho and Lau (1999) proposed a simulation-based method for evaluating the impact of different combinations of dynamic environmental factors such as no-shows, service times, and the number of customers per service session on the quality of service schedules.

The above-mentioned traditional scheduling methods encounter great difficulties when they are applied to real-world situations, as they use simplified theoretical models and are essentially centralized in the sense that all computations are carried out in a central computing unit. The intelligent agent technologies, on the other hand, suggest an innovative, lightweight approach to scheduling problems. The main characteristic of intelligent agents is their autonomy. Each agent makes its own decisions, based on its internal state and on the information it receives from its environment; thus, each agent can keep its independency from the rest of system. In other words, each agent, according to its own private information, may use a different policy independently from the rest of the system. Agent-based systems are inherently distributed and robust in dynamic environments. Agents can retrieve information from different resources, analyze it, filter redundant information, select and then present the data by means of an interface that is attractive to users. Another feature of agents is their sociability; agents can communicate with each other and exchange any kind of information. This sociability, makes it possible for them to overcome any inconsistency among their local schedules and resolve errors and collaborate in the process of scheduling. Based on the properties of agent-based systems, an agent-based approach should be a good candidate for service scheduling problem.

2.3 Agent-based Scheduling

Agent-based scheduling can be defined as an approach in which scheduling problem are decomposed among local decision makers who may have conflicting objectives but who coordinate with each other through certain communication mechanism to achieve overall system objectives. Local decision makers are called agents (Sycara et al. 1991, Kouiss et al. 1997, Shen 2001). Agents' properties include autonomy, so they can operate without human interaction, a social ability to communicate with other agents, pro-activeness, which allows them to take on an initiative role, and reactivity, to respond to changes in the system (Rahimifard and Newman 1998).

Agent-based scheduling has received considerable attention in manufacturing areas (Burke and Prosser 1991, Chung et al. 1996, Maturana and Norrie 1996), and it has also been applied to other application domains such as network power scheduling and packet scheduling (Chiussi and Francini 2000, Hohlt et al. 2004, Vaidya et al. 2005). Supply chain optimisation is also another important application area for agent-based scheduling. (Tsay et al.2000, Lau et al. 2005a, 2005b, Frayret 2009).
Several papers provide reviews of the literature on agent-based scheduling (Sen 1997, Tharumarajah 2001, Shen 2002, Caridi and Cavalieri 2004, Giret and Botti 2004, Shen et al. 2006). Tharumarajah (2001) provides a classification for the literature based on the following attributes: problem decomposition, problem-solving organisation, coordination and control. Problem decomposition describes how the global scheduling problem is decomposed between multiple decision makers (i.e. agents). Problem decomposition approaches are categorized based on three views: resource view, task view and hybrid view. The other two attributes focus on how agents communicate and cooperate with each other in order to achieve improved global performance. Shen (2002) classifieds the reviewed papers in terms of four issues in agent-based manufacturing scheduling. Shen et al. (2006) extended that work to include studies on agent-based approaches to manufacturing process planning. The issues are identified as agent encapsulation, coordination and negotiation protocols, system architectures and decision schemes for individual agents. Caridi and Cavalieri (2004) propose a taxonomy that includes application domain, agent, control, organisation and communication for classifying multi-agent systems. .

This section provides a brief literature review and classifies the papers based on three attributes: information flow structure, communication mechanism and schedule generation. With respect to information flow structure, the literature can further be classified into two main groups: mediated structure and autonomous structure. In a mediated structure there is a coordinator agent that other agents communicate through. Each agent makes its own local schedule considering its goals, and then communicates that schedule to the coordinator agent. The coordinator agent evaluates the local schedules with respect to the overall system objectives, resolves potential conflicts, and finalizes the scheduling decisions. The scheduling system proposed by Lau et al. (2005) has a mediated structure. Their supply chain is modelled as a multi-agent system, with three types of agent: project agent, contractor agent and middle agent. Companies aim at completing a project, which consists of a network of operations that act as project agents, and contractor agents offer their bids for performing those operations. A middle agent facilitates and coordinates the scheduling process between project agents and contractor agents. Babayan and He (2004) use a mediated structure for scheduling jobs in a flexible flow shop. In this system, local agents correspond to the jobs, while a manager agent decides whether rescheduling should be performed at any point and sets the rules of the game. Cooperative game theory is used to regulate the competition among the agents for scheduling their jobs. The game consists of two steps. The first step determines the agents that are eligible to schedule their jobs, and the second step enables competition among the agents by allowing them to schedule their jobs. Wang et al. (2009) solve the job shop scheduling problem by using auction theory. Their proposed system consists of two types of agents: job agents and resource agents. Job agents are associated with each job and they participate in the auction to process their jobs. Each bid includes a latest completion time and the price for completing a job before a specified time. A resource agent is associated with all of the resources in a job shop; it is considered as a mediator agent that can determine the resource allocation to the job agents that maximizes the sum of bidder prices.

In an autonomous structure, agents communicate directly with each other, and the interactions between agents are not coordinated by mediator agents. Duffie and Prabhu (1994) use an autonomous structure for manufacturing systems. Each resource is assigned to an agent. The information regarding a new job is passed to resource agents when a job arrives. The resource agents affected by the arrival of this job develop alternative scheduling plans. Scheduling plans may conflict because resource agents act independently. One or more agents discover the conflicts and send a feedback report to the local agents. With this feedback, local agents develop alternative local plans. In a study by

Sycara et al. (1991), each local agent determines its overall demand in time for each resource and presents this data to the resource agents. After receiving all the demands from the local agents, each resource agent starts to schedule the most critical operations. The peak of the aggregate demand determines the critical operations. The critical operations are scheduled based on the survivability measure. After scheduling each operation, local agents update their demand information. Information exchange between local agents and resource agents continues iteratively until a feasible schedule has been achieved.

With respect to the second classification attribute, communication mechanism, the literature can be broken into three main groups: Contract-Net protocol, economic models, and iterative refinement. In the communication mechanism based on Contract-Net protocol, a new job broadcasts its arrival and requests bids for its processing. The agents (cells or machines) prepare bids. The best bid is selected by a manager agent according to some criteria. Smith (1980), Smith and Davis (1981), Parunak (1987), Shaw and Whinston (1988) and Lima et al. (2006) are examples of studies that apply the Contract-Net protocol to the communication mechanism between agents.

Several economic models that support distributed rational decision making were studied in Sandholm (1999); among them, auction is the most relevant to scheduling. Auctions assume game-theoretic agent behavior. The equilibrium state is defined by the condition that agents play a best-response strategy to each other and cannot benefit from a unilateral deviation to an alternative strategy. In Kutanoglu and Wu (1999), iterative auctions are applied to the job shop scheduling problem. The focus is to investigate the links between combinatorial auctions and Lagrangean relaxation, and then to design auctions based on Lagrangean-based decomposition. In MacKie-Mason et al., (2004) and Wellman et al., (2003), price prediction and bidding strategies for simultaneous auctions are studied in the setting of market-based scheduling. Simultaneous auctions sell multiple goods in separate markets simultaneously. Agents have to interact with simultaneous but distinct markets in order to obtain a combination of resources sufficient to accomplish their task. These auctions fail when agents bid cautiously to avoid purchasing an incomplete bundle.

The third communication mechanism is iterative refinement. In this mechanism, scheduling information is exchanged between agents to eliminate conflicts and to revise an existing schedule to achieve better system performance (Sycara et al., 1991, Liu and Sycara, 1993). In Liu and Sycara (1993), the resource agents generate initial schedules using the earliest due date rule. Since resource agents act independently, any generated schedule may violate precedence constraints. A coordinator agent extract the information about the operation of each job and send it to the local agent. Next, the local agent identifies and eliminates any precedence violations. The procedure continues in an iterative manner until a feasible schedule is generated. In later studies, the same authors propose mechanisms for loop prevention (Sycara and Liu 1994, 1995).

The third attribute applied to classify the literature, schedule generation, describes how the sub-problems are solved by the agents. Bid preparation (Yang et al. 1993), dispatching rules (Kouiss et al. 1997, Hadavi et al. 1992), heuristic methods (Sycara et al. 1991, Trentesaux et al. 1998), constraint-based branch and bound (Wang et al. 2009) and mixed integer programming (Babayan and He 2004) are some of the methods found in the literature.

It is important to emphasise that, although there are several review papers on agent-based scheduling, there is no survey of the literature covering agent-based service scheduling. The next section provides that review of agent based service scheduling.

2.4 Agent-Based Service Scheduling

Agent-based service scheduling is essentially a distributed approach that is flexible, efficient, and adaptable to real-world dynamic environments. By applying agent-based service scheduling architecture, the distributed nature of service scheduling can be modelled naturally. In addition, each agent can be assigned different objectives. In this way, the complicated multiple objectives in service scheduling can be decomposed to individual agents. This decomposition significantly simplifies the modelling of the objectives. Agent-based scheduling systems have been proposed for several important service sectors. However, there is a lack of general problem formulations, classifications, solution frameworks, and test beds in service scheduling. Therefore a domain-specific approach is applied here. Several representative application domains are reviewed through the lens of how agent-based scheduling addresses service scheduling challenges. Since the challenges of distributed scheduling information and complicated multiple objectives have been naturally modelled in agent-oriented design paradigms, in this section, the focus is on how agent-based scheduling addresses the challenges of dynamic environments and users' private information.

2.4.1 Meeting scheduling

A meeting scheduling problem signifies a decision-making process affecting several users, in which it is necessary to decide “when” and “where” one or more meetings should be scheduled (Hassine et al., 2004). Since it usually involves inputs from multiple users, meeting scheduling can be classified as a service scheduling problem. Agent-based meeting scheduling approaches have been proposed in the literature. Some of them are distributed implementations of constraint-satisfaction algorithms in the multi-agent systems environment. In the multi-agent meeting scheduling system developed by Franzin et al. (2002), agents communicate over several proposal phases. Whenever agents communicate during the proposal phases, the information they exchange can be used to build an approximation of the constraint set(s) of the other agents. In other words, each agent in the proposal phase is able to elicit other agents’ availabilities. To deal with the challenge of a dynamic environment, Hassine et al. (2004) formalize meeting scheduling as a dynamic valued-constraint satisfaction problem. Agents negotiate with each other to achieve a schedule that maximizes global utility. In the negotiation process, a host agent proposes a set of time slots as a solution to the other agents who will participate in the meeting. Each participant agent that has received this message ranks the obtained time slots according to its preferences and constraints and returns them to the proposer/host agent. The host/proposer agent tries to find the best solution, one which maximizes its utility, from the received time slots. The same process continues until an agreement is reached among all of the agents. Course timetabling at universities, which can be seen as a type of meeting scheduling problem, is modelled as a constraint satisfaction problem by Meisels et al. (2003). An inter-agent negotiation protocol is used to overcome inconsistencies among local schedules.

The presence of users' private information is also addressed in agent-based meeting scheduling. Wainer et al. (2007) defined four levels of privacy protocol (or modes of agents' interaction) to model users' private information, namely, full information, approval, voting and the suggestion protocol. These modes of interaction are defined based on whether the participants are comfortable sharing their private information during the negotiation process with the host or not. In Modi et al. (2005), the agents' private information is modelled as their utilities. Each agent makes a decision about accepting a meeting time based on how the decision will impact its own utility. The utility of a time slot is calculated based on the difference between the value of a meeting scheduled in the time slot and the predicted cost of continued negotiating with other agents.

Crawford et al. (2004) designed a mechanism for meeting scheduling which is incentive-compatible. A mechanism is incentive-compatible if it is every agent's dominant strategy to reveal their private utility values truthfully. The mechanism motivates agents to reveal their valuation for each of the feasible schedules. The schedule that maximizes the social welfare is selected. Each agent's payments are VCG auction payments, which justifies the incentive compatibility of the mechanism. Iterative auction are also used in agent-based meeting scheduling. In a course timetabling system proposed by Sönmez and Ünver (2007), students are assigned a certain amount of bid endowments which they use to bid for different schedules of courses. Students are modelled as price-takers under a belief system. In other words, students' bids are based on their guess about the market-clearing price they will face. Krishna and Ünver (2007) also proposed a course bidding system and conducted a field test in the spring 2004 semester at the Ross School of Business, University of Michigan. In their bidding system, student bids are used to infer students' preferences for courses and to determine their course priorities. In addition to handling users' private information, the challenge of dynamic environments is addressed in agent-based meeting scheduling (Sönmez and Ünver 2007).

2.4.2 Healthcare

The agent-based approach, in which patients and hospital resources are modelled as autonomous agents with their own goals, reflects the decentralized structures of the health care environment. Most of the agent-based healthcare scheduling literature focuses on the challenge of the distributed and dynamic environment of healthcare management. In a recent study of operation room scheduling, Zhiming (2011) developed a two-stage approach which addresses the challenges of dynamic scheduling. Mixed integer programming is used in the first stage for assigning surgical operations to each operation room. The second stage utilizes a dynamic rescheduling approach, in which agents reallocate tasks using the contract net protocol in a way that minimizes the cost of the operation rooms.

Agent-based approaches have also been proposed for patient scheduling. Hannebauer et al. (2001) formulated patient scheduling as a distributed constraint optimization problem. They proposed a Multi-phase Agreement Finding (MPAF) algorithm for coordinating the agents and covering the constraints. The MPAF consists of two phases, the proposal phase and the assignment phase. In the proposal phase, a diagnostic unit agent selects a set of feasible appointment time slots based on its optimization criteria and proposes these to the patient agent. In the assignment phase, the patient agent decides whether to accept the proposed time slots. This decision is made based on the agent's scheduling constraints and its scheduling objective, which is to minimize the waiting time between appointments. Other agent-based patient scheduling approaches model the scheduling environment as a market. Given the distributed and dynamic nature of patient scheduling, markets can efficiently distribute scarce resources among patients. Paulussen et al. (2003) developed a bidding mechanism for patient scheduling, in which patient agents communicate their (private) utility for certain time slots of a resource via a price mechanism. The price that patient agents are willing to pay is the difference between the cost-value of the current allocation and the cost-value for the desired appointment. Resources are assigned to the patients that are willing to pay the highest price (to the patients who gain the highest health sate improvement). The scheduling objective is to maximize resource utilization and minimize patient's hospital time. For patients who need to schedule several related appointments, a multi-round auction mechanism is proposed by Hosseini et al. (2011). In this approach, patients calculate the value of obtaining each resource by solving their Markov decision problem. In each auction round, agents submit their bids; the auctioneer determines the winner and then moves to the next step. The objective of winner determination is to minimize the global regret values of patients. A patient’s regret value on a resource is defined as the difference in value between getting the resource and not getting the resource, given a patient’s current health state.

Agent-based approaches have also been proposed for nurse timetabling. Grano et al. (2009) proposed an auction-based nurse scheduling approach that considers both nurse preferences and hospital requirements. In the auction, nurses bid for their work shifts and rest days using points instead of money values. The nurses’ private information, which consists of their availability and preferences for specific days and shifts, are thereby obtained in the bidding stage. Winners are selected using an optimization model which seeks to award shifts to the highest bidders while simultaneously meeting hospital requirements.

2.4.3 Transportation

An agent-based approach have been adopted in transportation planning and scheduling research for more than two decades. Fischer et al. (1995) pointed out that transportation planning and scheduling are inherently distributed, complex tasks. Geographically, trucks and jobs are distributed and maintain some level of autonomy. To implement traditional methods, a scheduler must gather a large amount of information to a central place where the solution can be computed. However, using an agent-based approach, an agent only requires local information. In their review on multi-agent systems in logistics, Lang et al. (2008) concluded that transportation planning and scheduling problems have specifications that comply with the particular capabilities of agent systems. Specifically, these systems are able to deal with inter-organizational and event-driven scheduling settings that meet a supply chain’s planning and execution requirements. Davidsson et al. (2005) also identified a number of the positive aspects of the agent-based approaches to logistics. Existing surveys (Lang et al., 2008; Davidsson et al., 2005) mainly focus on research addressing the distributed and dynamic aspects of transportation services. The rest of this section provides a review of papers focusing on the challenge of the presence of customers' private information, which is mainly tackled by the design of the various auction systems in the context of multi-agent systems.

Auction mechanisms, especially combinatorial auctions, have been adopted by a large number of shippers and 3PL (third-party logistic) providers. Leading companies such as Wal-Mart, Procter & Gamble and Sears have used combinatorial auctions to reduce their logistic costs (Sheffi, 2004). Song et al. (2003) proposed an auction-based mechanism, the Collaborative Carrier Network, for carriers to exchange their excess capacities in a TL (truckload) spot-market. Through this network, carriers can buy and sell transportation capacities. The network is structured as a group of auctions launched by carriers. Each carrier can be both a contractor and a sub-contractor in different auctions. A carrier will launch, at most, one auction at a time, and if new loads come in during the previous auction round, they will be simply held and until the next round. The network attempts to ease the exchange of information, lower transaction costs and make it possible for both carriers and shippers to access larger markets.

Kwon et al. (2005) proposed an iterative auction mechanism for TL transportation procurement. Each agent (carrier) bids for a package of lanes. A descending multi-round format is used to allocate the lane packages to agents. Agents compute their preferred packages based on their cost structures and submit them to the auctioneer. The auctioneer then performs a provisional allocation of lanes to the agents by solving a winner determination (WD) problem with the objective of minimizing the payments. Simulation results have indicated that both carriers and shippers reduced their cost through a better collaboration.

For the LTL (less than truckload) setting, Krajewska et al. (2006) proposed an auction model for the collaboration among individual freight forwarding entities. Cooperating forwarders exchange their orders through a combinatorial auction. The auction is individually rational, which means each individual partner increases their profit by participating in the coalition.

Effective collaboration among agents in a distributed system leads to a better utilization of resources and, thus, greater efficiency and profit for the whole system. However, before entering into the partnership, agents have to agree upon how to share the profit that results from the collaboration. In a collaborative environment where, for example, carrier companies belong to a common holding organization, profit sharing may not require incentive-compatible mechanisms. Gujo et al. (2009) proposed an exchange mechanism, called ComEx, for inter-enterprise logistic services. In ComEx, transportation capacity in each division is managed by a profit centre which can exchange delivery orders with other profit centres based on the geographical zones and time windows of the orders. The profit gained is shared proportionally among the profit centres based on the cost savings of each centre that participated in the exchange. A profit precondition for this type of sharing is that ComEx has access to the cost saving data of the profit centres. ComEx works well in a collaborative setting. However it is not suitable for game theoretic settings where profit centres do not belong to a common holding organization and so they may be reluctant to share their cost savings data. In this case, a profit distribution mechanism based on game theory and combinatorial auction should be applied (Krajewska et al., 2006; Gomber et al., 1997). Other agent-based models in transportation services distribute the benefits of collaboration from a loss-sharing rather than a profit-sharing perspective (Schönberger, 2005; Schönsleben et al., 2004). Krajewska et al. (2006) presents an overview of these benefit sharing models.

2.4.4 Computing

Modern computing services aggregate a large number of independent computing and communication resources and data stores. They are built onto the bases of distributed computing, grid computing and virtualization. A computing service environment is inherently complex, heterogeneous and dynamic. Service resource management systems need to provide mechanisms and tools that allow resource consumers (end users) and providers (resource owners) to express their requirements and facilitate the realization of their goals. This objective necessitates seamless scheduling of providers' resources to support the dynamic scaling of user activities across multiple domains. Scheduling computing services under varying loads, diverse application requirements and heterogeneous systems is a challenging problem. An agent-based approach can be an effective way to realize information sharing, given the unpredictable dynamism and increasing heterogeneity in computing service scheduling.

With the aim of tackling the challenge of dynamic environments in computing services, An et al. (2010) proposed a distributed negotiation mechanism for dynamic and uncertain resource demand and supply in computing as a service (cloud computing) platforms. The mechanism is an extension to the alternating offers protocol with the added feature of allowing agents to decommit from contracts at a cost. The mechanism facilitates the agents' negotiation over both a contract price and a decommitment penalty. They evaluated their approach experimentally using representative scenarios and workloads, which showed that their model achieves a higher level of social welfare compared to both combinatorial auctions and the fixed-price model used by Amazon's EC2.

Scheduling mechanisms for computing services typically deal with the dynamics of both resource and service markets. Sim (2012) proposed a concurrent negotiation mechanism for agents to negotiate in multiple interrelated e-Markets. He developed an agent-based test bed consisting of provider agents and consumer agents acting on behalf of resource providers and consumers, respectively, along with a set of broker agents. The mechanism consists of: (1) a bargaining-position-estimation strategy for the multilateral negotiations between consumer and broker agents in a service market, and (2) a regression-based coordination strategy for concurrent negotiations between broker and provider agents in resource markets. The negotiation outcomes between broker and provider agents in a resource market can potentially influence the negotiation outcomes between broker and consumer agents in a service market. Using this mechanism, the broker agent accepts service requests from consumer agents, and purchases resources from provider agents. The collection of resources that satisfy consumer agents' requirements is composed dynamically. Mobile agents are also designed in this way, providing scalability in cloud computing. In Singh and Malhotra (2012), a mobile agent is capable of transporting its state from one environment to another with its data intact and able to perform appropriately in its new environment. The agents are supported with algorithms to search for another cloud with better response time when the most-approachable cloud becomes overloaded.

To deal with the challenge of customer's private information, game-theoretic based methods have been proposed to solve the resource allocation problem in network systems. Gagliano et al. (1995) presented an auction allocation of computing resources. In their proposed auction, computing tasks are provided sufficient intelligence to acquire resources by offering, bidding and exchanging them for funds. Wolski et al. (2001) compared commodities markets and auctions in grids in terms of price stability and market equilibrium. Zaman and Grosu (2011) studied and implemented combinatorial auction-based mechanisms for efficient provisioning and allocation of computing services (VM instances) in cloud computing environments, with the objectives of maximizing the revenue of the service provider and providing an efficient allocation of resources. A recent survey on market-oriented resource management and scheduling in computing services can be found in Garg and Buyya (2011).

Table 2-1 summarizes the agent-based scheduling approaches aimed at addressing service scheduling challenges.

Table 2-1 Agent-based scheduling approaches that address service scheduling challenges

Agent-based service scheduling approaches	Dynamic environment	objective	Private information
Franzin et al., (2002), P. Modi et al., (2005), Crawford et al., (2004), Krishna and Ünver (2004), Grano et al., (2009)	Not addressed	Maximize the social welfare by maximizing the customer's satisfaction	Addressed
Hassine et al., (2004), T.Sönmez and M.Ünver (2006),	Addressed	Maximizing the global utility and ensuring near fulfillment of customers' preferences	Not Addressed
Zhiming(2011)	Addressed	Minimize the cost of the operation rooms including the overtime cost	Not Addressed
Wainer et al., (2007)	Not Addressed	Minimize the cost of the operation rooms including the overtime cost	Addressed

Muller et al., (2001), Hosseini et al., (2011)	Not addressed	Maximize patient satisfaction by considering patient preferences and minimize patient waiting time between appointments	Addressed
Paulussen et al., (2003)	Addressed	Maximize the resource utilization and minimize patient's hospital stay time	Not addressed
Meisels et al. (2003), Hannebauer et al. (2001)	Not addressed	Minimize the patients' waiting time between appointments	Not addressed
Kwon et al., (2005), Krajewska et al. (2006), Sheffi, 2004, Song et al. (2003)	Not addressed	Minimize the total shipping cost	Addressed
An et al. (2010), Sim (2012)	Addressed	Maximize the social welfare	Not addressed
Gagliano et al. (1995), Zaman and Grosu (2011), Wolski et al. (2001)	Not addressed	Maximizing the revenue of the service provider as well as providing an efficient allocation of resources.	Addressed

2.5 System Design Issues

Adopting the agent-based approach has made it possible to model the challenges of a distributed environment and complicated multiple objectives in service scheduling naturally in the agent-oriented architecture. The main design issue is how to design agent-based scheduling systems such that they can effectively address the challenges of a dynamic scheduling environment and the presence of customers' private information. The previous section reviewed typical agent-based scheduling approaches aimed at addressing these challenges from a domain-specific perspective. This section presents a summary of the existing agent-based service scheduling approaches from the system design perspective and identifies some promising research opportunities.

2.5.1 System structures

Most of the available agent-based service scheduling system designs adopt the physical decomposition approach for agent encapsulation. Service providers who control the service resources are modelled as provider agents, and users who request services are modelled as

customer agents. In some cases, such as carrier collaboration in transportation services, a service provider can also request services from other providers. In this situation, a service provider will play the role of provider agent as well as that of customer agent. Given the agent encapsulation scheme, agent system architectures provide the organizing framework within which agents interact with each other. In the context of agent-based service scheduling, two types of system structures are usually adopted, namely mediated structure and autonomous structure. A mediated structure utilizes a mediator to coordinate the allocation of resources to users. A service provider agent often assumes the role of mediator. For example, in healthcare scheduling, provider (resource) agents usually take the role of mediator and coordinate the resource allocation among patients (Paulussen et al., 2003; Hannebauer and Muller, 2001; Hosseini et al., 2001).
Autonomous structure appears in the settings, where a service provider also requires services from other providers, that is, an agent is both a provider and a customer. In autonomous structure, interactions between agents are not coordinated by mediator agents. Instead, agents optimize their schedules by exchanging their resources (Krajewska and Kopfer, 2006, Gujo et al., 2009). In some service scheduling settings, such as meeting scheduling or workforce scheduling, there are no explicit resource times to be allocated. Instead, the main issue is to find a meeting time or work schedule which is agreeable to all participants. For example, in Becker and Hans (2006), agents representing operation room staffs negotiate with each other, based on the Nash bargaining solution, to schedule their work shifts. Autonomous structure is also often used in agent-based meeting scheduling applications (Hassine et al., 2004, Modi et al., 2004, and Franzin et al., 2002).

2.5.2 Negotiation mechanisms

Given its inherently decentralized nature, agent-based service scheduling must coordinate agents' behaviour using some type of negotiation protocol. The most commonly used protocols are the Contract Net protocol (CNP) and economic based models, such as auctions. CNP is essentially a general tendering procedure. However, unlike auctions, the awarding decision may not be related to price or cost factors. To summarize, in the CNP, each agent (manager) with work to subcontract broadcasts a call for bidding messages and waits for other agents (contractors) to send back their bids. After receiving bids from all the agents or waiting for a certain time period, the manager evaluates the bids received based on the evaluation criteria and awards its contracts to one or more contractors, which then process the subtask. CNP coordinates task allocation, providing dynamic allocation and natural load balancing. Unlike general equilibrium market mechanisms or auctions, which usually require a mediator, contract nets are purely distributed models, in which any agent can act as a manager and subcontract tasks to other agents. CNP can easily be embedded into the autonomous system structure and is suitable for distributed dynamic scheduling. For example, in Zhiming (2011), CNP is used to dynamically reallocate tasks among agents in an operation room scheduling setting. The drawback of CNP is that there is no built-in mechanism to motivate agents to reveal their private information. Therefore, it is not sufficient in service scheduling settings where customers' private information is present.

Auctions can accommodate customers' preferences to minimally reveal their private information by providing appropriate incentives to customers. There is a wealth of literature on auction design. Different auction formats such as sequential auctions, simultaneous auctions and combinatorial auctions have been studied extensively.

Agent-based service scheduling usually uses combinatorial auctions (also called bundle auctions), because scheduling is, in its essence, a combinatorial optimization problem. Typical examples include various implementations of VCG auctions (Crawford & Veloso, 2004; Sheffi, 2004; Berger and Bierwirth, 2010). However, due to high computational complexity, VCG is not practical for large-scale problems, especially in dynamic environments. To provide better responsiveness, sequential auctions, simultaneous auctions and iterative implementations of combinatorial auctions have also been adopted in service scheduling (Paulussen et al., 2003; Song and Regan, 2003; Sönmez & Ünver, 2007; Kwon et al., 2005; Gujo et al. 2009). These auction models are compared and their applicability to agent-based service scheduling is analysed in the following subsection.

2.6 Research opportunities

This chapter provides a survey on system design for service process scheduling, covering several representative service domains. The approaches reviewed here focus on either dynamic scheduling environments or customers' private information. These approaches may not be sufficient for many real world service scheduling applications, as they usually deal with only part of the challenges. Based on this survey, as well as on first-hand research and development experience in this area, I believe that further research on an integrated approach that tackles service scheduling challenges concurrently is very much needed. While there is no built-in mechanism in CNP to address customers' private information, a logical step to the integrated approach is to design auctions which can accommodate dynamic changes and handle bundles of resource requirements in service scheduling. The key issue is how to deal with the enormous computational complexities of combinatorial auctions in dynamic environments.

In general auction terms, combinatorial auctions (CA) allow bidders to place bids on bundles of items. It addresses bundle preferences explicitly. However, the computations required to solve difficult valuation problems and winner-determination problems can be prohibitive. In general, CAs are likely to be practical for smaller-sized problems. In addition, CAs require that a complete valuation on alternative schedules be revealed to the auctioneer. In service scheduling, customers are often reluctant to make a complete revelation to prevent any information from leaking out and adversely affecting other decisions or negotiations. Lack of transparency is another practical concern in CAs. It can be difficult to explain to customers why a certain schedule has been selected.

Iterative bundle auctions are iterative implementations of CAs. This class of auction has practical significance because it addresses the computational and informational complexities of CAs by allowing bidders to reveal their preference information only as necessary as the auction proceeds, and bidders are not required to submit (and compute) complete and exact information about their private valuations. In many cases, iterative auctions present better computational and privacy properties than those of CAs. In addition, iterative auctions have the potential to accommodate dynamic events, which is an important requirement in service scheduling applications. With a careful design of the structure and its components, iterative bundle auctions have the potential to significantly reduce computational costs while accommodating the dynamic environment and users' private information in service scheduling.

Differently from how CAs and their iterative implementations price bundles, sequential and simultaneous auctions price bundles as the sum price of the individual items. However, they do not allow bidders to bid on bundles of items. Sequential auctions suppose that the set of items is auctioned in sequence. Bidders bid for items in a specific known order and can choose how much (and whether) to bid for an item depending on past successes, failures, prices and so on. Sequential auctions are particularly useful in situations where setting up combinatorial or simultaneous auctions is not feasible. Simultaneous auctions sell multiple items simultaneously in separate markets. Bidders have to interact with simultaneous but distinct markets in order to obtain a combination of items sufficient to accomplish their task. Real-world markets quite typically operate separately and concurrently despite significant interactions in their preferences. Sequential and simultaneous auctions tackle the complementarities over resources in the spirit of general equilibrium theory. These auctions fail when there are no prices that support an efficient solution (the existence problem) and when agents bid cautiously to avoid purchasing an incomplete bundle (the exposure problem). However, given that these auctions are more practical in terms of computation, they are important models worthy of further study.

In addition to the design of core negotiation mechanisms, other research needs remain to be addressed in agent-based service scheduling. For example, there is a lack of systematic analysis and comparison on how system design factors affect computational time in agent-based service scheduling systems. To adequately test and evaluate various approaches, benchmark problems are also needed. Furthermore, the systems must be designed to integrate a wide range of real-time information and uncertain parameters into the dynamic service scheduling process. Unlike the auction designs found in the literature, dynamic pricing cannot be applied to some services, such as healthcare and government services. These settings require bidding-based service scheduling systems without dynamic pricing. This would also be an interesting research topic for auction design in general.

2.7 Summary

Service scheduling systems are inherently distributed and dynamic. The presence of customers' private information imposes additional challenges in finding high quality solutions. Agent-based systems can be an appropriate approach to service scheduling due to their distributed and autonomous nature. This chapter reviewed agent-based scheduling approaches in representative service domains through the lens of how they address the challenges of service scheduling. Despite the many domain-specific design applications in agent-based service scheduling, there is a lack of general problem formulations, classifications and solution frameworks. Constructing these general models for service scheduling will greatly facilitate the collaboration of researchers in this area and guide the effective development of integrated service scheduling systems. Moreover, the applicability of a service scheduling approach to industrial settings will largely depend on how it copes with distributed and dynamic environments and on how it computes high-quality solutions despite the presence of customers' private information.

The position taken in this thesis is to develop service scheduling approaches based on an iterative implementation of VCG auction. Since agents are not required to submit (and compute) complete and exact information about their private valuations, in many cases, iterative auctions present improved computational and privacy properties. In addition, iterative auctions have the potential to support dynamic scheduling, a common requirement in service scheduling. By carefully investigating the features of iterative combinatorial auctions and the nature of service scheduling problems, an effective and practical auction-based service scheduling approach can be developed. Compared to the existing agent-based scheduling literature, this work is focused on an integrated framework that simultaneously addresses a dynamic distributed environment and customers' private information. In addition, the framework can accommodate complicated objective functions into the service scheduling process.

Chapter 3 The Service Scheduling Problem

Compared with traditional manufacturing scheduling, service scheduling poses additional challenges attributable to the significant customer involvement in service processes. Service scheduling should be generated in a distributed environment where the scheduling knowledge is distributed among customers and service providers.

Customers have jobs that need to be processed. In order to have their jobs processed, customers need to consume the processing time of the service providers' resources. The price that customers are willing to pay to a service provider and their preferences regarding the allocated service time slot can be their private information and they may be reluctant to fully reveal that information to a provider. Customers may behave strategically to protect their private information attributed to the different objectives of service providers and customers. With the advances in information technology, service scheduling has become a common requirement in many real-world automated trading systems. Clarifying the theoretical underpinnings and practical solutions to the problem would both be very much appreciated in this field.

This chapter describes the general service scheduling model studied in this thesis. The properties of a general Distributed Service Scheduling Problem (DSSP) are described, and the DSSP is then modeled as a game. A Vickrey-Clarke-Groves (VCG) auction that solves the game is constructed, and the computational challenges of applying this VCG auction to the DSSP are discussed.

3.1 Properties of a DSSP

The first property of a DSSP is that it is a distributed scheduling environment. In the context of this thesis, the distributed environment is specified using a description from Ghenniwa (1996): a distributed environment is constructed from entities that are able to perform some functions *independently* and exercise some degree of *authority* in sharing such capabilities. Such entities are put to work in the same spatial-time domain to achieve either a common goal or separate goals. As mentioned earlier, to recognize the *independent* and *autonomous* nature of the entities, they are treated as agents. In a distributed environment, there are situations where knowledge about a scheduling problem, e.g. customers' availability and preference information, is distributed among agents and the overall problem knowledge does not reside in a single agent.

[Definition 3.1 Distributed Scheduling Environment] A distributed scheduling environment is an environment where the knowledge of a problem is distributed among agents and no single agent has a global view of the problem (distribution of knowledge).

The second property of a DSSP is that it is a dynamic environment. Customers' involvement in service production may cause knowledge about scheduling problem may change over time. Uncertainty in customer demand, uncertainty in service time duration, customer cancelation and no-shows, and changes in customer preferences are some examples of dynamic changes in a service scheduling environment.

[Definition 3.2 Dynamic Scheduling Environment] A dynamic scheduling environment is an environment where the knowledge of the scheduling problem may change over time.

The third property of a DSSP is the presence of customer's private information. As mentioned earlier, in order for a service to be produced, a customer has to be present personally or he/she must present his/her property or information. Service scheduling should therefore be generated by considering that a customer's inputs are available for the service process. Customers' preferences regarding the timing of delivering their inputs would then be considered in service scheduling. However, a customers' availability may very well be their private information and they could behave strategically to protect that private information. In addition to a customers' availability, which is (almost always) their private information, the price that a customer is willing to pay to a service provider for a given service time slot is also their private information. Customers are motivated to not reveal the highest price they are willing to pay to the service provider, and in most cases, these prices are considered to be very sensitive private information that they are reluctant to reveal.

[Definition 3.3 Customer's private information] In a DSSP, customers may not want to reveal some information (e.g. the value that customers give to different scheduling alternatives, a customer's full availability). They may act strategically to protect their private information.

The fourth property of a DSSP is that there are complicated, perhaps conflicting objectives. The distributed service scheduling environment enables each agent to have their own scheduling objectives. The objectives of these agents can vary from one to another. In addition to multiple objectives, since agents are self-interested, they are likely to behave strategically to achieve their own objectives without considering the global objectives of the system.

[Definition 3.4 Complicated objectives] The objective in a service scheduling problem is a combination of multiple objectives, each from an individual agent. These may be in conflict with each other, and each agent may behave strategically to advance their own objective. This characteristic derives from the self-interested nature of agents in this environment.

3.2 Centralized Formulation

In a DSSP, customers have private information: their actual valuations of different scheduling alternatives, such as completion times, are part of their private information, which is not known to the provider. However, to clearly demonstrate the combinatorial optimization nature of the problem, one can first assume a centralized environment, i.e., where customers' valuations are known to the provider. With this assumption, the problem can be conveniently modeled as a mixed integer program. The decentralized characteristic of the problem will be considered during the development of the game's theoretic modeling.

Consider a set of customers and a service provider. The service provider can provide a set of different services, with a limited capacity for each time slot. A customer has a request which can be a combination of different services. The customers compete with each other to schedule their own requests according to their respective objectives. Each customer's value of a schedule (i.e., the price that she is willing to pay for the request to be completed at a specific time) is their private information. Each customer is motivated by their own objectives and is not controlled by other customers or by a system-wide authority. The service scheduling problem involves the allocation of service time slots to the customer's requests, such that a provider's capacity constraints are satisfied and the sum of customers' values is maximized.

Formally, the DSSP problem consists of a set of n customers and a service provider that can provide a set of m services. Within the scheduling horizon, each service $i(i = 1, \ldots m)$ has a sequence of service time slots $l_{i,k}(k = 1, \ldots, m_i)$ available for processing customers' service requests. Let L be the set of all available time slots $L = \sum_{i=1}^{m} \sum_{k=1}^{m_i} l_{i,k}$. For each service time slot $l_{i,k} \in L$, its capacity is limited by $capacity(i, k)$, which means that no more than $capacity(i, k)$ number of customers can be served within $l_{i,k}$. Each customer $j(j = 1, \ldots n)$ has a request, which is a combination of different services provided by the provider. A customer may need a bundle of time slots to process their request.
For a bundle $B \subseteq L$ that contains an allocation of the provider's service time slots to a customer's request, that customer will have a valuation on B. Let $v_j(B)$ be the value customer j attaches to the time slot bundle B. This thesis follows the *private value model* introduced by Vickrey (1961). Therefore, a customer has a value for each $B \subseteq L$, and these values do not depend on the private information of the other customers. Each customer knows his or her own values, but not the values of others.

Let $x_j(B) = 1$ if B is allocated to customer j , and be equal to zero otherwise. The DSSP problem involves the selection of a set of time slot bundles for customers such that the service provider's capacity constraints are respected and, at the same time, the sum of customer value (social welfare, in terms of microeconomics) derived from the selected bundles is maximized. The problem can be formulated as the following integer programming.

$max \sum_{j=1}^{n} \sum_{B \in L} x_j(B) v_j(B)$

Subject to

$\sum_{B \in L} x_j(B) \leq 1, \quad j = 1, \ldots, n$ (1)

$\sum_{B \ni l_{i,k}} \sum_{j=1}^{n} x_j(B) \leq capacity(i, k),\ i = 1 \ldots m;\ k = 1 \ldots m_i$ (2)

$x_j(B) = \{0,1\},\ \ B \in L, \quad j = 1, \ldots, n$ (3)

Constraints (1) ensure that a customer can only obtain one bundle of time slots. Constraints (2) ensure that the allocation of a time slot to customers does not exceed the capacity limit of the service time slot. Constraints (3) are a set of integer constraints. The centralized formulation of the DSSP problem is NP-hard, as stated in the following theorem.

Theorem 1: *The centralized formulation of the DSSP problem is NP-hard.*

Proof: To show that the centralized formulation of DSSP is NP-hard, consider a special case in which $capacity(i,k) = 1,$ for all $i = 1 \dots m;\ k = 1 \dots m_i$. In this case, the special case is a set packing problem, which is NP-complete (Karp, 1972). It follows that, as a general case, the centralized formulation of DSSP problem is NP-hard■.

3.3 Game theoretic modelling and VCG auction construction

Centralized formulation in the previous section was created by assuming that customers' valuations are known to the service provider. This assumption should be removed in the game theory modeling. Since customer's valuations are their private information they may behave strategically to maximize their own benefits.

The solution to this challenge is to design mechanisms that will induce agents to behave so that a certain outcome prevails. In other words, to provide incentives to the agents in the system such that they behave in a way that is prescribed by the system.

As the computational complexities inherited from the combinatorial nature of the scheduling problem are not related to the game theoretical modeling, the scheduling details can be ignored to focus only on strategic interactions. The DSSP is first modeled as a game, and then a Vickrey-Clarke-Groves (VCG) auction that solves the game with an economically efficient outcome is constructed.

Let N be the set of n customer agents. Each agent has a service request from a service provider. Service requests need to be scheduled on the service provider's resources. Let Ω be the set of all feasible schedules. Each feasible schedule determines the allocations of the time slots of service resources to the customers' request. For each schedule $S \epsilon \Omega$, each agent j has a monetary value $V_j(S)$. A value is the maximum price that an agent is willing to pay to process its service request as scheduled in S. $P_j(S)$ is the price the agent j, needs to pay to the service provider in exchange for processing its request. It is payoff on schedule S is $v_j(S) - p_j(S)$. Agents' objective is to maximize their payoffs and service provider objective is to maximize market efficiency. The goal here is to design a mechanism that select a schedule which maximize the sum of agent's valuations. A VCG auction is constructed here for the service scheduling problem.

Let V^* be the maximum of the total values of all agents that can be obtained by a schedule in Ω and $\widetilde{V}$ be the maximum of the total values of agents when j is excluded from the schedule.

$$V^* = max_{S \in \Omega} \sum_{j=1}^{j=n} V_j(S)$$

$$\widetilde{V} = max_{S \in \Omega} \sum_{j \in N \setminus j} V_j(S)$$

When the auction starts, agents submit their values for all feasible schedules Ω. If an agent's request was not included in a schedule, the agent's valuation on the schedule will be zero. After receiving agents' value, the service provider (auctioneer) chooses the final schedule S^* from Ω in a way that it solves V^*. Service provider also generate a schedule for each agent, in a way that the schedule solves $\widetilde{V}$. After the schedules are computed, the amount that agent j pays for final schedule S^* is $p_j(S^*) = \widetilde{V} - [V^* - V_j(S^*)]$ and agent j's payoff from participating in the auction is
$V_j(S^*) - p_j(S^*) = V_j(S^*) - (\widetilde{V} - [V^* - V_j(S^*)]) = V^* - \widetilde{V}$. It is clear that $V^* \geq \widetilde{V}$, which means agents always get non negative payoff when they participate in the auction. In addition to motivate agents to participate, the designed auction is also incentive-compatible. A mechanism is incentive-compatible if it is every agent's dominant strategy to reveal their private values truthfully.

Theorem 2: *Given the auction constructed for the game theoretic model of DSSP, for all customers, submitting truth valuations to the auctioneer is a dominant strategy.*

Proof: Suppose agent j reports w_j as its value instead of v_j, $w_j \neq v_j$. Service provider then chooses $\widetilde{S} \in \Omega$ as a final schedule by solving $max_{S \in \Omega}[\sum_{i \neq j} v_i(S) + w_j(S)]$.

Agent j's payoff then becomes

$$V_j(\tilde{S}) - P_j(\tilde{S}) = V_j(\tilde{S}) - [\,\tilde{V} - \textstyle\sum_{i\neq j} v_i(\tilde{S})] = \sum_{i\neq j} v_i(\tilde{S}) + V_j(\tilde{S}) - \tilde{V} \leq V^* - \tilde{V}.$$

From the above formula it is obvious that agents will not get benefit by misreporting their valuations.

By using centralized formulation of service scheduling problem the set of all feasible schedules Ω can be obtained and by using constructed VCG auction the optimal schedule in Ω can be found. It appears that everything needed to solve the DSSP have been found. However in reality there are several implementation limitations in applying VCG auction to DSSP. The limitations can be described from three different perspectives; service provider (auctioneer), customer agents, and system requirements.

From the customer agents' side, there is a valuation complexity. For each agent valuation complexity refers to the effort needs to determine its values over an exponential number of schedules in Ω.

Form service provider' side, there is a high computational complexity. In the VCG auction, service provider needs to find the solution of V^* and $\tilde{V}$ for all the agents, which means n+1 NP-hard optimization problems. It is obvious that if the VCG auction applied to non-trivial sized problems the computation cost can be prohibitively expensive. More importantly, service provider needs each customer's complete valuation on the alternative schedules. In service scheduling environment, customers are reluctant to reveal their complete valuations. They fear that their information could leak out and adversely affect the service provider decisions.

From system requirement's side, there is a high communication complexity. VCG auction requires a large number of schedules communicated between service provider and agents. In addition, VCG auction does not support the dynamic changes (e.g. changes in the number of customers and their valuations on different schedules) that occur during an auction. Once an auction starts, customers should be ready to submit their complete valuations on alternative schedules. In many service scheduling environments it is not practical to ask all customers to be prepared to start the auction at a predefined time. The VCG auction does not have the potential to accommodate a customer arriving during an auction. In the following chapter, we propose an iterative bidding framework aimed at addressing the limitations arising in the application of the VCG auction to a DSSP.

3.4 Summary

This chapter defines the properties of the general DSSP and it provides the centralized formulation of a general service scheduling problem. In a DSSP, the knowledge of a scheduling problem is distributed among agents, and the strategies of agents cannot be controlled by outside parties, such as other agents in the environment. In this situation, agents can be assumed to perform strategically in service of their own objectives. The solution to this challenge is to design mechanisms to induce agents to behave such that a certain outcome prevails. For this purpose, I first modeled the service scheduling problem as a game and then constructed a VCG auction as a mechanism design that solves the game with an economically efficient outcome. However, the VCG auction's limitations, in terms of implementation, restrict its application to DSSPs. The next chapter shows how the computational and communication complexity issues derived from applying VCG auctions to DSSPs are addressed by proposing an iterative bidding framework.

Chapter 4 Iterative Bidding Framework for Distributed Service Scheduling

This chapter presents a solution framework for DSSP. Chapter 3 showed how applying VCG auction directly to DSSP requires every agent to reveal its valuation of all of the feasible schedules, and the auctioneer need to solve a sequence of NP-hard optimization problem to determine the outcome, which is computationally expensive requirement. In addition, VCG is a one-shot auction that does not accommodate dynamic changes during the bidding processes. The Iterative Bidding Scheduling Framework (I*b*SCHF) proposed in this chapter addresses the inherent limitations that arise when applying VCG to DSSP. I*b*SCHF is an iterative combinatorial auction-based approach to the DSSP. Iterative combinatorial auctions are indirect implementations of VCG auction and it addresses the computational and informational complexity of VCG. In this class of auction agents are not required to submit (and compute) complete and exact information about their private values. Agents are allowed to reveal their preference information as it becomes necessary, as the auction proceeds. Typical examples of iterative combinatorial auction include Parkes and Ungar, 2000, Parkes and Kalagnanam, (2005), Bikhchandani and Ostroy, (2006).

Parkes and Ungar (2000) proposed *i*Bundle, an iterative combinatorial auction for the combinatorial allocation problem. *i*Bundle computes the efficient resource allocation when agents follow a myopic best-response bidding strategy, bidding for the items that maximize their surplus taking the current price as being fixed. Their approach solves the combinatorial allocation problem without requiring complete information revelation from agents. A comprehensive survey for combinatorial auctions can be found in Vries and Vohra (2003).

The above-mentioned combinatorial auctions are not designed for scheduling problem they are designed for general combinatorial allocation problem. However they can be applied to scheduling problem by exploring the specific problem characteristics derived from the scheduling domain.

In the literature there are few combinatorial auctions that designed specifically for scheduling problems. Combinatorial auction for job shop scheduling problem is applied by Kutanoglu and Wu, (1999) with the focus on investigating the links between combinatorial auctions and Lagrangean relaxation. The properties of several auction protocols are investigated in the context of decentralized scheduling by Wellman et al., (2001).

An iterative combinatorial auction-based framework for a particular type of scheduling problem, DSSP, is presented in this chapter. In addition to addressing the computational complexities of applying VCG auction to DSSP, the framework has the potential of accommodating dynamic changes in the scheduling environment. The chapter is organized as follows. Section 4.1 describes the IbSCHF. Section 4.2 validates the effectiveness of this framework through a computational study. Section 4.3 explains how the proposed framework can be applied to accommodate dynamic changes, and section 4.4 summarizes the chapter.

4.1 The IbSCHF

A key challenge in the development of solutions to DSSP is the design of a mechanism that allocates limited service capacities to customers, such that the overall value of customers is maximized despite the self-interest of individual agents. Auctions have long been considered an effective way of allocating limited resources to competing users and to discover market prices for products and services. In recent years, the pervasive inter-connectivity provided by the Internet has made auctions a popular mechanism that directly links the capacities of service providers with end customers. In this chapter, we present an auction-based framework for DSSP. The auction is implemented using an iterative bidding protocol, which can be seen as a collaborative negotiation procedure between the provider and customers. This iterative bidding protocol is described below.

4.1.1 Iterative Bidding Protocol

The iterative bidding protocol is a price mechanism in which a service provider balances the request requirements among customers by adjusting the prices of time slot bundles. The protocol adopts the non-anonymous bundle price structure, under which a customer's bid is represented as a tuple $\langle bundle,\ bidding\ price\rangle$, where $bundle$ is the set of time slots that the customer wants, and the $bidding\ price$ is the price that the customer is willing to pay for the services to be delivered during those specified time slots. The bidding price is customer-dependent. There is no common public price for a bundle. A non-anonymous price structure allows providers to price the same bundle differently for different customers, which is a common practice in many service industries. The bidding procedure consists of four components: initialization, price update and bidding, termination checking, and winner determination.

4.1.2 Initialization

Before the bidding starts, the service provider first presents the set of available services and available time slots for each service to the customers. For each service time slot $l_{i,k}(k = 1, \dots, m_i)$, the provider sets up a base cost. Customers then compute their respective sets of feasible bundles. For each feasible bundle, a customer computes the value attached to it based on the utility derived from that bundle. The initial bidding price of a bundle is the cost of the bundle, which is calculated by adding up the base costs of the service time slots included in the bundle. Knowing the values and initial bidding prices of bundles, a customer computes the payoff of each bundle. I assume a private value model (Vickrey, 1961) for all customers. Under this model, each customer has a value for their bundles. A customer's payoff for a bundle is the difference between their value for a bundle and its bidding price. To maintain a positive payoff, the customer is willing to pay up to their value to get the bundle. After obtaining the payoffs for their feasible bundles, the customer selects the bundle with the highest payoff (breaking ties randomly) as their first bundle to bid on.

4.1.3 Price Updating and Bidding

At the beginning of round t $(t > 1)$, customer agents must update their bidding prices for the bundles they submitted in round $t - 1$. Customer agents update their bidding prices based on the provisional allocation that resulted from the winner determination at round $t - 1$. Customer agents have three price updating option at round t if their bid was not awarded in the provisional allocation at round $t - 1$: (1) they can increase their bidding prices by ε on the bundle they bid for at round $t-1$ or rounds before $t-1$. Here ε is the minimum price increment imposed by the service provider. In general customer agents do not increase their bids more than ε, because they are rational in maximizing their utilities. However, they are allowed to increase their bid with higher value than ε ; (2) they can take an ε discount and keep the bidding prices unchanged. If a customer agent takes this ε discount, the service provider will consider that the customer has entered into *final bid status.* A customer agent with final bid status is forbidden from increasing the bidding prices on any of their bundles in future rounds; and (3) they can, of course, withdraw from the bidding process.

A customer agent can keep its bidding price unchanged at round $t,$ if it is included in the provisional allocation at round $t-1$.

After updating their bidding prices, a customer agent needs to compute their set of utility-maximizing bundles. In computing such a set, customer agent j solves a maximization problem $max_{B \in E_j}\left[v_j(B) - p_j^t(B)\right]$ and obtains the set of bundles that equally maximizes their utility, where $p_j^t(B)$ is the bidding price for B at round t. That is, for any two bundles B and B' in the utility-maximizing set, $v_j(B) - p_j^t(B)$=$v_j(B') - p_j^t(B')$. After obtaining the set of utility-maximizing bundles, the customer randomly picks one and submits it to the provider with the updated bidding price.

If a customer agent has entered into final bid status, it is not allowed to increase its bidding price anymore. However the service provider can allow customer agents to repeat their final bid in future rounds until termination. Final bid repeating can enhance efficiency and increase service provider's revenue. The reason is that, in each bidding iteration it is possible that provisional allocation changes due to newly received bids with higher values. Such changes in provisional allocations may allow the space for allocation of previously submitted bids that have been temporarily excluded. In the absence of final bid repeating, a customer agent will not acquire its required resources even if the future bids make the capacities available for acceptance of its final bid.

4.1.4 Bid Screening and Termination

After XOR-Bids have been received from the customer agents, the service provider first screens out invalid bids; a bid is considered as invalid when: (1) it includes increased price form a customer agent who has already declared his final bidding status in previous rounds; or (2) it includes bidding price for a bundle that is below the highest bidding price for that bundle received in previous rounds. Invalid bids will not be considered in the following winner determination procedure.

On this stage, the service provider determines if the termination condition is satisfied based on the valid bids in this round. Termination condition examines if all the customers have repeated their bids in the last round, i.e. the price of none of the valid bids are updates from the last round. If termination condition is not satisfied, the winner determination model should be solved with the valid bids as input. Otherwise, the customers will be informed about the final allocation and they will be charged based on their bidding prices.

4.1.5 Winner Determination

The service provider needs to compute a new provisional allocation in each round as long as the bidding is not terminated. The winner determination model selects a subset of the bids submitted by the customers such that the overall bidding price of the provisional allocation is maximized and the capacity constraints of the provider are not violated. Let N^t be the set of customers that submitted their bids at round t , and $p_j^t\left(B_j^t\right)$ be the bidding price of customer j at round $t, j \in N^t$, where B_j^t is the bundle submitted by customer j at round t. Let $Z_j = 1$ if customer j wins and $Z_j = 0$ otherwise. The winner determination model can be expressed using the following integer programming.

$$max \sum_{j \in N^t} Z_j p_j^t\left(B_j^t\right)$$

Subject to

$$\sum_{\substack{j \in N^t \\ B_j^t \ni l_{i,k}}} Z_j \leq capacity(i,k),\ i = 1 \dots m;\ k = 1 \dots m_i \qquad (1)$$

$$Z_j = \{0,1\},\quad j \in N^t \qquad (2)$$

Constraints (1) ensure that the bids awarded in a provisional allocation do not violate the provider's capacity constraints. Constraint (2) is a set of integer constraints.

The winner determination problem is a general form of the set packing problem, which is NP-hard. The commercial optimization package ILOG CPLEX 12.0 is used to solve the winner determination problem. Although winner determination problems in combinatorial auctions are generally NP-hard, many of them can be solved quickly by modern optimization algorithms, up to fairly large sizes. Anderson et al. (2000) report that CPLEX 6.5 performs very well in terms of running time for many of the common winner determination problem benchmark distributions. The solving speed is comparable to the special-purpose winner determination algorithms, such as those in Fujishima et al. (1999) and Sandholm (2002). Sandholm et al. (2005) show that some winner determination distributions, with thousands of bids, can be solved by CPLEX 8.0 within a couple of seconds.

4.1.6 Implementation considerations

The efficiency of auctions largely depends on the level of competition among customers. The Internet provides pervasive accessibility to virtually any electronic market; customers may come at quite varied times. To aggregate demand and facilitate competition, Internet auctions usually span a couple of days or even longer. Customers can enter the auction and place bids at any time before the auction ends. To spare customers the task of continuously monitoring the bidding process and repeatedly placing their bids, Internet auctions allow bidders to provide direct value information to an automated bidding agent, called a proxy agent, which bids on behalf of the customer.

In the I*b*SCHF for DSSP, a proxy agent can be designed to manage a set of feasible bundles for the customer, and decides which bundle to submit, at which round, and at what price. The customer should therefore inform the agent regarding the value it places on each of the feasible bundles. Meanwhile, the agent should be equipped with the algorithm used to update bidding prices and select the payoff maximization bundle along the bidding process. If the customer prefers, the agent can also inform the customer regarding the bidding status and allow the customer to update the values before the auction ends. For easy access, customers may install the proxy agent on a personal computer, a smart phone, or other mobile devices.

Many online auctions provide a “buy it now” option to accommodate those buyers who cannot wait until the auction ends. A buyer can purchase an item immediately by paying the buy-it-now price. However, the buy-it-now price is usually a regular retail price which can be much higher than the final auction price. For the purpose of this model, buy-it-now should not be considered as part of the auction design.

4.2 Simulation results

This section evaluates the I*b*SCHF through computational analysis. The assessing metrics are those commonly found in the literature.

4.2.1 Metrics

Efficiency and Information Revelation are used as the performance measures in the evaluation. Parkes (2001) developed these metrics for testing the performance of iBundle, an iterative combinatorial auction for general combinatorial allocation problems. These metrics are redefined in the context of service scheduling as follows:

Efficiency of Scheduling: $eff(S)$, is measured as the ratio of sum of the values in final schedule S to the sum of the values in optimal schedule S^* that maximizes total value across the agents:

$$eff(S) = \frac{\sum_{j \in N} V_j(S)}{\sum_{j \in N} V_j(S^*)}$$

where $V_j(S)$ is agent j’s valuation on a schedule $S, V_j(S^*)$ is agent j’s valuation on the optimal schedule S^*, and N is the set of all agents.

Information Revelation: $inf\ Rev(j)$, is measured for agent j as the sum of the final price bid for all bundles that agent j has placed bids on, as a fraction of the sum of the values for all feasible bundles.

$$inf\ Rev(j) = \frac{\sum_{B \in Bid_j} P_j^*(B)}{\sum_{B \in E_j} V_j(B)}$$

where $P_j^*(B)$ is the maximum bidding price of agent j for bundle B during the auction; Bid_j is the set of bundles that agent j has placed bids on; and E_j is the set of feasible bundles for agent j. The average information revelation over all agents is considered as overall information revelation (inf).

Bidding process often terminates before agents have revealed the complete information about their values for service time slot bundles. The information revelation metric measures the extent to which an agent has revealed its value for each service time slot bundle to the provider during the auction.

The DSSP model is coded in ILOG Optimization Programming Languages (http://www-01.ibm.com/software/websphere/products/optimization/) and the ten groups of problem instances are solved using ILOG CPLEX. The flow control of the iterative bidding process is coded in the OPL (Optimization Programming Languages) script language. A desktop PC with 2.4G Intel CPU and 8 GB memory was used to run the experiments.

4.2.2 Problem Sets

Ten problem groups are generated, with the customer number ranging from100 to 1,000. For each group, ten instances are randomly generated. Service time slots' capacity are allocated in proportion to the number of customers such that, for most of the instances, around 80–90 % of the customers will be awarded a feasible bundle. The configuration of the test problem sets are summarized in Table 4-1.

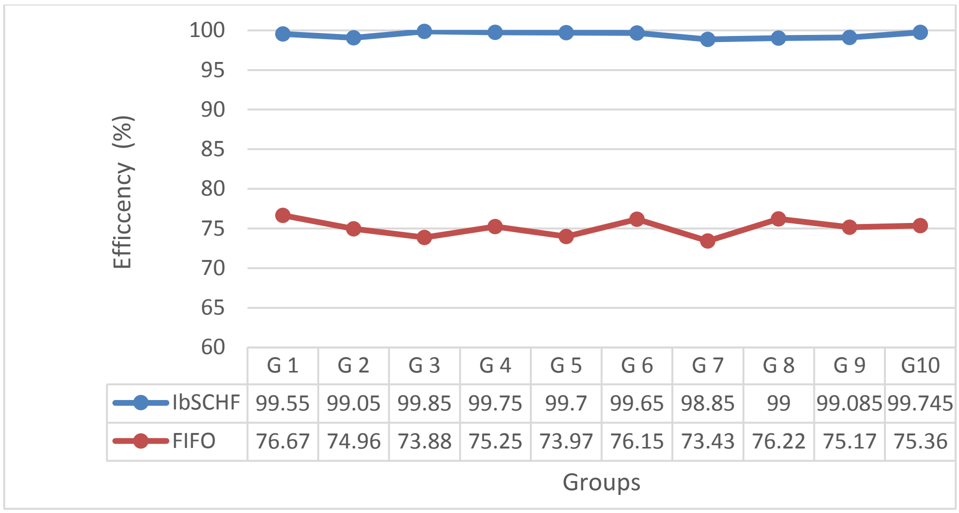
In the design of the testing data, it is assumed that there is a regular retail price for each of the available service time slots, and the retail price for a bundle is the sum of the retail prices of the service time slots included in the bundle. The reservation price for a bundle is set to be 40% of its retail price, since it is common practice in online service auctions that the termination price can be as low as a 60% discount from the regular retail price. It is assumed that customers who enter the auction expect some discount. They are not interested in purchasing the bundle at a price higher than the regular retail price. Customers' values on a bundle are randomly drawn from a uniform probability distribution between reservation price and its regular retail price.

Table 4-1 Configuration of testing problems

Problem		Number of agents	# of service time slots	# of feasible bundles per agent	Number of Instance
#	Name				
1	Group 1	100	20	Random(5,10)	10
2	Group 2	200	30	Random(5,10)	10
3	Group 3	300	40	Random(5,10)	10
4	Group 4	400	50	Random(5,15)	10
5	Group 5	500	60	Random(5,15)	10
6	Group 6	600	70	Random(5,15)	10
7	Group 7	700	80	Random(5,15)	10
8	Group 8	800	90	Random(5,20)	10
9	Group 9	900	100	Random(5,20)	10
10	Group 10	1000	110	Random(5,20)	10

4.2.3 Computational Results

The I*b*SCHF is compared against the commonly used first-come-first-served capacity allocation policy. This approach is easy to implement and performs reasonably well in terms of enhancing revenue when capacity supply and demand are balanced. However, when demand exhibits strong seasonality, an auction-based policy performs better. To compare the performance of an auction-based policy against that of a first-come first-served capacity allocation policy, each policy is applied to the ten groups of testing problems. In the first-come first-served policy scenario, customers in an instance are first randomly ordered. Capacity is allocated according to their position in the sequence until no more customers can be satisfied. Figure 4-1 shows the efficiency of the first-come-first-served policy and of the proposed approach over the ten test problems. It is observed that the first-come-first-served policy achieves on average 75 % of the efficiency obtained by the proposed approach.

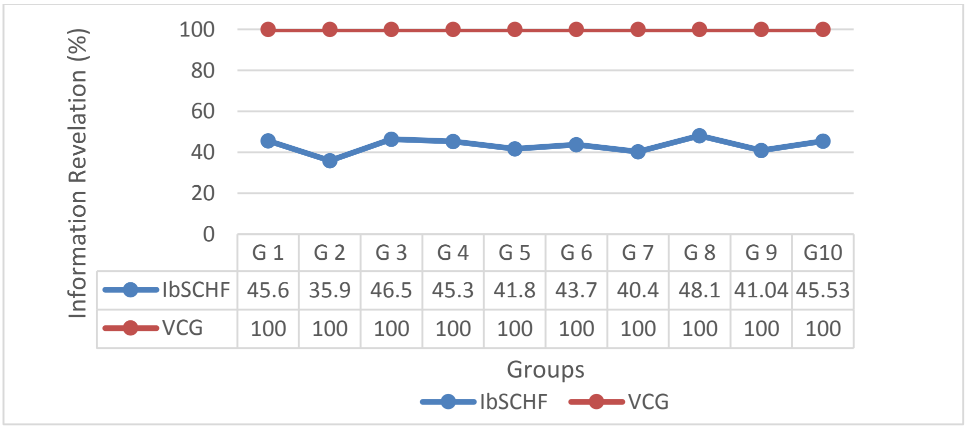


	G 1	G 2	G 3	G 4	G 5	G 6	G 7	G 8	G 9	G10
IbSCHF	99.55	99.05	99.85	99.75	99.7	99.65	98.85	99	99.085	99.745
FIFO	76.67	74.96	73.88	75.25	73.97	76.15	73.43	76.22	75.17	75.36

Figure 4-1 Efficiency of the FIFO and of the IbSCHF over ten groups

The I*b*SCHF is also compared against the VCG auction. In the VCG auction, all agents report their complete valuations over all service time slot bundles at the beginning of the auction. Figure 4-2 plots the information revelation performance of the I*b*SCHF. Compared to the VCG, which requires 100% information revelation, I*b*SCHF requires a less than 50% information revelation with bid increment ε =5, which comes with the cost of losing only 1%-2% of the efficiency, as shown in Figure 4-3.

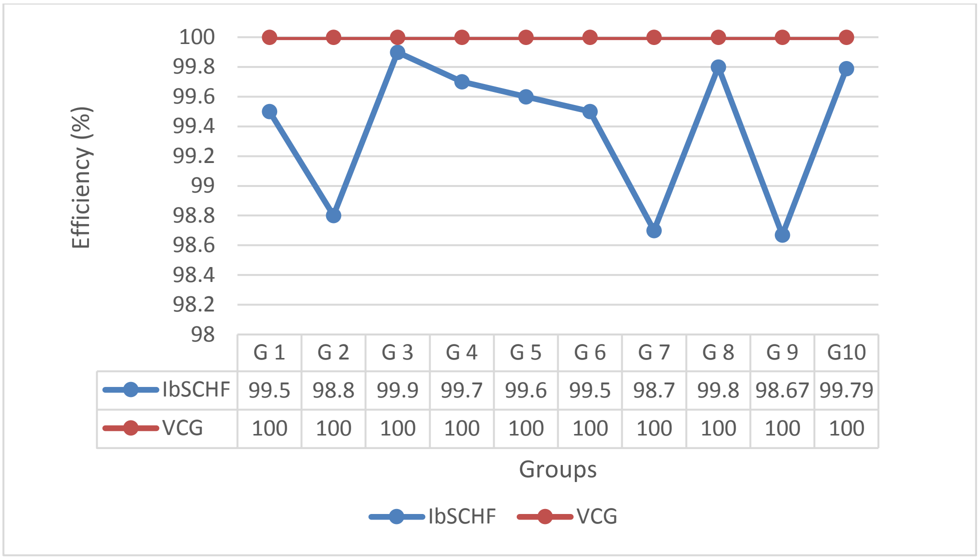
The comparison results presented in Figure 4-2 and Figure 4-3 reflect the difference between the iterative bidding structure (in I*b*SCHF) and the one-shot bidding structure (in the VCG auction) in the context of distributed service scheduling.



	G 1	G 2	G 3	G 4	G 5	G 6	G 7	G 8	G 9	G10
IbSCHF	45.6	35.9	46.5	45.3	41.8	43.7	40.4	48.1	41.04	45.53
VCG	100	100	100	100	100	100	100	100	100	100

Figure 4-2 Information revelation of the VCG and of the I*b*SCHF as the problem difficulty is

increased

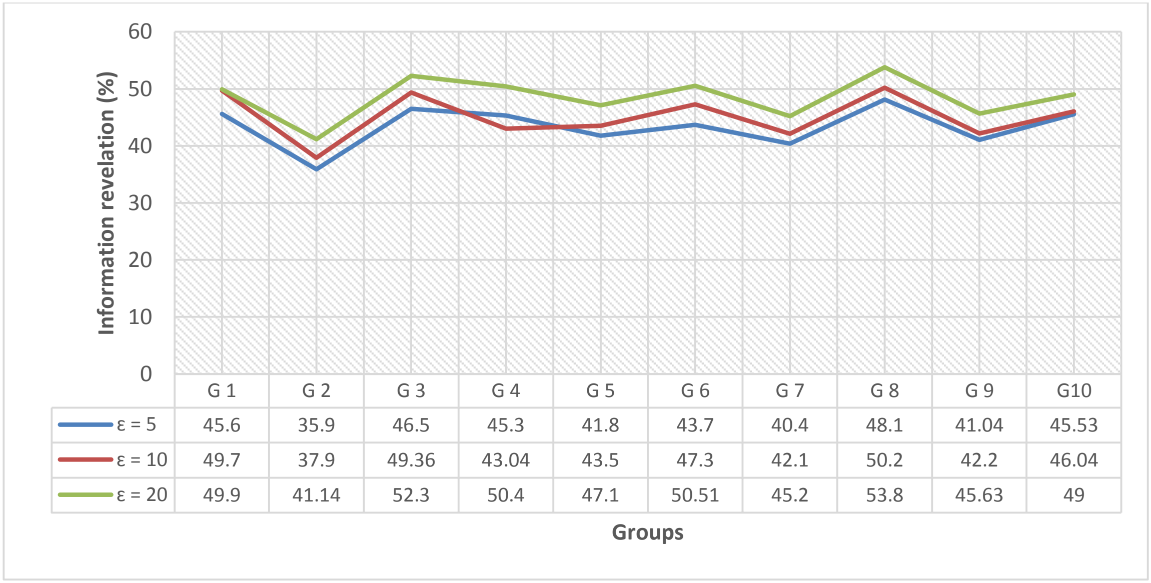


	G 1	G 2	G 3	G 4	G 5	G 6	G 7	G 8	G 9	G10
IbSCHF	99.5	98.8	99.9	99.7	99.6	99.5	98.7	99.8	98.67	99.79
VCG	100	100	100	100	100	100	100	100	100	100

Figure 4-3 Efficiency performance of the VCG and of the I*b*SCHF as the problem difficulty is increased

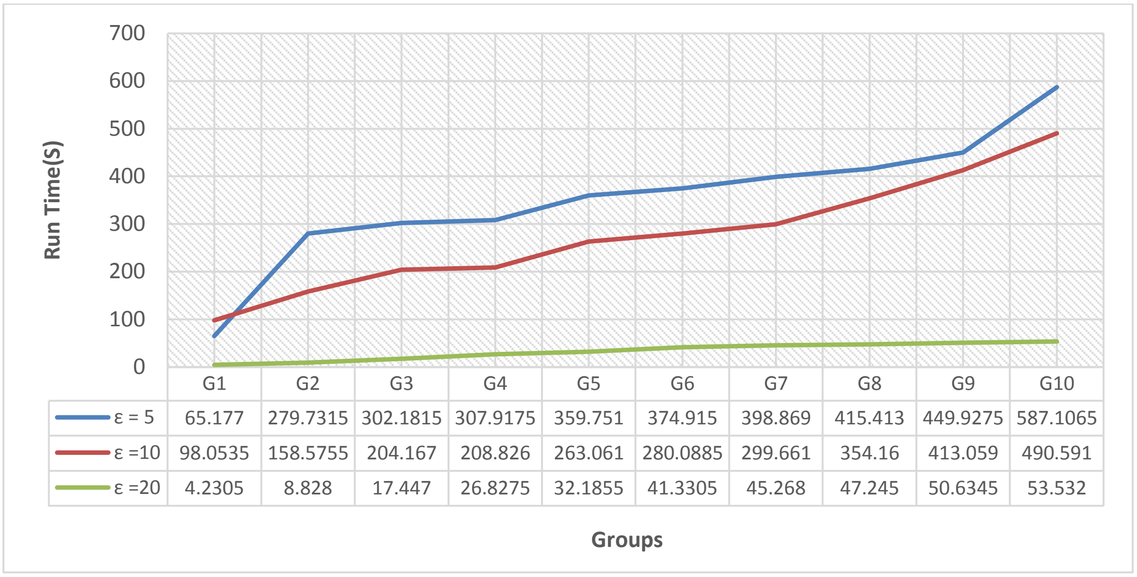
4.2.4 The effect of bid increments

Figure 4-4 plots the information revelation performance of the I*b*SCHF over different bid increments. Bigger bid increment leads to more information revelation. The reason is that bigger bid increment values may overcome some low price equilibrium point that smaller increments could find.



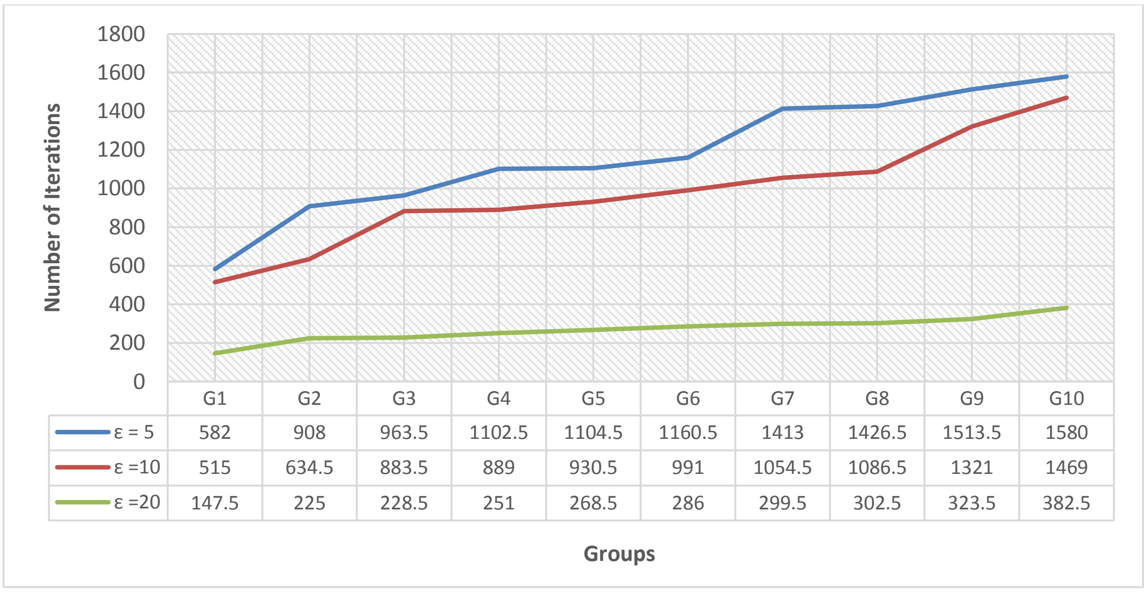
	G 1	G 2	G 3	G 4	G 5	G 6	G 7	G 8	G 9	G10
ε = 5	45.6	35.9	46.5	45.3	41.8	43.7	40.4	48.1	41.04	45.53
ε = 10	49.7	37.9	49.36	43.04	43.5	47.3	42.1	50.2	42.2	46.04
ε = 20	49.9	41.14	52.3	50.4	47.1	50.51	45.2	53.8	45.63	49

Figure 4-4 Information revelation performance of the I*b*SCHF over 10 groups with different bid increments



	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
ε = 5	65.177	279.7315	302.1815	307.9175	359.751	374.915	398.869	415.413	449.9275	587.1065
ε =10	98.0535	158.5755	204.167	208.826	263.061	280.0885	299.661	354.16	413.059	490.591
ε =20	4.2305	8.828	17.447	26.8275	32.1855	41.3305	45.268	47.245	50.6345	53.532

Figure 4-5 Run time of the I*b*SCHF over10 groups with different bid increments



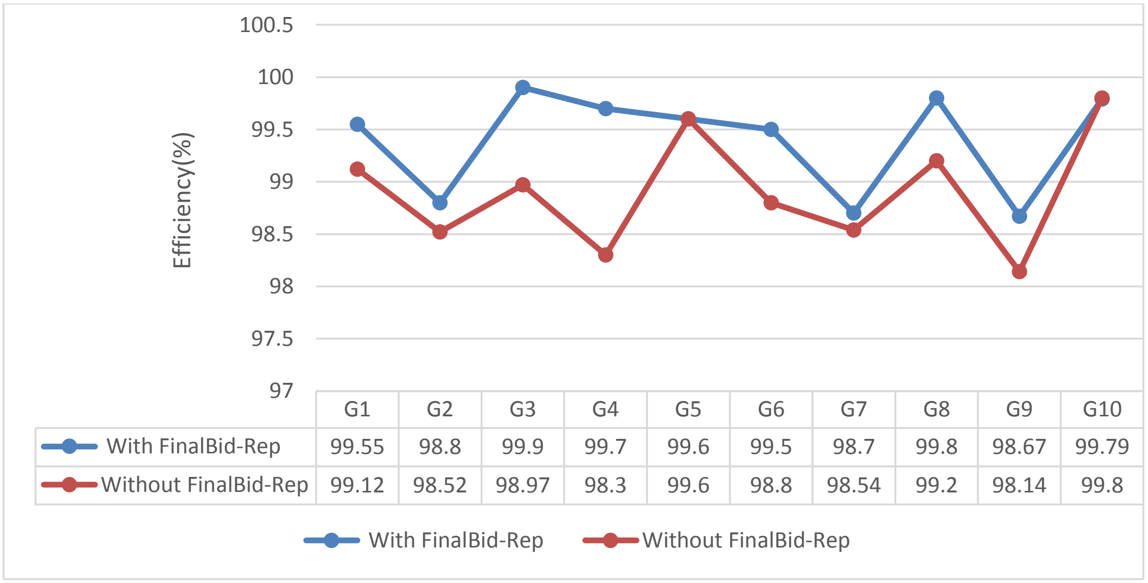
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
ε = 5	582	908	963.5	1102.5	1104.5	1160.5	1413	1426.5	1513.5	1580
ε =10	515	634.5	883.5	889	930.5	991	1054.5	1086.5	1321	1469
ε =20	147.5	225	228.5	251	268.5	286	299.5	302.5	323.5	382.5

Figure 4-6 Number of Iterations of the I*b*SCHF over10 groups with different bid increments

Figure 4-5 plots the run time of the I*b*SCHF for different bid increments. The results show that bigger increment values requires less time for the auction to terminate. This makes sense, because bigger increments lead to a lower number of iterations (Figure 4-6), and many agents quickly drop out as the prices get too high. Figure 4-5 also illustrates that when the number of agents increases the level of completion increases, and it takes more time to compute the solution.

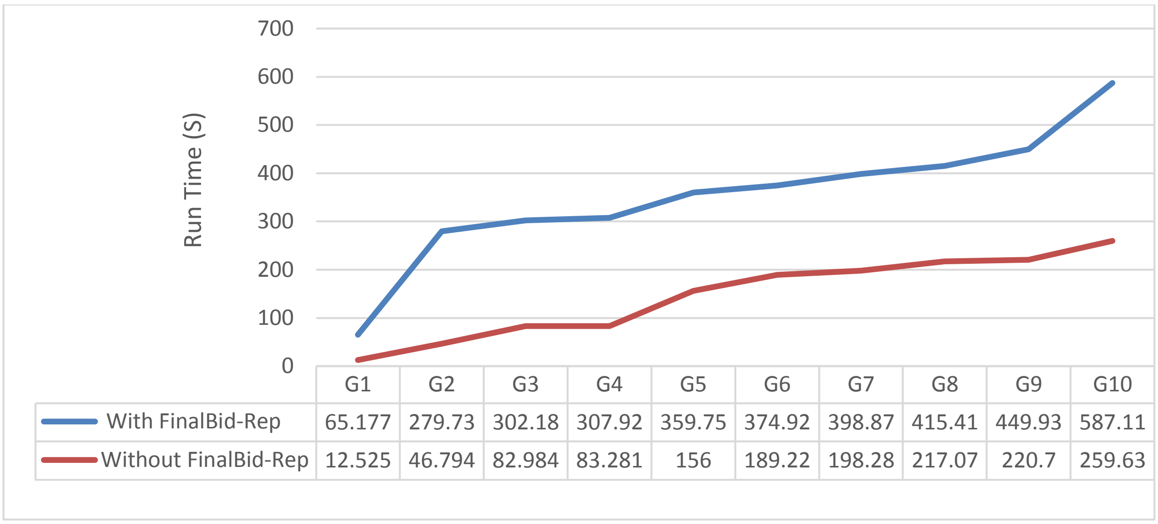
4.2.5 The effect of final bid repeating

The reason for considering final bid repeating rule in I*b*SCHF is to boost efficiency and service provider's revenue. As shown in Figure 4-7 efficiency will be increased by considering final bid repeating rule. However considering final bid repeating rule will increases the level of completion, and consequently increases the run time (Figure 4-8).



	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
● With FinalBid-Rep	99.55	98.8	99.9	99.7	99.6	99.5	98.7	99.8	98.67	99.79
● Without FinalBid-Rep	99.12	98.52	98.97	98.3	99.6	98.8	98.54	99.2	98.14	99.8

Figure 4-7 The effect of final bid repeating on efficiency



	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
With FinalBid-Rep	65.177	279.73	302.18	307.92	359.75	374.92	398.87	415.41	449.93	587.11
Without FinalBid-Rep	12.525	46.794	82.984	83.281	156	189.22	198.28	217.07	220.7	259.63

Figure 4-8 The effect of final bid repeating on run time

The next section explains how I*b*SCHF can be applied to accommodate dynamic changes in the service scheduling environment.

4.3 Accommodating Dynamic Changes

Generated schedules in a dynamic service environment cannot be used for a long time because of unexpected events. Therefore revising the schedules at some point in time is necessary to accommodate dynamic changes. Two main question will arise for revising the schedules: when to revise and how to respond?

4.3.1 When to revise?

There are several ways to decide on timing for revising the scheduling decisions. The first approach is called periodic rescheduling policy in which generated schedules are revised periodically. In this approach revisions are made at the beginning of each time interval by taking into account new information gathered from the scheduling environment. Determining the period length depends on the application domain. Muhlemann et al. (1982), Ovacik and Uzsoy (1994), and Sabuncuoglu and Karabuk (1999) investigated the effects of different rescheduling frequency in manufacturing environments. The second approach is called event driven policy in which revisions are made in response to an unexpected event that change the system states. Church and Uzsoy (1992) provide a comparison of periodic and event driven policies for dynamic shops. A comparison between the performance of periodic policy and the performance of event driven policy in a single machine environment is also presented by Vieira et al. (2000). Hybrid policy is another method in which rescheduling is triggered when an unexpected event occurs and at the end of each time interval (Yamamoto and Nof 1985).

4.3.2 How to respond?

In general there are two main strategies: 1) completely regenerate a new up-to-date schedule for all remaining jobs and 2) repair the existing schedule to take into account of the current state of the system. The first strategy may in principle be capable of maintaining optimal solutions, however computation times are likely to be prohibitive and production may be significantly delayed while the schedule is regenerated. Furthermore, completely regenerate a schedule is not applicable in a service environment because generating new schedules for customers is not possible without their permission. In the second strategy several techniques such as heuristics, knowledge-based systems, fuzzy logic, neural networks, and hybrid techniques can be used to repair a schedule. Ouelhadj et al. 2009 provide a review of the state of the art of research on dynamic scheduling techniques and compare their relative merits.

In a service environment, the repair strategy is the most appropriate approach; when dynamic changes happen, generating a completely new schedule may find customers unsatisfied with their new schedules. A repair strategy that tends to minimize the perturbation to the original schedule would be more appropriate to apply in service scheduling environments.

A periodic repair approach by using I*b*SCHF for dealing with dynamic changes in service environments with the objectives of automation and optimization is described in the next section.

4.3.3 Periodic repair approach in service scheduling environment

The automated repair scheduling approach for accommodating dynamic changes is proposed here, along with how to effectively allocate the newly-available service time slots created by customer cancellations. To fill newly available service time slots some service providers keep an on-call list of customers who could be available and interested to fill those newly time slots. However, calling the customers on the on-call list incur high administrative costs to the service provider and is a multiple-round rescheduling process because, once a customer is allocating to a newly-available time slot, its original time slot (if he/she had already been assigned one) becomes available and will have to be reallocated.

This process of reallocating time slots can go multiple rounds until all the available time slots have been filled or there is not any customer willing to be rescheduled. Manually conducting multiple round of rescheduling process will significantly increase the administrative workload. In addition, constructing a high quality schedule is an optimization problem that requires computing-based decision making tools, so manually-created schedules is not an appropriate approach for generating high quality schedule.

Thus, to improve the current practice of dynamic service scheduling, two challenges need to be addressed. The first one is automate the process of rescheduling to reduce the administrative workload and enhance the efficiency of the process; the second one is optimization, to systematically optimize the quality of the rescheduling solutions. To address both these challenges, the approach proposed here is periodic repair scheduling in response to dynamic changes.

The periodic repair scheduling algorithm can be described as follows:

```
$Step\ 0 : Set\ i = 1;$
$Step\ 1 :$
    $ATS = all\ Available\ Timeslots\ between\ (i-1)T\ and\ iT;$
    $SCA = all\ Standby\ Customer\ Agents\ for\ ATS$
    $While\ (\ ATS \neq \emptyset\ \&\ SCA \neq \emptyset\ )$
      $\{\ \ Result \leftarrow \mathrm{I}b\mathrm{SCHF}\ (ATS, SCA);$
          $for\ any\ customer\ agent\ j \in Result$
          $\{\qquad SCA \leftarrow SCA - j\ ;$
                  $ATS \leftarrow ATS - \{timeslots\ awarded\ to\ j\}\ ;$
```

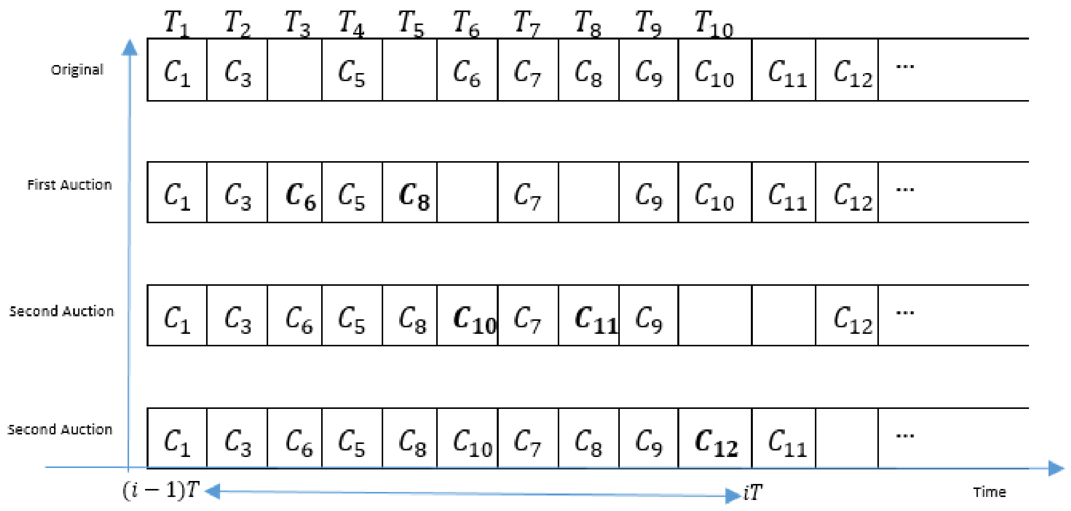
```
                        $if\ agent\ j\ was\ holding\ a\ timeslot\ between$
                        $(i-1)T\ \&\ iT\ \ before\ this\ new\ allocation$
                        {
                          $ATS \leftarrow ATS \cup \{\ timeslots\ previously\ held\ by\ j\}$
                        }
    }        }
$Step\ 2$:  $Set\ i = i + 1; and\ go\ to\ step\ 1;$
```

In step 0, the service provider agent identifies the set of newly-available time slots along the scheduling time horizon between $(i-1)T\ \ and\ iT$. It then determines which customer agent has registered as a standby status for available time slots between $(i-1)T\ \&\ iT$. Next, the service provider agent starts the IbSCHF by sending out a message containing the set of available service time slots to the customer agents with the standby status. Those standby customer agents who are interested in that set of available time slots participate in the auction. When IbSCHF terminates and new time slot allocations are determined, the service provider agent will update the existing schedule and notify the participating customer agents with the results. The customer agents that gained their requested time slots will update their status levels from standby to reserved.

Since each customer agent can only have one time bundle, if a customer agent changes its status from standby to reserved and is assigned to new bundle, the time slots it previously held become available. If this newly-available time slot is between $(i-1)T\ \&\ iT$ it will be added to the ATS . Therefore, additional round of IbSCHF need to be conducted until either no customer agents are interested in the available time slots or no time slots are available between$(i-1)T\ \&\ iT$. This process is called subsequent repair scheduling.

4.3.4 A Worked example

An example of a service scheduling solution using the periodic repair scheduling approach is shown in Figure 4-9, which represents the allocation of ten time slots (T_1 *to* T_{10}). In this example, T are taken for the following two work days. For the sake of simplicity, in this illustrative example it is assumed that, only one time slot is allocated to each customer. However, in the IbSCHF, the winner determination model does not have this restriction.



T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}
Original
C_1 C_3 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} ...
First Auction
C_1 C_3 C_6 C_5 C_8 C_7 C_9 C_{10} C_{11} C_{12} ...
Second Auction
C_1 C_3 C_6 C_5 C_8 C_{10} C_7 C_{11} C_9 C_{12} ...
Second Auction
C_1 C_3 C_6 C_5 C_8 C_{10} C_7 C_8 C_9 C_{12} C_{11} ...
$(i-1)T$
iT
Time

Figure 4-9 Example of a problem solved using the periodic repair scheduling approach

The service repair scheduling problem parameters which include the list of customers willing to be rescheduled, their preferred service time(s) and their value, are shown in Table 4-2.

At the beginning of time window T, $ATS = T_3$ *and* T_5. T_3 *and* T_5 become available as two customers cancel their allocated service time slots. The repair scheduling process starts at the beginning of time window T with the set of ATS.

Table 4-2 Problem parameters of example

Customer Agent Code	Service provider time slot	Value($)
C_6	T_3 T_6 T_9	11 10 5
C_8	T_8 T_5	5 6
C_{10}	T_{10} T_6	10 8
C_{11}	T_{11} T_5 T_8	6 4 4
C_{12}	T_{10} T_{12} T_3 T_8	8 8 8 3

The service provider agent starts the IbSCHF at the beginning of time period T. From Table 4-2 we can see that customers C_6 *and* C_{12} are interested in available time slot T_3, and customer agents C_8 *and* C_{11} are interested in available time slot T_5. Their customer agents will participate in the auction. After the IbSCHF termination, time slot T_3 has been awarded to C_6 and time slot T_5 has been awarded to C_8 , because they have higher values for time slots T_3 *and* T_5 . Consequently, now that C_6 and C_8 have been rescheduled, the time slots T_6 *and* T_8 which originally belonged to those customers are now available. Another auction by using IbSCHF will be started to allocate these newly-available time slots to interested customers. In this example, customer agent C_{10} participates in the second auction for time slot T_6 and agents C_{11} and C_{12} for T_8. The result of this second auction is that time slot T_6 has been awarded to C_{10} and time slot T_8 has been awarded to C_{11} . In this case, since C_{11} was outside of time window T, the only available time slot for the next auction will be T_{10}, which belonged to customer C_{10} in the previous round. The third auction begins, this one for time slot T_{10} and with the participation of customer agent C_{12}. The third auction instantly awards T_{10} to customer C_{12}, the only customer that participated. At this point, all the openings in the time window T have been allocated and no further auctions are required.

4.3.5 Efficiency Analysis

The proposed periodic repair scheduling system provides the potential for successful automated service scheduling. It is reasonable to assume that a substantial number of customers would be willing to participate in this program. In order for such a system to be functional, the customer agent will register itself with standby status for certain unavailable service time slots in the service provider internet portal and then automatically participate in the rescheduling process when any of those time slots become available.

To evaluate the responsiveness of the system, we have randomly generated nine groups of problem instances of different sizes and structures. The configuration of the test problem sets and the corresponding solving times by means of CPLEX are summarized in Table 4-3. The flow control of the repair scheduling is coded in OPL (Optimization Programming Languages) script language. A desktop PC with a 2.4G Intel CPU and 8 GB of memory was used to run the experiments.

Table 4-3 Configuration of testing problems and computational results

Group	# of Customer agents	Window length (# of time slots)	# of Available time slots (ATS)	Running time (seconds)	# of Auctions
1	100	8	2	7.4062	1
2	200	16	3	24.008	2
3	300	24	4	39.648	2
4	400	32	5	54.254	2
5	500	40	6	92.898	3
6	600	48	7	112.878	3
7	700	56	8	126.806	3
8	800	64	9	141.596	3
9	900	72	10	194.584	4
10	1000	80	11	237.154	4

In these problem instances, the maximum number of time slots required for all service requests has been limited to two. For each group, CPLEX can find optimal solutions to instances with 1000 customer agents within less than 4 minutes, which is satisfactory for the responsiveness requirement of our repair scheduling system.

4.4 Summary

This chapter presents the IbSCHF for the distributed service scheduling problem. The approach is incentive-compatible in the sense that customers will follow the myopic best-response bidding strategy prescribed by the auction protocol. The simulation results indicate that the proposed framework requires lower information revelation with the cost of losing only 1%-2% of efficiency, compared to the one-shot VCG auction. By applying the IbSCHF in a multi-agent environment the three main service scheduling challenges of service scheduling problem namely, distributed environment, the presence of customers' private information and complicated objectives can be overcome. This framework can also be applied to effectively allocate the newly available service time slots created by dynamic events.

Chapter 5 Scheduling Non-Commercial Services

5.1 Introduction

An iterative bidding framework for DSSP was proposed and detailed in chapter 4. In this proposed framework, a price mechanism is used to allocate service time slots to customers. However, in non-commercial service scheduling environments, such as scientific facility service scheduling, government service scheduling and healthcare service scheduling, for social economic and political reasons service providers cannot use a price mechanism to schedule customers along the service timelines. Therefore, novel mechanism design models need to be developed for scheduling non-commercial services without using a price system or payment transfers.

In this chapter, I study the scheduling aspect of non-commercial services. I am especially interested in learning how to design effective mechanisms for non-commercial service scheduling, and how customers' private information and efficiency interplay under such mechanisms. I have designed an auction-based (with iterative bidding) scheduling framework under two constraints (1) service providers are restrained from using any price mechanisms to allocate service time slots to customers and (2) customers are reluctant to share their complete availability information. The next section introduces the non-commercial services scheduling problem and customers' private information implication.

5.2 Non-commercial services scheduling and customers' private information

The Non-Commercial Service Scheduling (NCSS) problem concerns the allocation of limited resources to the service activities at specific times. This allocation must obey a set of rules or constraints that reflect the temporal relationships between activities and the capacity limitations of a set of shared service resources.

5.2.1 The implication of customers' private information on efficiency

NCSS can be modelled as an optimization problem in which customers' private availability information constrains the solution space.

According to the definition of service presented in chapter 1, a service relies significantly on customer inputs (Sampson & Froehle, 2006). In other words, in order for a service to be produced, a customer has to present personally or he/she has to present his/her belongings or information. By considering customers' available time for their required service, the number of no-shows can be decreased and customer satisfaction increased. For example, in health care services, it has been shown that matching patients with their preferred provider and offering them a convenient appointment time can decrease the number of no-shows and thereby increase operational efficiency (Barron 1980). If all customers report their full availability, the service provider can obtain an optimal schedule by solving the optimization problem. However, if customers only reveal partial availability to the service provider, the quality of the solution will be compromised. High-quality schedules could be deemed infeasible due to the partial availability of customer information. The scheduling problem facing service providers is a distributed scheduling problem in the sense that the customers' true availability is their private information and may not be fully known to the service provider. Customers are reluctant to reveal their complete availability because a complete revelation increases the possibility of being assigned an undesirable time slot. Generating high quality schedules and, at the same time, accommodating customers' preferences is a challenge. In addition to dealing with strategic behaviours from customers, the administrative workload of collecting customers' availability information and negotiating with them for possible changes can be very difficult due to a large number of customers and a manually managed process. The proposed approach makes it possible to automate the NCSS procedure and improve the quality of schedules. In the next section, I formulate the service provider's and customers' decision problems in NCSS.

5.3 Formulation of service provider's and customers' decision problems

Service scheduling is a multilateral decision making problem with the service provider and customers as independent decision makers. The service provider needs to decide how to schedule service requests to achieve its objectives and, at the same time, respect the customer's availability constraints. The decision facing a customer is how much availability information she needs to reveal in order to maximize her benefit.

5.3.1 Service provider's decision problem

Consider an NCSS problem consisting of a service provider and a group of customers. The provider receives a set of service requests from customers. Each request is assigned a weight which reflects its contribution to the provider's objective. The provider has a limited service capacity and knows the time required for processing each request. The provider's objective is to maximize the sum of the weights of a schedule. An important type of constraints of NCSS is customers' availability. Since customers need to be present for the service, the provider cannot schedule a customer for a time slot when she is not available. I describe a customer's availability by a set of available time intervals along the scheduling timeline. As I will later develop an iterative bidding framework for NCSS problems, I represent an available time interval as a bid from a customer using the bundle bidding language (Nisan ,2006) developed for combinatorial auctions. To apply the language, I need to first discretize the provider's service timeline into fixed-size time slots. In this way, without loss of generality, an available time interval can be defined by a bundle of adjacent time slots contained in the interval. Unlike general combinatorial auctions, customers do not attach prices to their bids in NCSS. In our case, bids are used by the customers to indicate their availabilities. If a customer submits a bid (their available time interval), she informs the provider that she is available to be scheduled during that interval.

The set of intervals that contains a customer's complete availability is referred to as the customer's set of Feasible Time Intervals (FTIs).

Let E_j be the set of availability intervals revealed by customer. It is clear that E_j is a subset of customer$j$'s FTIs. The service provider will not schedule customer j 's request outside her E_j. Let w_j be the weight scale of customer j assigned by the service provider and p_j the processing time of customer j's request. Let Ω be the set of time slot available for allocation and J the set of customers who have service requests to be scheduled; let $x_j(B) = 1$ if the time slot bundle $B \subseteq \Omega$ is allocated to customer j and zero otherwise. The provider's decision problem is to determine the allocation of limited service time to the requests in a way that the sum of the weights of the awarded requests is maximized. The problem can be formulated as the following integer programming.

$$max \sum_{B\subseteq\Omega} \sum\nolimits_{j\subseteq J} x_j(B)\, w_j$$

Subject to

$\sum_{B\subseteq\Omega} x_j(B) \leq 1, \quad \forall j \in J$ (1)

$\sum_{B\ni i} \sum_{j=1}^{n} x_j(B) \leq 1, \quad \forall i \in \Omega$ (2)

$\sum_{B\subseteq\Omega} x_j(B) = \sum_{B\subseteq E_j} x_j(B), \quad \forall j \in J, \forall B \in \Omega$ (3)

$|B| + Hx_j\,(B) \leq p_j + H, \qquad \forall j \in J, \forall B \in \Omega$ (4)

$|B| + H \geq p_j + Hx_j\,(B), \qquad \forall j \in J, \forall B \in \Omega$ (5)

$x_j(B) = \{0,1\}, \quad \forall B \subseteq \Omega, \quad \forall j \in J$ (6)

The set of constraints (1) ensures that any customer can only obtain one bundle of time slots. The set of constraints (2) ensures that a time slot is not included in two bundles that have been assigned to customers. The set of constraints (3) ensures that if a bundle is assigned to a customer, it must belong to the set of available intervals submitted by that customer. These constraints prevent a service provider from assigning customers' time bundles which they are not willing to accept. Constraints (4) and (5) ensure that if a bundle is assigned to a customer, the length of the bundle is equal to the processing time of the customer's request, where H is a large positive constant that is used for the linearization of the logical constraint "if." The minimum value of H depends on the problem instance. In general, a value of H that is greater than the number of available time slots of the service provider is large enough to enforce the logical "if" constraint.

Constraints (6) are integer constraints. The provider's decision problem is NP-hard, as stated in the following theorem.

Theorem 1 *The service provider's decision problem in NCSS is NP-hard.*

Proof: Consider a special case of the provider's decision problem, in which a set of service requests from customers needs to be scheduled. A request may be scheduled on one of the l intervals on a discrete time scale on a single resource. The decision version of this special case of provider's decision problem is identical to the job interval selection problem, which is NP-complete (Keil, 1992). Therefore, the decision version of provider's decision problem is NP-complete. It follows that the provider's decision problem is NP-hard.

5.3.2 Customers' decision problem

To model the customers' decision problem, I first introduce their preference structure over the time intervals in their FTIs. A customer's FTI list is her private information, and is not known to the service provider. She may behave strategically, for example, may hide a portion of her FTIs, to maximize her benefits. To reflect this self-interested property of customers, I call them agents. I assume that an agent prefers some time intervals over others within its FTIs and that the preferences can be quantified by associating a preference violation cost to each time interval. The preference violation cost reflects the level of the preference violations to an agent. It is essentially a subjective measure adopted by an agent. For example, it can be a function of the number and severity of preference violations that a time interval may cause to the agent. In many cases, it is reasonable to assume that an agent can order the time intervals in its FTIs according to the increasing order of their preference violation costs. That is, given an ordered FTI, $c_1 < c_2 < c_3 < \cdots c_k \ldots < c_{|FTI|} < c_0$ is known to the agent, where c_k denotes the preference violation cost of the kth time interval in an FTI and c_0 denotes the preference violation cost of not being allocated any time intervals. Note that an agent may have identical preference violation costs for more than one time interval. In an FTI, the highest preference violation cost is that of not being awarded anything in the service schedule.

An agent would prefer to be assigned a time interval with their lowest preference violation cost. However, the final schedule is computed based on the time intervals submitted from all agents. Because of the potential time conflicts among agents' requests, it is difficult for them to decide how much availability information should be revealed in order to obtain a preferred assignment. If an agent only submits a few low -cost time intervals, it can control the upper bound of its preference violation cost as the awarded bundle must be within the set of submitted intervals. However, by doing so, it runs the risk of not being allocated anything if the submitted time intervals are also demanded by other agents with higher weights. On the other hand, if an agent submits its complete FTI list, it maximizes its probability of getting an assignment. However, reporting complete FTIs increases the possibility of ending up with an interval with high preference violation cost. In fact, there is not a clear strategy for agents to minimize their expected preference violation costs. The effectiveness of an agent's bidding strategy depends on how heavy the competition is for its desired time intervals and on other agents' bidding strategies. This uncertainty leads to speculation during bidding, which will increase agents' computation cost and may render a final schedule that is arbitrary and far from optimal. The goal, therefore, is to design a mechanism which systematically evolves the solution towards an optimal one given the constraint that agents try to avoid high cost assignments by not revealing their complete availability. Since no payment is allowed in the NCSS setting, the possibility of applying standard one-shot VCG mechanisms (Clarke, 1971 ; Groves, 1973; Vickrey,1961) and even its iterative implementations (Parkes,2006) is eliminated. In the following section, I propose a non-price bidding approach to the NCSS problem and evaluate its performance.

5.4 The iterative bidding framework

The iterative bidding framework proposed here is an auction-based approach to the NCSS problem. The framework contains two major components: an iterative bidding procedure and an integer programming model for winner determination. The winner determination model computes provisional schedules which maximize the sum of the weights of the winning bids at each round. The iterative bidding procedure provides a structure through which the agents and the service provider (auctioneer) can interact in a systematic way and eventually evolve provisional solutions towards an optimal one. Iterative bidding also reduces the level of agents' information revelation and adds the potential of accommodating dynamic changes during the bidding process. The iterative bidding framework is a single-attribute auction that allows negotiation over a non-price attribute: the level of availability of agents revealed to the auctioneer. The framework requires agents to reveal their availability only on a necessary basis.

5.4.1 Iterative bidding

The iterative bidding procedure is depicted as a flow chart in Figure 5-1. There are mainly four components of the bidding procedure: initialization, availability update and bidding, termination checking and winner determination.

5.4.1.1 Initialization

Initially, an agent has a service request and provides a set of time slots during which the request can be processed. The agent constructs its initial bid by selecting the time slot with the lowest preference violation cost and sends it to the auctioneer.

5.4.1.2 Availability Update and Bidding

Agents update their availability by sending new feasible time slots to the auctioneer. At the beginning of round t $(t > 1)$, an agent needs to decide whether it submits additional time slots to the auctioneer at round t. This decision is made based on the provisional schedule which resulted from the winner determination at round $t-1$. If an agent was not awarded in the provisional schedule at round $t-1$, it has two availability update options at round t: (1) it can submit additional time slots, or, (2) it can keep the set of submitted time slots unchanged by submitting an *empty bid* (a bid without time slots). However, if an agent submits an empty bid, the auctioneer will consider that the agent has entered into final bid status and so will be forbidden from updating its availability in future rounds.

Given these options, I will show in the next section that, since agents are assumed to be rational in minimizing their preference violation costs, they will always follow the *myopic* *bidding strategy*. Figure 5-2 depicts the agent's myopic bidding strategy in the format of a flow chart. After receiving the winner determination results from round $t-1$, the agent will submit an empty bid at round t if it won at round $t-1$. However, if the agent lost, it will check whether all its FTIs have been submitted. If yes, the agent will still submit an empty bid because there are no more available time slots to be added; if no, the agent will select the one with the lowest cost from its remaining FTIs and submit it to the auctioneer.

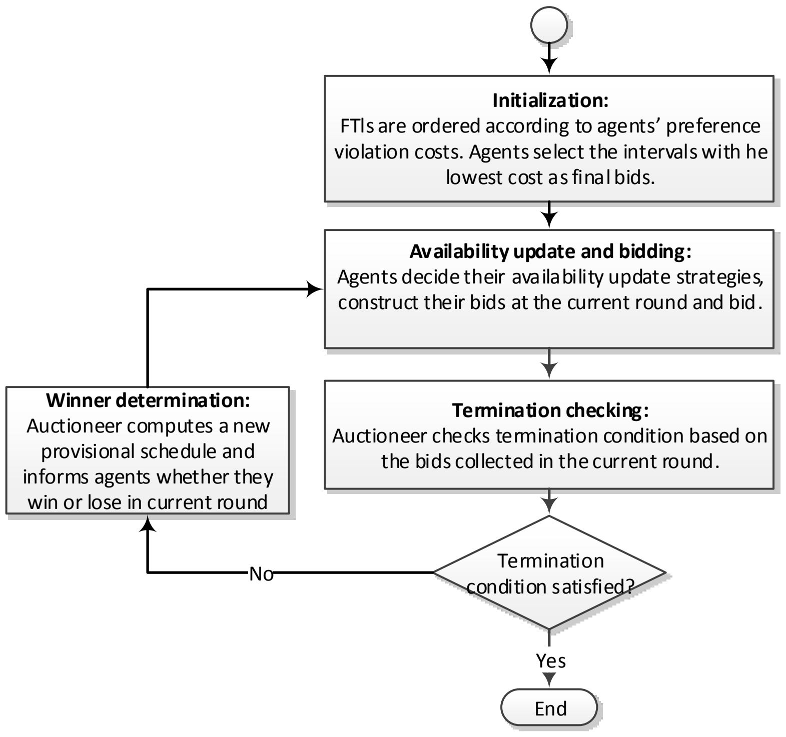

Initialization:
FTIs are ordered according to agents' preference violation costs. Agents select the intervals with he lowest cost as final bids.
Availability update and bidding:
Agents decide their availability update strategies, construct their bids at the current round and bid.
Termination checking:
Auctioneer checks termination condition based on the bids collected in the current round.
Winner determination:
Auctioneer computes a new provisional schedule and informs agents whether they win or lose in current round
Termination condition satisfied?
No
Yes
End

Figure 5-1 Flow chart of the iterative bidding procedure for NCSS problems

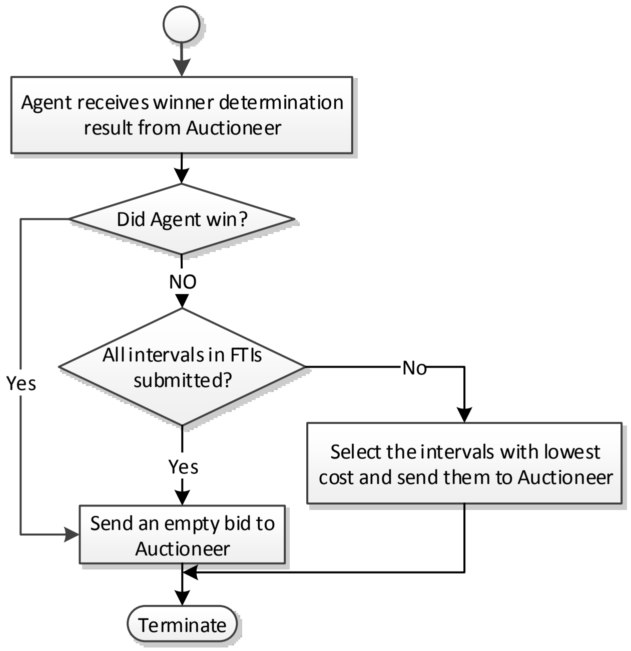

Agent receives winner determination result from Auctioneer
Did Agent win?
NO
Yes
All intervals in FTIs submitted?
No
Select the intervals with lowest cost and send them to Auctioneer
Yes
Send an empty bid to Auctioneer
Terminate

Figure 5-2 Agents' myopic bidding strategy at a specific round

5.4.2 Bid screening and termination checking

Once bids are received from the agents, the auctioneer first screens out all the invalid bids. Those bids will not be considered in the subsequent winner determination procedure. Invalid bids are defined as having (1) any time slots that were submitted in previous rounds; or (2) new time slots from agents who have already declared their final bidding status in a previous round.

The auctioneer then checks the termination condition against the valid bids. The auction terminates if there are no new availability updates for all the valid bids in this round. That is, each agent that bid in the last round has either submitted an empty bid or withdrawn from the bidding process. After the auction terminates, the auctioneer implements the final schedule. If the termination condition is not satisfied, the auctioneer will update its agents' available time slot pool by adding the newly submitted time slots to those already submitted in previous rounds and solve the winner determination model using the updated availability information as input.

5.4.3 Winner determination

The auctioneer needs to compute a new provisional schedule in each round until the auction has terminated. At round t, the new provisional schedule S_t solves the provider's decision problem model with updated availability from all bidding agents at round t as input. It is possible that multiple schedules could have the same optimal overall weight. Which optimal schedule the auctioneer will find first is determined by a combination of multiple factors, such as the design and configuration of the winner determination algorithm and the organization pattern of the input data. After winner determination, the auctioneer will inform all the bidding agents with the results as to whether they win or lose in round t. After receiving the results, the agents will decide their strategy on availability updating and start a new round of bidding. It is important to note that the winner determination model here is different from that of many other combinatorial auctions, in which the losing bids will not be considered in future rounds (Vries & Vohra, 2003). In our model, the bid from an agent is just a new addition to its already submitted availability. When computing the provisional schedule, the winner determination algorithm will consider all the time slots submitted from an agent during the current and previous rounds. In addition, the provisional schedule is determined by the updated availability at the current round. It is not affected by the bidding sequence(s) in previous rounds.

5.5 Properties of the iterative bidding framework

In the design of the iterative bidding framework, agents bid according to the myopic bidding strategy described in Figure 5-2. As I have assumed that agents are self-interested, a question arises naturally: will the agents really follow the myopic strategy? I now study the iterative bidding framework from the incentive compatibility perspective. I prove that the myopic bidding strategy I have designed is the dominant strategy for agents, as stated in the following proposition.

Proposition 1 *Given the proposed iterative bidding mechanism, myopic bidding is the dominant strategy for agents.*

Proof: It is clear that if an agent has already been awarded in the previous round, there is no reason for it to add new time slots in the current round because more availability will increase the upper bound of its preference violation cost. Therefore, it will follow the myopic strategy by reporting an empty bid. Next, consider the situation where an agent is not awarded in the previous round. Assume that the agent has reported its first $k-1$ time slots in its FTI during the previous rounds. If the agent follows the myopic strategy, it should add the kth time slot at the current round and update its availability to the first k time slots. To compare with the myopic strategy, I present here an *alternative strategy* for the agent, in which it reports first $(k+1)$ time slots. In the following I will prove that the myopic strategy weakly dominates the alternative strategy. Consider three cases:

Case#1: The agent is not awarded in the current round, no matter it submits its first k or $k+1$ time slots. In this case, both its first k and first $k+1$ time slots end up with the same preference violation cost, which is c_0. There is no difference between the myopic and the alternative strategies.

Case#2: The agent is awarded by submitting its first k time slots. In this case, the agent must be awarded by reporting its first $k+1$ time slots because its first k is a subset of first $k+1$. Since the awarded time slot can fall into any one of the submitted time slots, I compare the expected preference violation cost of the myopic strategy and that of the alternative strategy. Let a_k denotes the number of time slots, which costs c_k , that the agent can possibly be allocated to. Since $c_1 \leq c_2 \leq c_3 \leq \cdots \leq c_k \leq c_{k+1}$, then $c_{k+1}\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} a_i c_i$. Since $a_{k+1} > 0$, it follows that $a_{k+1}c_{k+1}\sum_{i=1}^{k} a_i \geq a_{k+1}\sum_{i=1}^{k} a_i c_i$. Adding $\sum_{i=1}^{k} a_i \sum_{i=1}^{k} a_i c_i$ to both sides of the inequality yields $\sum_{i=1}^{k} a_i \sum_{i=1}^{k} a_i c_i + a_{k+1}c_{k+1}\sum_{i=1}^{k} a_i \geq \sum_{i=1}^{k} a_i \sum_{i=1}^{k} a_i c_i + a_{k+1}\sum_{i=1}^{k} a_i c_i$, that is, $\sum_{i=1}^{k} a_i \sum_{i=1}^{k+1} a_i c_i \geq \sum_{i=1}^{k+1} a_i \sum_{i=1}^{k} a_i c_i$ which is equivalent to $\sum_{i=1}^{k+1} a_i c_i / \sum_{i=1}^{k+1} a_i \geq \sum_{i=1}^{k} a_i c_i / \sum_{i=1}^{k} a_i$.

Since adding the time slot $k+1$ will increase the feasible schedule space of the winner determination, the value of $a_1, a_2, \ldots, a_k$ will not be changed. The left-hand side of the last inequality can be interpreted as the expected cost of reporting the first $k+1$ time slots and the right-hand side can be interpreted as the expected cost of reporting only the first k time slots. It is clear that since the agent can be awarded by just reporting its first k time slots, the myopic strategy will always lead to the lowest expected preference violation cost.

Case#3: The agent is not awarded by reporting its first k time slots, but is awarded by reporting its first $k + 1$ time slots. In this case, although by the myopic strategy, the agent is not awarded at the current round, it always has the option of reporting its first $k + 1$ by repeatedly applying the myopic strategy in the next round. Given that the bidding sequence does not affect the winner-determination result, that is, the same set of time slot availability will result in the same provisional schedule, the agent will not lose any opportunity by adopting the myopic strategy.

It follows that the myopic strategy weakly dominates the alternative strategy with the first $k + 1$ time slots. This conclusion also applies to initial round of bidding. Since the provisional schedule before the initial round is empty, which can be interpreted as no agent is allocated a bundle. Therefore the best strategy for agents' initial round bidding is myopic strategy. That is, at the first round, an agent should bid with its lowest cost time intervals in its FTIs. By mathematical induction, it follows that, myopic bidding is the dominant strategy for agent given the proposed iterative bidding mechanism. ■

5.6 Iterative bidding with partial allocation during each round

The iterative bidding procedure I have proposed computes provisional allocation during each round. It does not permanently award time slots to customers until the termination condition has been reached. The procedure may reach higher quality solutions since it collects more agents' availability along the process of bidding. However, as the bidding proceeds, the size of the winner determination problem will increase continuously. Since I have shown that the winner determination problem is NP-hard, it follows that for a service scheduling problem with a large number of customers, winner determination will be slowed down considerably as more availability information is added. As a variant of the proposed iterative bidding procedure, it is possible to award the provisional allocation to customers during each round. In the subsequent round, those awarded time slots will be removed from the service provider's service time inventory, the awarded customers will withdraw from the bidding process, and the customers who were not awarded in the current round will construct their bids based on the updated inventory. The service provider will solve the winner determination problem formulated by the updated inventory and the bids submitted in the current round. This process allows the size of the winner determination problem to decrease along the iterations, as both the size of the provider's inventory and the number of bidding agents decrease. The bidding terminates in fewer rounds than the original procedure.

5.7 Simulation results: information revelation and efficiency analysis

By designing an iterative bidding framework, agents only need to reveal their availability information when it is necessary. In addition, the higher system transparency makes adoption of the framework easier. However, these benefits are gained at the cost of efficiency. If at the termination of bidding all the agents have revealed their full availability, the winner determination algorithm will compute an optimal schedule which maximizes the sum of the weights of awarded agents. However, when bidding terminates before all feasibility information has become known to the diagnostic service agent, the optimality of the solution is not guaranteed. In this section, I evaluate the information revelation/efficiency performance of the proposed approach through a computational study.

Given a solution schedule, the measure of its efficiency is defined as the ratio between its overall weight and that of an optimal solution for the same problem instance. The measure of information revelation is the ratio of the revealed availability of all agents when the solution is reached and to their complete availability. Intuitively, submitting more availability incurs higher information revelation, which increases the expected preference violation cost.

I used ILOG CPLEX 12.1 as optimization engine for solving the winner determination model, with the set of bids from agents as the input. The iterative bidding control logic is coded using the OPL Script language (Van and Michel, 2000). The control module and the optimization engine were integrated using the ILOG OPL environment (http://www. 01. ibm.com/software/integration/optimization/cplex-optimization-studio).All experiments were conducted on a PC with a 2.4 GHz CPU and 4 GB of memory.

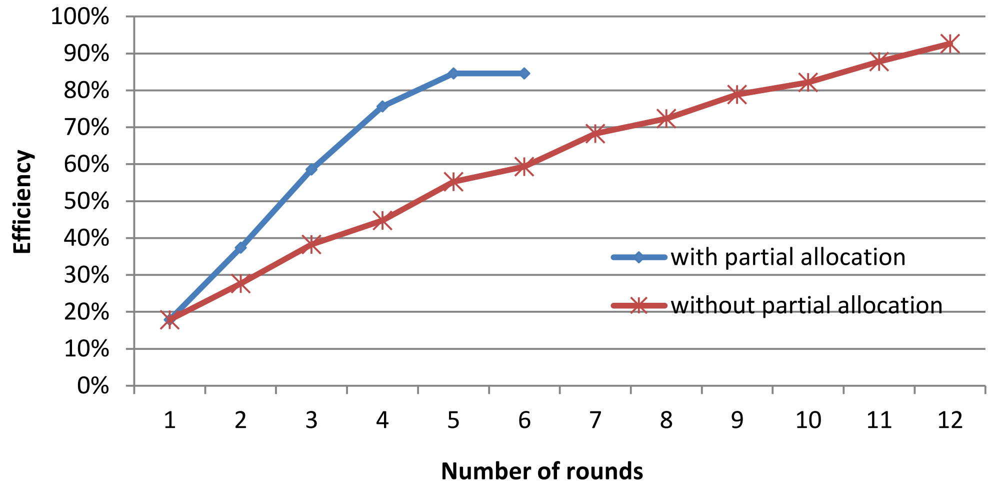


Figure 5-3 Efficiency increment during iterative bidding

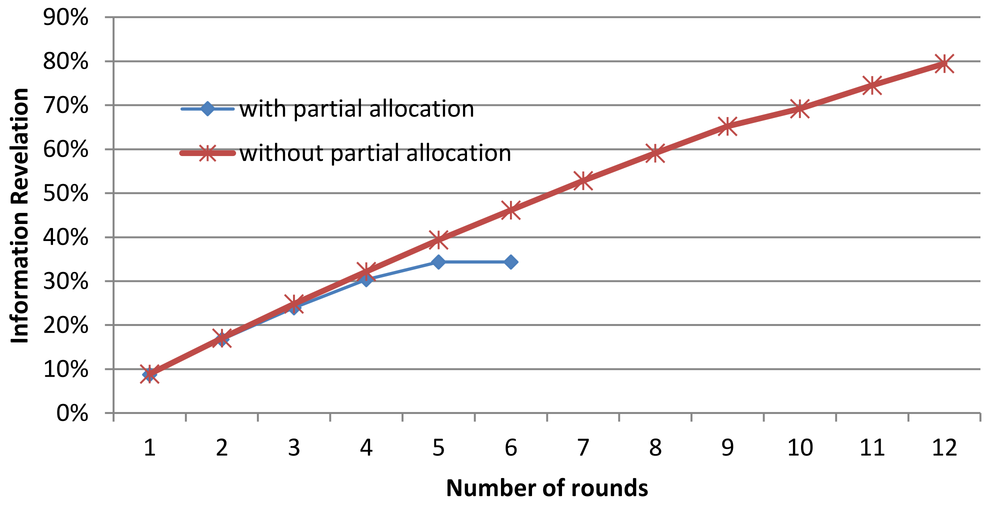


Figure 5-4 information revelation increment during iterative bidding

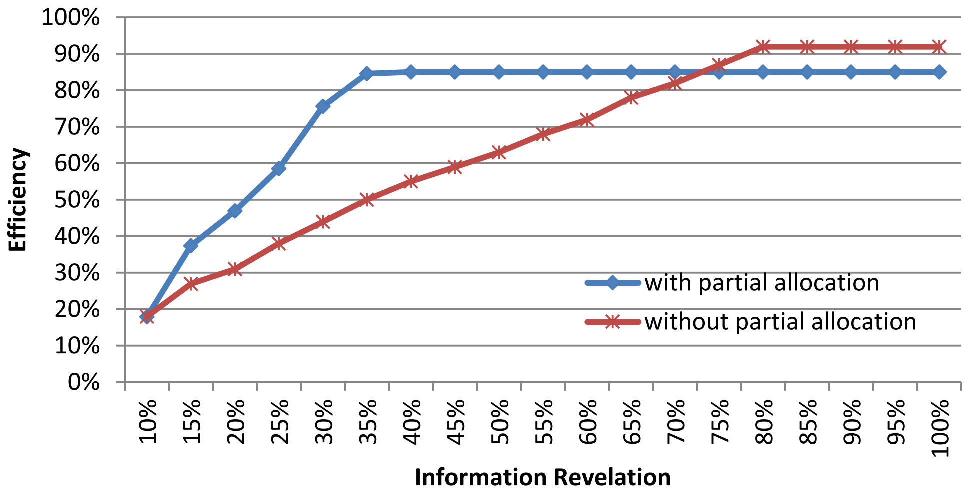


Figure 5-5 Trade-off between efficiency and information revelation

I generated a set of test problem instances by fixing the service provider's time slot inventory at 20 and the number of customers at 50. Customers' weights were drawn from a uniform distribution ranging from 1 to 3. The processing times for agents' requests are identical and restricted to one time slot. For each agent, I randomly selected a set of time intervals from the service provider's available time inventory to form its FTI. The sizes of the agents' FTIs were drawn from a uniform distribution in the range of 8 to 16 with a mean of 12.The length of the time intervals in FTIs is restricted to one. The time intervals in the FTIs were randomly ordered.

I solved the set of problem instances using the proposed iterative bidding framework and computed the average efficiency and information revelation at each round of bidding. The bidding processes without partial allocation usually terminated within 12 rounds, which is the mean of the size of the FTIs. The bidding processes with partial allocation usually terminated within 6 rounds, which is, as expected, much faster than the bidding without partial allocation.

Figures 5-3 and 5-4 show the efficiency and information revelation increment during the bidding process. At round 6, the modified bidding procedure with partial allocation achieves on average of 84% efficiency, whereas the original bidding procedure without partial allocation achieves on average 93% efficiency at round 12. The bidding procedure with partial allocation is essentially a ‘greedy’ distributed search algorithm which can find a solution quickly. However, its solution quality can be compromised. The bidding procedure without partial allocation involves backtracking. However, it normally reaches a higher quality solution with additional bidding rounds. From Figure 5-4, it is clear that the information revelation of bidding with partial allocation is always lower than that of bidding without partial allocation, and that this difference increases along the bidding process. Compared to bidding without partial allocation, it appears that bidding with partial allocation can achieve a reasonably good solution with much less computation costs and information revelation. Figure 5-5 shows the trade-off between efficiency and information revelation; as expected, high efficiency demands more information revelation. The results confirm that increasing information revelation has a diminishing return in efficiency.

Bidding with partial allocation can reach 84% efficiency with only 34% information revelation, whereas bidding without partial allocation needs to double the information revelation (70%) in order to reach the same efficiency level. For bidding without partial allocation, a solution with 93% efficiency demands information revelation of 79%. Since it is the agents that decide when to stop submitting more availability information to the auctioneer, the bidding procedure actually provides them with the option of setting their respective information revelation limits based on their own calculation of the costs caused by information revelation. In this experiment, I did not consider the situation where agents have information revelation limits. However, Figure 5-5 gives an indication of the efficiency that can be reached given various levels of information revelation.

5.8 Summary

In recent years, the economy has evolved from manufacturing to services. Service supply chain management has become an important research area with significant practical implications. Scheduling non-commercial services for self-interested customers who behave strategically to protect their private information is a challenging problem to resolve in accordance with the different objectives of service provider and customers. In non-commercial service scheduling environments, no payment transfers are allowed, which eliminates the possibility of designing price- or payment-based mechanisms to balance the supply and demand. I have proposed a bidding framework for scheduling non-commercial services and evaluate its efficiency and information revelation performance through theoretical analysis and computational experiments. I show that, under the proposed auction mechanism, myopic bidding is the dominant strategy for customers. In terms of the efficiency and information revelation performance, the computational study shows that bidding with partial allocation can find a reasonably good solution with much less computation costs and information revelation. For both cases of bidding, with and without partial allocation, increasing information revelation has a diminishing return in efficiency.

Chapter 6 Applications

This chapter presents the application of the I*b*SCHF to two problem domains: service mass customization under capacity constraints and appointment scheduling in a health care system. The objective is to demonstrate the applicability of I*b*SCHF to both commercial and non-commercial service scheduling domains, rather than providing complete solutions to these problems.

6.1 Applying framework

The I*b*SCHF can be applied to various service scheduling domains in which customers compete to schedule their service activities to make their best use of resources. To apply the framework in a problem domain, the first step is identifying customers and modeling them as agents. To carry out this step, an agent's job, their constraints, and their valuations over different schedules need to be specified. The second step is the configuration of the bidding process, which includes modeling the winner determination problem, specifying the bid structure and the bid update rules.

The rest of this chapter demonstrates how the two-step procedure is used to configure the I*b*SCHF to solve two different problems.

6.2 Service mass customization under capacity constraints

Mass customization aims at producing what customers need with near mass production efficiency. It can be seen as a collaborative optimization process between a company and its customers, with the goal of finding the best match between the company's capabilities and their customers' needs. A company's core capabilities are the basis of its product families and their successive platforms (Meyer and Utterback1993). These capabilities are reflected in the people and assets applied to the development of new products. A company's capabilities can be represented by its product family architecture (PFA) (Tseng and Jiao 1996; Jiao and Tseng 1999) which consists of a common base, a differentiation enabler, and a configuration mechanism. While a PFA can serve as a systematic protocol that customers can use to navigate through a company's capabilities and define their own requirements, capabilities can also be organized and presented using scalable product family design (Simpson et al. 2001) and configurational product family design (Du et al. 2001; Ulrich1995). In scalable product family design, a variety of customer needs are satisfied through the configuration of scaling variables which are used to "stretch" or "shrink" the product platform in one or more dimensions. Configurational product family design, on the other hand, aims at developing a modular product platform on which product family members are derived by adding, substituting, and/or removing one or more functional modules. The search for a better match between a company's capabilities and their customer's needs has been the central theme in mass customization literature for more than a decade (Jiao et al.2007; Simpson 2004; Da Silveira et al. 2001). In this chapter, a different perspective is taken to examine the impact of a company's *capacity* on product customizability and customer value. Here the term capacity is defined as a company's ability to produce customized products for a group of customers within a predefined time schedule.

It is imperative to consider a company's capacity constraints in customization decision making when production schedules are important to customers. This is particularly true in service customization. Unlike product manufacturing, service production usually involves customer's labour in the process (i.e., co-production), or it requires the physical presence of the customer. Common examples can be seen in health care offices, buffet restaurants, and travel services. For service customers, it is desirable to have convenient production schedules because they need to be physically present during service production. In addition, the service provider's capacity is perishable, as service operations cannot rely on inventories to adjust to demand fluctuations. Perishability alludes to the time-sensitive nature of a service provider's capacity to produce a service (Sampson 2001). In service customization, capacity constraints directly affect customers' satisfaction, as well as provider's profitability. Therefore, capacity constraints should be integrated into service customization decision making.

To motivate the research from a practical perspective, consider the case of the mass customization of travel packages. Major online travel brands such as Expedia Inc.

(Expedia.com), Opodo (Opodo.com), and Orbitz Worldwide (Orbitz.com) are giving their customers the tools to customize their own adventures in the form of "build your own package". Compared to pre-packaged vacations, customized packages are more attractive to customers because everyone's travel experience is unique and personal. A customized vacation package usually includes one or more of the following components: flight reservation, hotel reservation, car rental, and tickets to entertainment events. For a specific destination and a specific time window, the capacity limits of these components restrict customers' options and affect the customizability of travel products. This is particularly the case during high seasons, when the capacities of service providers are stretched to their limits. Similar situations occur in manufacturing mass customization. For example, in configurational product family design, a customer customizes its individual product by adding a group of functional modules to a base product. If a particular module takes excessively longer time to be delivered due to the manufacture's capacity constraints, the customer may switch to an alternative module or even cancel the order.

This chapter addresses the capacity aspect of mass customization. Specifically, it answers the question: Given limited capacity, how can a company maximize the value provided to its customers by coordinating customers' customization requirements? The main objective of the proposed approach is to maximize value across a large group of customers, which is, in economics terms, to maximize the social welfare (Mas-Colell et al. 1995). To facilitate clear formulation of the problem and meaningful presentation of the solution, the scope of the chapter is restricted to service customization settings. However, the proposed model could be applied to manufacturing customization. In this chapter, Service Customization under Capacity Constraints (SCCC) is modeled as an optimization problem. The contribution to the literature is two-fold. First, customers' customization decision making is integrated with a company's capacity constraints, which is of particular relevance in service customization settings where a provider's capacity is perishable and often expensive to expand. Second, at the system level, the overall value provided to customers is maximized by coordinating customers' customization requirements through auction-based multilateral negotiations. It is assumed that a company's objective is to maximize overall customer value. This objective is desirable because, in the long run, a company can improve its profit only by providing customers with high value-added products and services.

6.2.1 Centralized problem formulation

This section provides a centralized formulation of the SCCC problem, which consists of a group of customers and a service provider. The customers want to customize the service products. To provide a common design domain, the provider is assumed to adopt a configurational product family design approach (Du et al. 2001; Ulrich 1995) such that it can present its capabilities in the form of a set of building blocks (services). Customers can customize the product by choosing a base product (a pre-defined group of services) and adding optional services according to their preferences. A customized product is a package of services chosen by a customer. For example, a vacation package can include transportation services, accommodation services, and additional entertainment activities. For a provider, a service has a capacity limit which is defined as the number of customers the service can accommodate during a specified time window. The customer attaches a value to each package of services.

Formally, the SCCC problem consists of a set of n customers and a set of m services. A customer can configure its service package by selecting a group of services. A service package has to include a pre-configured set of services, that is, the base configuration, denoted $\bar{S}$. For service i, its capacity is limited by capacity (i). Let E_j be the set of service packages which are acceptable by customer j (i.e., feasible packages) and E be the union of the sets of acceptable service packages from all customers, so that $E = \cup_{j=1\ldots n} E_j$ Let $v_j(B)$ be the value of customer j attached to the service package $\in E$ $.v_j(B) > 0$ if $B \in E_j$; $v_j(B) = 0$ otherwise. Let $x_j(B) = 1$ if the bundle $B \in E$ is allocated to customer j and zero otherwise.The SCCC problem involves the selection of a set of service packages for customers such that the service provider's capacity constraints are respected and, at the same time, the sum of the customer value (social welfare, in terms of microeconomics) derived from the selected packages is maximized. The problem can be formulated as the following integer programming.

$max \sum_{j=1}^{n} \sum_{B \in E} x_j(B) v_j(B)$

subject to

$\sum_{B \in E} x_j(B) \leq 1, \quad j = 1, \dots, n$ (1)

$\sum_{B \ni i} \sum_{j=1}^{n} x_j(B) \leq capacity(i),\ i = 1 \dots m$ (2)

$\sum_{B \in E} x_j(B) = \sum_{B \in E_j} x_j(B), \quad j = 1, \dots, n$ (3)

$\sum_{B \in E} x_j(B) = \sum_{B \supseteq \bar{S}} x_j(B), \quad j = 1, \dots, n$ (4)

$x_j(B) = \{0,1\}, \quad B \in E, \quad j = 1, \dots, n$ (5)

Constraints (1) ensure that a customer can only obtain one service package. Constraints (2) ensure that the allocation of a service to customers does not exceed the capacity limit of the service provider. The set of constraints (3) ensure that if a package is assigned to a customer, it must belong to the set of product configurations that are acceptable to that customer. These constraints prevent the provider from assigning customers packages which they are not willing to accept. Constraints (4) enforce the selection of the base configuration in each awarded package. Constraints (5) are a set of integer constraints.

Theorem 1: *The problem of service customization under capacity constraints (SCCC) is NP-hard.*

Proof: To show that SCCC is NP-hard, consider a special case in which $E_j = E$ for all $j = 1, \dots n$ and $\bar{S} = \phi$. In this case, constraints (3) and constraints (4) always hold. The relaxed model is a set-packing problem, which is NP-complete (Karp, 1972). It follows that, as a general case, SCCC problem is NP-hard■.

The SCCC is an integer programming model which takes customer value as input. The key question to be asked here is how the values that each customer assigns to the packages can be obtained. Computing value from product configurations can be customer-specific. One approach, suggested by Tseng and Du (1998), is to use methods designed to measure consumer preferences in marketing research, such as conjoint analysis (IntelliQuest 1990). Conjoint analysis assumes that a product could be described as vectors of attributes, and each attribute can include several discrete levels.

To apply conjoint analysis to the SSCC, each service is modelled as an attribute and the discrete levels of attributes are restricted to 1 (service included) and 0 (service not included). As the SCCC requires customers' complete valuation on all feasible packages, computing a value for each and every configuration may become impractical when the range of feasible packages becomes large. Although customers can determine the value of feasible packages, they may be reluctant to report the value back to the service provider because, by the definition of the private value model, value is the highest price that a customer is willing to pay for a given package. In many cases, these prices are sensitive private information. The proposed iterative bidding framework in chapter 4 computes high quality solutions to SCCC problem without requiring valuations from customers. The proposed auction is a price mechanism in which a provider coordinates the customization requirements among its customers by adjusting the prices of service packages.

6.2.2 Service mass customization under capacity constraints using I*b*SCHF

Bidding

Each agent has a valuation function expressing its values on different service packages. An agent's valuation can be expressed as an XOR-bid. For example $< Package, price >$ expresses the agent's willingness to pay the *price* for the services included in the *Package.* We also assume that the reserve prices are common knowledge. For the first round of bidding, agents use the reserve prices as the asking prices. At the beginning of round t, a customer agent g selects a feasible package that maximizes its utility function given the asking prices, and generate the bid.

Bids screening and termination

After receiving bids from the agents, the service provider agent first screens out any invalid bids. Invalid bids are defined as those with (1) any bidding price for a package which is below the highest bidding price that same package received in previous rounds, (2) higher prices from customers who have already declared their final bidding status in a previous round, and (3) packages which do not contain the base configuration or that violate other configuration rules.

Winner determination

The winner determination model designed is to select a subset of the bids submitted by the customers such that the overall bidding price of the provisional allocation is maximized and the capacity constraints of the provider are not violated. Let N^t be the set of customers that submitted their bids at round t and $p_j^t(B_j^t)$ be the bidding price of customer j at round t, $j \in N^t$, where B_j^t is the package submitted by customer j at round t. Let $Z_j = 1$ if customer j wins and $Z_j = 0$ otherwise. The winner determination model can be expressed using the following integer programming.

$$max \sum_{j \in N^t} Z_j p_j^t(B_j^t)$$

subject to

$$\sum_{\substack{j \in N^t \\ B_j^t \ni i}} Z_j \leq capacity(i),\ i = 1 \ldots m \qquad (6)$$

$$Z_j = \{0,1\},\quad j \in N^t \qquad (7)$$

Constraints (6) ensure that the bids awarded in a provisional allocation do not violate the provider's capacity constraints. Constraint (7) is a set of integer constraints.

Price Update

In the bidding procedure for the service mass customization problem, the prices are updated according to the price update rules in the I*b*SCHF.

A worked example is presented next, to demonstrate the application of the I*b*SCHF to travel package customization.

6.2.3 A worked example

This subsection presents a worked example to demonstrate the bidding process of applying the I*b*SCHF to the travel package customization problem. Suppose a travel agency offers a "build your own package" tool to its customers so they can customize their vacation packages for a 7-day holiday at a popular destination. Customers should travel to the destination on Day 1 and return on Day 7. The agency offers a list of travel components including flight reservations, hotel reservations, car rental, and tickets to entertainment events. There are multiple services for each of the components to accommodate various customer preferences. For example, a Departure Ticket (DT) can be scheduled in the morning (DT-1), afternoon (DT-2), or evening (DT-3). For illustrative purposes, an example of an unrealistically small number of customers (five customers) is set up. The available services and their respective capacities are summarized in Table 6-1. Table 6-2 shows the customers' feasible packages and their valuations of them, where B(a, b) represents feasible package b from customer a. The base configuration includes one and only one service of each of the components DT, RT, and HL. Customers can have one to five services from the component ET. To limit the number of rounds of bidding, high reservation prices are set for the packages (see Table 6-2). Submitted bids, provisional allocation, provider's revenue, and customer's value at each round of bidding are summarized in Table 6-3. ε is set to be five. The auction terminates at round 12 with overall customer value at 7370. Compared to the optimal value of 7790, the auction reaches 95% efficiency in this example. The sum of the prices paid by the customers (i.e., the provider's revenue) is 7240, which is close to the overall solution value due to competition among customers. The provisional allocations along the bidding process manifest the heuristic search guided by the changing package bidding prices.

Table 6-1 Summary of service capacity

Service ID	Service Description	Capacity
DT-1	Departure Ticket in the morning of Day 1	3
DT-2	Departure Ticket in the afternoon of Day 1	2
DT-3	Departure Ticket in the evening of Day 1	2
RT-1	Return Ticket in the morning of Day 7	2
RT-2	Return Ticket in the afternoon of Day 7	2
RT-3	Return Ticket in the evening of Day 7	3
HL-1	First-class hotel	1
HL-2	Second-class hotel	3
HL-3	Motel	2
ET-1	Sporting event ticket	2
ET-2	Performing arts ticket	2

ET-3	Museum ticket	3
ET-4	Cruise trip ticket	3
ET-5	Fine dining ticket	2

Table 6-2 Customers' feasible packages and corresponding reservation prices and value

Customer	Feasible Packages	Reservation Price ($)	Value ($)
Cus#1	B (1,1) = {"DT3" "RT3" "HL3" "ET2" "ET3"}	1410	1445
	B (1,2) = {"DT2" "RT1" "HL1" "ET1""ET3" "ET4"	2200	2250
	B (1,3) = {"DT3" "RT2" "HL2" "ET4"}	1830	1870
Cus#2	B (2,1) = {"DT1" "RT3" "HL1" "ET1" "ET2"}	2120	2145
	B (2,2) = {"DT1" "RT1" "HL1" "ET1" "ET4" "ET5"}	2320	2360
	B (2,3) = {"DT3" "RT3" "HL2" "ET1" "ET3""ET4"}	2060	2085
Cus#3	B (3,1) = {"DT3" "RT3" "HL1" "ET1" "ET2" "ET3"	2210	2235
	B (3,2) = {"DT1" "RT2" "HL1" "ET1" "ET3""ET4"}	2360	2370
Cus#4	B (4,1) = {"DT2" "RT1" "HL3" "ET1""ET2" "ET3"	1660	1695
	B (4,2) = {"DT2" "RT1" "HL2" "ET1" "ET3"}	1730	1740
Cus#5	B (5,1) = {"DT3" "RT3" "HL2" "ET5"}	1610	1660
	B (5,2) = {"DT1" "RT3" "HL1" "ET1" "ET3" "ET4"	2360	2375
	B (5,3) = {"DT2" "RT2" "HL1" "ET2""ET4" "ET5"}	2130	2135
	B (5,4) = {"DT3" "RT3" "HL3" "ET3" "ET5"}	1290	1295
	B (5,5) = {"DT1" "RT3" "HL2" "ET3""ET4"}	1910	1945

Table 6-3 Submitted bids, provisional allocation, provider's revenue, and customer's value at each round of bidding

Round #	Submitted Bids	Provisional Allocation	Provider Revenue	Customer Value ($)
1	B (1,2) , B (2,2) , B (3,1) , B (4,1) , B (5,1)	B (2,2), B (4,1) , B (5,1)	5590	5715
2	B (1,2) , B (2,2) , B (3,1) , B (4,1) , B (5,1)	B (2,2), B (4,1) , B (5,1)	5590	5715
3	B (1,2) , B (2,2) , B (3,1) , B (4,1) , B (5,1)	B (2,2), B (4,1) , B (5,1)	5590	5715
4	B (1,3) , B (2,2) , B (3,1) , B (4,1) , B (5,1)	B (1,3), B (2,2), B (5,1)	5760	5890

5	B (1,3) , B (2,2) , B (3,2) , B (4,1) , B (5,1)	B (1,3), B (3,2), B (5,1)	5800	5900
6	B (1,3) , B (2,2) , B (3,2) , B (4,1) , B (5,1)	B (1,3), B (3,2), B (5,1)	5800	5900
7	B (1,3) , B (2,2) , B (3,2) , B (4,1) , B (5,1)	B (1,3), B (3,2), B (5,1)	5800	5900
8	B (1,3) , B (2,1) , B (3,2), B (4,1) , B (5,1)	B (1,3) , B (2,1), B (4,1) , B (5,1)	7240	7370
9	B (1,3) , B (2,1) , B (3,1), B (4,1) , B (5,1)	B (1,3) , B (2,1), B (4,1) , B (5,1)	7240	7370
10	B (1,3) , B (2,1) ,B (3,2) , B (4,1) , B (5,1)	B (1,3) , B (2,1), B (4,1) , B (5,1)	7240	7370
11	B (1,3) , B (2,1) , B (3,1) , B (4,1) , B (5,1)	B (1,3) , B (2,1), B (4,1) , B (5,1)	7240	7370
12	B (1,3) , B (2,1) , B (3,1) , B (4,1) , B (5,1)	B (1,3) , B (2,1), B (4,1) , B (5,1)	7240	7370

6.2.4 Incentive Issues

Given the assumed customers' private value model, no customer bids above their valuation. In all cases, customers will not get negative payoffs, which encourages them to participate in the auction. However, understanding the incentives that a company has for setting up and conducting the proposed auction requires some explanation of the company's objectives for auction design. In auction design there are two common objectives an auctioneer may have. The first is economic efficiency, and the second is revenue maximization (de Vries and Vohra, 2003). An auction is economically efficient if the allocation of objects to bidders chosen by the auctioneer maximizes the overall values of the bidders. Economic efficiency is supported by well-developed auction theories. A typical example is the canonical Vickrey-Clarke-Groves (VCG) mechanism (Vickrey 1961; Clark 1971; Groves 1973) which simultaneously achieves incentive compatibility and efficiency and has guided the design of many auctions. As a result, the majority of the auction literature takes economic efficiency as their design objective.

It is argued in Parkes and Kalagnanam (2005) that the goal of economic efficiency is well suited for the design of stable long-term markets that will form the basis for repeated trading. They expect that efficient markets will come to dominate the electronic market landscape based on their experience with procurement auctions deployed with a large chocolate manufacturer (Hohner et al., 2003). In the context of mass customization, economic efficiency is also desirable for a company seeking to build long-term business relationships with their customers. It is agreed in mass customization literature that one of the major objectives of mass customization is to improve customer value. In the long run, a company can only improve its profit by providing customers with high value-added products and services. The long term benefits brought by efficient auction design provide an incentive for companies to adopt economic efficiency as their auction design objective.

The objective of revenue maximization (optimal auction design), on the other hand, maximizes the auctioneer's revenue. Optimal auctions maximize the seller's revenue at every transaction, which are perhaps more appropriate for a one-shot procurement problem, and in a settings where the buyer has considerable market power (Parkes and Kalagnanam, 2005). However, even if a company only cares about short term benefits and wants to get the most out of every transaction, an efficient auction design may still be the more reasonable choice, especially when iterative bidding is used as an implementation structure. This is because there are no known optimal (i.e. revenue-maximizing) general-purpose combinatorial auctions, iterative or otherwise (Parkes, 2006). In fact, the dynamic exchange of value information between bidders that is enabled within iterative combinatorial auctions is known to enhance revenue and efficiency in single-item auctions with correlated values (Milgrom and Weber, 1982). One should expect efficient iterative combinatorial auctions to retain this benefit over their sealed-bid counterparts (Parkes, 2006). Therefore, from both long-term and short-term perspectives, a company has clear incentives to deploy an efficient combinatorial auction.

The I*b*SCHF is an efficient auction design which is implemented using an iterative bidding process. The bidding process is guided by a price mechanism. The revenue that the auctioneer collects is the sum of the bidding prices from the awarded customers at winner determination. Given the design of the bidding procedure, the company's revenue is guaranteed to increase along the bidding process and reach its highest at termination. Despite the formulation of the economic efficiency objective of the SCCC, the iterative bidding structure itself achieves high seller revenue in the same spirit of many real-world iterative auction applications, which supports our claim that the proposed model provides incentives to the seller. The performance gained by applying the I*b*SCHF on general SCCC problems is evaluated through a computational study in the next section.

6.2.5 Value and revenue performance under various product customizability

Products with a higher level of customizability will likely meet individual customer needs better. However, a higher level of customizability often leads to higher costs. To manage the customization costs and improve operational efficiency, service providers usually restrict customers' choices in choosing service combinations by imposing configuration rules. The proposed customization model allows providers to adjust the customizability of packages by defining different base configurations. When customizing a package, a customer is required to incorporate the services defined in the base configuration into the package. In terms of platform-based product development, the base configuration serves as a base product on which customers build their customized products. In this section, the value and revenue performance of applying the I*b*SCHF to SCCC problems is validated under various levels of product customizability imposed by the service provider. The proposed framework is also compared with the commonly used first-come-first-served capacity allocation approach in terms of solution values. The design of the set of testing data used for the experiments is described as follows.

6.2.6 Design of the testing data

The customization environment in which the computational study is conducted is the one described in the worked example. However, to demonstrate the practical relevance of the experiments, the number of customers and the capacity of services are now increased to a realistic scale. Customer value is generated from common pricing schemes found in online travel auctions. In travel auction websites, such as eBay Travel (http://www.ebay.com), Luxury Link (http://www.luxurylink.com), and Sky Auction (http://www.skyauction.com), a package to be sold has a "buy it now" price which is usually its regular retail price. A customer can purchase the package immediately at the regular retail price if unwilling to wait until the termination of the auctions. However, if the customer wants a bargain, they must participate in the auction.

The final auction price is determined by the market competition at the termination of the auction. A package also has a reservation price. The reservation price is often unknown to the customers. In the design of the testing data, it is assumed that there is a regular retail price for each of the available services, and the retail price for a package is the sum of the retail prices of the services included in the package. The reservation price for a package is set to be 40% of its retail price, since it is common in the online travel auctions that the termination price can be as low as 60% below the regular retail price. It is assumed that customers who enter the auction expect some discount. They are not interested in purchasing the package at a price higher than the regular retail price. Customer values for a package are randomly drawn from a uniform probability distribution between reservation price and its regular retail price. Ten SCCC problem groups are generated, with the customer number ranging from 100 to 1,000. For each group, ten instances are randomly generated. Service capacity is also allocated in proportion to the number of customers such that, for most of the instances, around 80–90% of the customers will be awarded a feasible package. For all instances, a customer's feasible package must contain one DT, one RT, and one HL.

6.2.7 Simulation results

In this section the value and revenue performance of applying the I*b*SCHF to an SCCC problem is validated under various levels of product customizability imposed by the service provider. For the computational study, three levels of product customizability are considered. The three levels are defined by different base configurations: Config#1 = {one of DT, one of RT, one of HL}, Config#2 = {one of DT, one of RT, one of HL, one of ET}, Config#3 = {one of DT, one of RT, one of HL, three of ET}. The numbers of services contained in the three configurations are 3, 4, and 6, respectively. The solutions computed under Config#1 are used as the baseline for comparison. For each group of problem instances, the optimal solution value under Config#1 is computed by solving the SCCC integer programming model presented in section 6.2.1 "Centralized problem formulation". The SCCC model is coded in ILOG Optimization Programming Language (http://www-01.ibm.com/software/websphere/products/optimization/) and the ten groups of problem instances are solved using ILOG CPLEX.

The flow control of the iterative bidding is coded in the OPL (Optimization Programming Languages) script language. A desktop PC with a 2.4G Intel CPU and 8 GB of memory is used to run the experiments.

Table 6-4 Customer value and provider revenue generated at different levels of package customizability

Group	Base-Config#1				Base-Config#2		Base-Config#3	
	(1) Optimal value	(2) Auction value	(3) Auction revenue	(4) First-come-first-served Value	(5) Auction value	(6) Auction revenue	(7) Auction value	(8) Auction revenue
1	$211,705	$210,535	$174,380	$166,420	$110,080	$96,585	$73,085	$57,890
2	$421,970	$418,100	$333,470	$326,270	$221,990	$197,225	$129,240	$102,740
3	$633,215	$618,880	$482,370	$493,610	$336,620	$294,485	$173,650	$137,970
4	$848,365	$846,295	$691,550	$662,700	$448,860	$397,790	$211,955	$166,980
5	$1,055,680	$1,039,410	$814,790	$816,505	$563,895	$503,075	$279,435	$219,160
6	$1,269,615	$1,245,415	$963,130	$954,235	$676,915	$599,330	$333,085	$259,360
7	$1,473,780	$1,453,190	$1,130,300	$1,128,480	$787,980	$696,880	$390,545	$303,210
8	$1,688,120	$1,680,505	$1,354,670	$1,294,280	$900,455	$802,365	$453,520	$353,030
9	$1,907,200	$1,889,915	$1,476,390	$1,497,350	$1,014,995	$899,630	$515,165	$402,940
10	$2,114,810	$2,101,835	$1,681,890	$1,655,410	$1,126,325	$994,805	$568,815	$443,030

The solutions computed by applying the I*b*SCHF are compared to the optimal ones computed by ILOG CPLEX. The first column of Table 6-4 shows the average optimal solution values for the ten groups of testing problems. The second and the third columns show the solution value and revenues computed by applying the I*b*SCHF, respectively. All customers are assumed to adopt final-bid-repeating and ε = 20 for all bidding. It is observed that the solution computed by applying the I*b*SCHF can achieve, on average, 98% of the optimal value across the ten groups of problem instances. The average revenue computed is approximately 78% of the optimal value.

To evaluate the impacts of package customizability on customer value, the testing problems are solved again with Config#2 and Config#3. When conducting the iterative bidding, all bidding packages which do not satisfy Config#2's and Config#3's configuration requirements are excluded at the bid screening stage. Columns five and six of Table 4 show the solution values and the revenues, respectively, with Config#2. It is observed that, on average, the solution value decreases to 53% of that of Config#1, and revenues decrease to 59% of those achieved with Config#1. If Config#3 is applied, the solution value will decrease to 27% of Config#1's value, and revenues will decrease to 28% of those achieved with Config#1. It is evident from the simulation results that reducing product customizability can significantly decrease both customers' overall value and provider's revenue.

The proposed customization approach is also compared to the commonly used first-come-first-served capacity allocation policy. For example, "build your own package" applications in the travel industry usually allocate a provider's capacity on a first-come-first-served basis combined with dynamic pricing strategies. Again, take travel package customization as an example. To compare the performance of an auction-based policy against that of a first-come-first-served capacity allocation policy, each policy is applied to the ten groups of SCCC testing problems. Column 4 of Table 6-4 shows the solution value of the first-come-first-served policy compared to the testing problems under Config#1. It is clear that the first-come-first-served policy achieves on average 78% of the value obtained by the auction-based approach.

6.3 Appointment scheduling in the health care system

Today's healthcare systems face increasing demands in both the number of patients and the services that patients require, which often stretches limited resources beyond capacity. More and more patients must be treated with the same limited resources and budget. Nevertheless, the quality of service cannot be compromised. In addition to the perceived quality of medical services that they receive, patients' satisfaction with their healthcare providers is also affected by their appointment booking experiences. Patients want more personalized care, which includes involvement in selecting appointment times with their preferred doctors.

Most of the government policies and researchers focus on improving the speed of access to the health care system and decreasing waiting time, but for non-urgent care, patients place a value on seeing the doctor of their choice, and on doing so at a convenient time. Using the discrete choice experiment method among 1153 patients, G. Rubin et al (2006) found that the waiting time to make an appointment was only important if the appointment was for a child or when it was for a new health problem. In that survey, participants were asked to make their choices in a questionnaire that offered three categories: speed of access (time to appointment), choice of doctor and choice of time (they could choose their preferred time for an appointment). For responders who were employed, choice of time was six times more important than shorter waiting time. Older patients, women and those with long-standing physical illnesses preferred to see their own doctor for their appointment and they were willing to wait longer to do so. Gerard, K et al (2008) used discrete choice experiments to determine the important factors that influence patient choice in the booking an appointment. From their overall responses, the factors influencing patient choice in booking appointments were, in order of importance: seeing a doctor of their choice; booking at a convenient time of day; seeing any available doctor; and having an appointment sooner rather than later. These findings clearly demonstrate that the current focus of policy makers on speed of access is oversimplified. In addition, evidence shows that when patients were matched and scheduled according to their preferred provider, quality of care is improved (O'Hare and Corlett 2004); also, matching patients with their preferred provider and offering them a convenient appointment time can decrease the number of no-shows and thereby increase operational efficiency (Barron 1980).
In 2005, survey results indicated that patients complain about their difficulty in obtaining an appointment at a convenient time. (Healthcare Commission 2005).In another survey, one in four (25%) said they had been put off from going to their GP practice because the opening hours were inconvenient (National Survey of Local Health Services 2006).

However, accommodating scheduling preferences across a large number of patients is particularly challenging due to three areas of complexities, namely collection complexity, allocation complexity and elicitation complexity. Collection complexity refers to the efforts needed to collect preferences information from patients. However, collecting complete preference information from a large number of patients is not an easy task because a patient's preference is usually not binary. Instead, it is a continuous variable that spans the spectrum from highly like to highly dislike. Moreover, a patient's preferences may change over time for the same patient. Some examples of factors that change preferences are changes in work schedule or in marital status; this fluidity is one of the reasons why the vast majority of appointment booking systems are not automated. They have to rely on human schedulers to negotiate with patients to collect preferences information, a practice which incurs high administrative costs to the healthcare system. Allocation complexity refers to the computation needed to compute high-quality service time allocations. Accommodating dynamic preferences can easily make mathematical models of the appointment booking process intractable (Gupta and Denton 2008). These issues are further complicated by the fact that patients are reluctant to reveal all their availability.

The proposed Iterative bidding framework for non-commercial services presented in chapter 5 can be properly applied to appointment scheduling problem. The next section demonstrates the multi-agent systems architectural for healthcare scheduling problem that can be used to apply the proposed framework.

6.3.1 The environment

Multi-agent systems architecture for appointment scheduling can be modelled as shown in Figure. 6-1. In this architecture, there are three types of agents that work collaboratively to achieve the overall scheduling functions of the system. The Patient agent represents the personal assistant of a patient. This agent has a user interface through which patients directly input their preferences and availability. A patient can program her preferences and availability into the agent and the agent can act on behalf of the patient to automatically interact with the hospital scheduler. This agent should also be equipped with optimization algorithms to compute the best strategy that it should take given the current scheduling situation and the patient's preferences and availability constraints. The Diagnostic Services

(DS) agent represents the hospital scheduler or the secretary of the hospital. Registration and lookup services for other agents are provided by the Director Facilitator (DF) agent. In this architecture, patient schedules are computed through the negotiation of agents. Patient agents and DS agent need to make their local decisions based on their objectives during the negotiation process. The patient agents' and DS agent's decision problem is formulated in the following section.

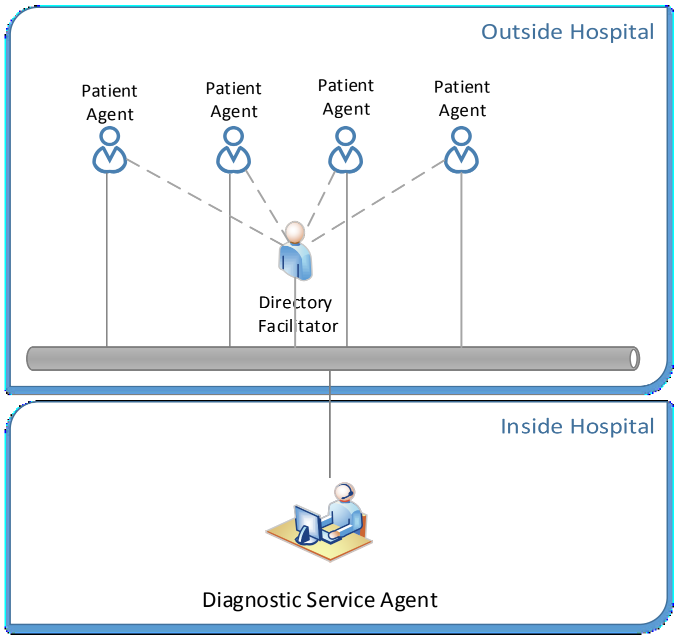


Figure 6-1 A multi-agent systems architecture for the appointment scheduling problem

6.3.2 Formulation of diagnostic service and patient agent decision problem

Patient scheduling is a multilateral decision making problem with the diagnostic service and patient agents as independent decision makers. The diagnostic service agent needs to decide how to schedule service requests to achieve its objectives and, at the same time, respect the patients' availability and their preferred doctor constraints. The decision facing a patient agent is how much preference information needs to be revealed in order to maximize the benefit for the patient.

A. Diagnostic service agent's local decision making problem

The diagnostic service agent receives a set of requests from patients' agents. Each request consists of a patient's preferences regarding her preferred time slot and preferred doctors. The model assumes the durations of all diagnose services are deterministic. A patient is assigned a weight scale by the diagnostic service agent, denoted as $\mathrm{w_j}$. Given the requests from patients and the available service time slots, the provider needs to solve an optimization problem: determining the allocation of limited service time slots to the requests so that the sum of the weights of the awarded requests is maximized. The diagnostic service agent will not assign patient p a time slot and a doctor outside her preferences.

Formally, let T be the set of time slots available at the time of scheduling; P be the set of patients who have diagnostic requests to be processed; D be the set of doctors; and $\mathrm{w_p}$ be the priority level assigned to patient p. Let $\mathrm{A_{dt}} = 1$ if doctor d is available at time slot t; let $\mathrm{R_{pdt}} = 1$ if patient p requests doctor d in time slot t; let $\mathrm{X_{pdt}} = 1$ if doctor d at time slot t is assigned to patient p. The patient scheduling problem for diagnostic services can then be formulated as follows.

$$Max \sum_{p \in P} \sum_{d \in D} \sum_{t \in T} X_{pdt} \, W_p$$

subject to

$$\sum_{d \in D} \sum_{t \in T} X_{pdt} \leq 1, \qquad \forall p \in P \qquad (1)$$

$$\sum_{p \in P} X_{pdt} \leq 1, \qquad \forall d \in D, \forall t \in T \qquad (2)$$

$$\sum_{p \in P} X_{pdt} \leq A_{dt}, \qquad \forall d \in D, \forall t \in T \qquad (3)$$

$$\sum_{d \in D} \sum_{t \in T} X_{pdt} = \sum_{d \in D} \sum_{t \in T} R_{pdt} \, X_{pdt}, \qquad \forall p \in P \qquad (4)$$

The set of constraints (1) ensures that any patient can only obtain one time slot. The set of constraints (2) ensures that a doctor's time slot can only be assigned to one patient. The set of constraints (3) ensures that if a doctor's time slot is assigned to a patient, that assigned time slot should belong to the doctor's working time slots. The set of constraints (4) ensures that a doctor's time slot should assigned be within a patient's requests.

B. Patient agents' Decision Problem

Each patient indicates its preferred time slot(s) and preferred doctor(s) in his agent interface, as depicted in Figure 6-2.

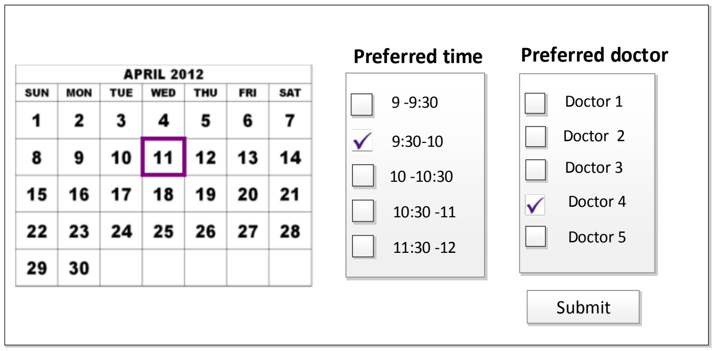


Figure 6-2 The patient agent interface

Each patients' preferences are their private information and are not known to the diagnostic service agent. We assume that a patient agent prefers some combination of time slot and doctor over others. The preferences can be quantified by associating a preference violation cost to each combination of time slot and doctor. We assume that a patient agent orders Preferred Combinations (PCs) according to the increasing order of their preference violation costs. That is, given an ordered PC, $c_1 < c_2 < c_3 < \cdots c_k \ldots < c_{|PC|} < c_0$ is known to the patient agent, where c_k denotes the preference violation cost of the kth combination in set of PCs, and c_0 denotes the preference violation cost of not being allocated any time slot. Patient agents try to avoid high cost assignments by not revealing their complete preferences. Since no payment is allowed in the patient scheduling setting, the possibility of applying the standard one-shot VCG mechanism (Clarke 1971, Groves 1973, and Vickrey 1961) or even its iterative implementations (Parkes 2006) is eliminated. The proposed iterative bidding framework for non-commercial services systematically evolves the solution towards an optimal one given the constraint that patient agents try to avoid high cost assignments by not revealing their complete preferred combinations.

6.3.3 Appointment scheduling using the proposed framework

If patients are modelled as agents and the hospital as the auctioneer, the appointment scheduling problem is mapped to a distributed non-commercial service scheduling problem. The agent modeling and bidding process configuration for an appointment scheduling problem are similar to those detailed in Chapter 5. We briefly describe the bidding process in the context of the appointment scheduling problem as follows.

- The DS agent (auctioneer) first collects the availability information of the hospital's resources and the doctors within the time window to be scheduled. Then, it sends messages to all its patient agents who have been registered by DF agent, indicating that the hospital is now ready to receive requests. The bidding process follows an iterative pattern.
- At the beginning of each round, a patient agent needs to decide whether it will submit additional preferences or not.
- Based on the bids and the available time slots, and on the availability of the doctors, DS agent computes a provisional schedule which includes the winning bids.
- The losing customers can bid in the subsequent rounds by adjusting their bids, i.e. they can select a combination of time slot and doctor with the lowest cost from the unrevealed part of their PCs and submit it to the DS agent.
- If a provisional schedule includes all the patients, or if there is no update of the bids from the losing patient agents, the bidding terminates and the current provisional schedule is implemented.

6.3.4 Summary

This chapter describes the application of the I*b*SCHF to two problem domains: service customization under capacity constraints and appointment scheduling in a health care system. Since service customization is a type of distributed service scheduling problem, the proposed iterative bidding framework becomes a natural solution. For the appointment scheduling problem, applying the I*b*SCHE is a novel way to unify the exploration of customers' preferences and the integration of hospital decisions within an auction structure.

Chapter 7 Design Specification and Implementation

This chapter presents the design and implementation of a prototype environment for the I*b*SCHF proposed in chapter 4. The main objective of developing such a prototype is twofold: 1) to test the feasibility of implementing the proposed distributed service scheduling system in a more realistic multi-agent environment; and 2) to evaluate the proposed system in terms of its communication costs and system responsiveness. In particular, the main tasks for the development of such a prototype include:

- Understanding the functional and non-functional requirements to develop the prototype in a distributed environment ;
- Designing a .NET-based iterative bidding system which integrates with the winner determination model implemented using ILOG OPL environment;
- Providing bidding interfaces for customers and an interface for the service provider for service definition;
- Developing a web service which enables customer agents to interact with the scheduling system;
- Evaluating the scalability of I*b*SCHF through the use of the developed prototype

7.1 Functional requirements

This section defines and describes the functional requirements that must be met in order to apply I*b*SCHF in a real distributed environment. Functionalities are defined from three different perspectives: the customer agents', the service provider agent's and the scheduling system's.

7.1.1 Customer Agents

a) Customer interactions require the following functionalities:

- **Login**: Allows customer agents to login to the scheduling system;

- **Edit feasible bundles:** Provides customers with the ability to edit bundles submitted to their agents if the bundle of time slots has not been executed in the schedule by the time a change has been requested;
- **Set status as standby:** Makes it possible for customers to set their status as standby for their preferred time slots; and
- **Cancel schedule:** Gives customers the ability to cancel their schedules.

b) Scheduling system interactions

The customer agent provides five fundamental functionalities on behalf of a customer: submit a bid, receive the winner determination result in each round, update a bid based on the received result, and provide the final result to the customer. In general, customers can achieve their business objectives and reflect their dynamic changes through the use of their agents. The following is the list of the required functionalities:

- **Retrieve available service time slots:** Allows customer agents, to retrieve the available service time slots and their properties' information from a web service in the scheduling system.
- **Submit bid:** Provides customer agents with the ability to submit bids to the scheduling system on behalf of customers.
- **Update bid:** Allows customer agents to update the parameters of a previously submitted bid based on the received result.
- **Receive result of winner determination:** Makes it possible for customer agents to receive the results of the winner determination problem in each iteration.
- **Get final result of schedule:** Allows customers to get the final schedule results from their customer agents when the auction terminates.

7.1.2 Service Provider Agent

The following is the list of functional requirements for a service provider agent.

- **Edit available services:** The service provider can add or remove a service. He/she can also add or remove a time slot for each service, as well as change the properties of individual service time slots, such as reservation price or capacity.

- **Start auction:** Utilized to initiate the auction; the service provider agent submits all the information about available services, available time slots for each service, and their properties to the scheduling system.

7.1.3 Scheduling System

The following is the list of the functional requirements for a scheduling system.

- **Receive and store available service information:** The scheduling system needs to receive and store the updated information about the available service time slots and their properties, acquired from the service provider agent.
- **Receive bids:** The scheduling system must be able to receive bids from customer agents after they submit them. After bids have been received, the scheduling system screens out invalid bids.
- **Compute provisional allocation:** The scheduling system needs to compute a new provisional allocation in each round until the auction is terminated.
- **Update OPL-WDM input:** Upon receiving the customer agents' bids, the scheduling system needs to transform the received data into an OPL- Winner Determination Model (WDM) input. OPL-WDM input will be used by ILOG to solve the winner determination problem.
- **Invoke CPLEX engine:** Upon creating the OPL data source, the scheduling system invokes and then sends the data source to the CPLEX engine. CPLEX engine determines the winning bids during iterations.
- **Terminate auction**: This functionality provides the scheduling system with the ability to check the termination condition, and, when the condition is satisfied, to terminate the auction.
- **Send provisional and final allocation results:** The scheduling system needs to send the result of the provisional allocation in each round and the final allocation results at the end of auction to the customer agents.

7.2 User Case Diagrams

Figures 7-1, 7-2, and 7-3 present the designed use case diagrams that address the presented required functionalities. Due to similarities between the requirements and the use cases, we only describe some of the more important use cases.

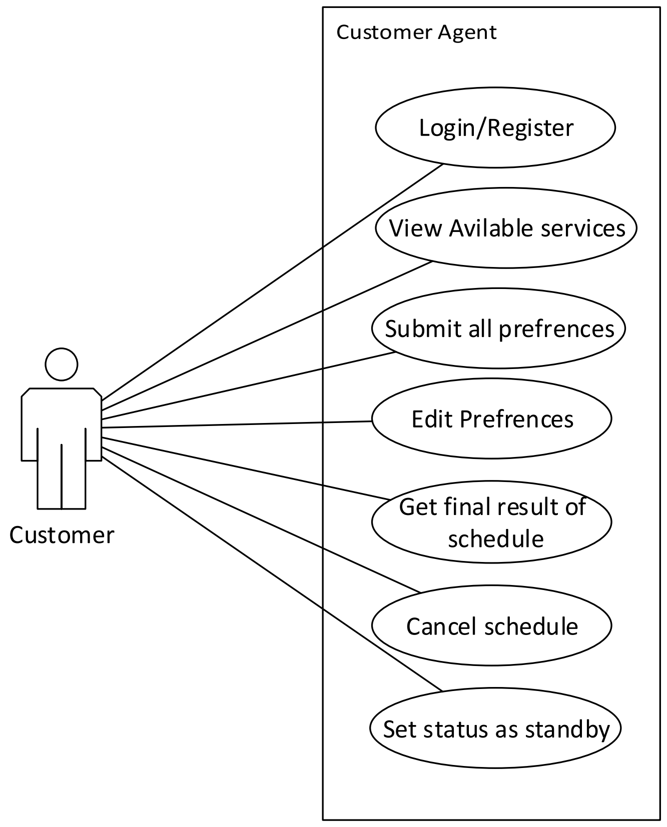


Figure 7-1 Use case diagram for a customer agent

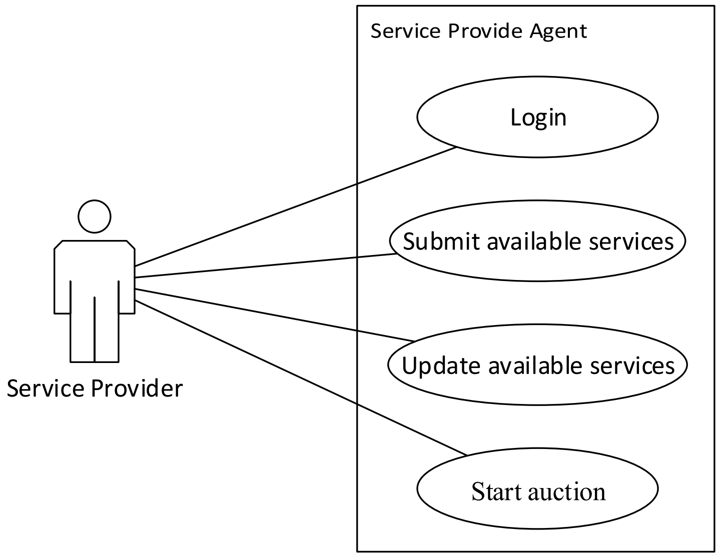


Figure 7-2 Use Case diagram for a service provider agent

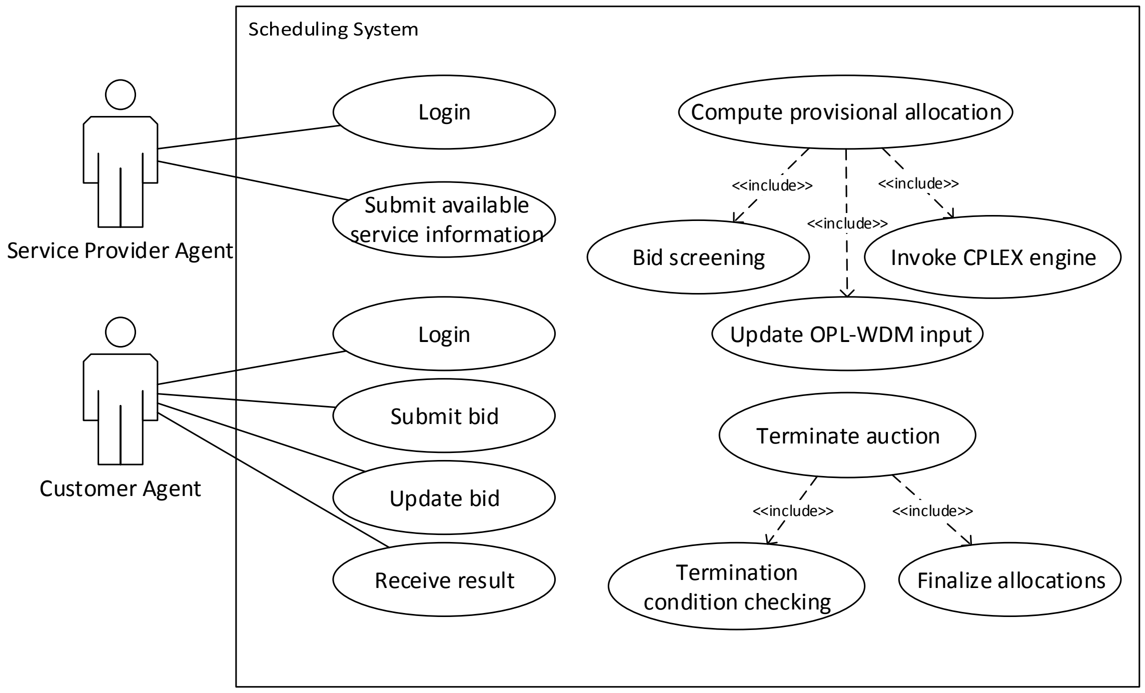


Figure 7-3 Use Case diagram for a scheduling system

Use case: Edit customer's preferences

Brief Description

Customers follow these steps to edit their set of feasible bundles.

Actors: Customer

Preconditions:

1. The customer has logged in to the customer agent.
2. The feasible bundle(s) to be edited has not yet been used as a bid by the time of submitting the bundle.

Main flow of events:

1. Customer agent shows all previously submitted bundles.
2. Customer adds, removes or edits any of the feasible bundle.

Use case: Get the final schedule results

Brief Description

Customers need this process to get the results of the final schedule.

Actors: Customer

Preconditions:

1. The customer has logged in to the customer agent.
2. The auction has terminated and the final schedule has been computed.

Main flow of events:

1. The scheduling system notifies the customer agent about the final schedule result.
2. The customer agent notifies the customer with the result of schedule.

Use case: Update bids

Brief Description

This use case is utilized by a customer agent to update the parameters of previously-submitted bids.

Actors: Customer agent

Preconditions:

1. The customer agent has received the result of the previous round's provisional allocation.

Main flow of events:

1. The customer agent receives the result of the provisional allocation of the previous round. If the latest bid has been accepted, the customer agent will keep the bidding price unchanged; otherwise, it has three price updating options: increase the bidding price, keep the bidding price unchanged, and withdraw from bidding process.
2. The customer agent selects the bundle with the highest payoff
3. The customer agent submit its bid.

Use case: Compute provisional allocation

Brief Description

The scheduling system utilizes this use case to compute a new provisional allocation based on the updated bids received from the agents.

Actors: The scheduling system

Preconditions:

1. The scheduling system has received updated bids from the customer agents.

Main flow of events:

1. After receiving the bids from the customer agents, the scheduling system screens out any invalid bids.
2. The scheduling system verifies the termination condition.
3. If the termination condition has not been satisfied, the system generates the OPL_WDM input.

4. CPLEX engine will be invoked by sending it the OPL_WDM input.
5. CPLEX engine solves the winner determination problem.
6. The winner determination result is transformed into a format that can be sent to the customer agents.
4. The scheduling system notifies the customer agents of the scheduling result.

7.3 Non-Functional requirements

Reliability: The system is a prototype built for research purposes; its reliability is not a major concern. However, the system should be able to function correctly with pre-tested problem sets and scenarios.

Scalability: The system should be scalable in terms of increasing the number of customers. It should be capable of dealing with service scheduling problems at realistic scales.

7.4 System Architecture

This section provides the overall system architecture and software architecture of the service scheduling system.

7.4.1 The overall system architecture

We use a distributed environment as the context for the design of this service scheduling system. As shown in Figure 7-4, in this architecture, the Customer agent functions as the personal assistant of a customer, keeping the customer updated about their preferred service time slot bundles and informing its customer(s) about the results of their requests. Scheduling system in this architecture has eight states. These states can be modeled as a state chart, as shown in Figure 7-5. After Initialization, the scheduling system will be in the state of receiving bids. Each new time period can trigger the transition from the receiving bids state to the bid screening state, where invalid bids are screened out. Each round, the scheduling system must verify if the termination condition has been satisfied.

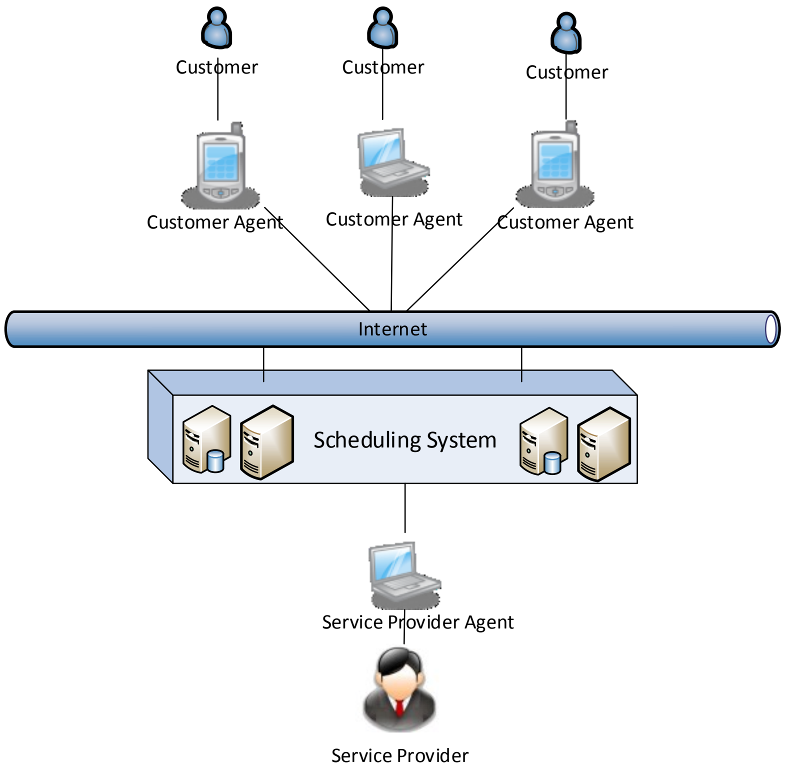


Figure 7-4 The service scheduling system's overall system architecture

If the termination condition is not satisfied, the scheduling system updates the OPL-winner determination model input in the update OPL-WDM input state. At the computing state, the system computes the new winner, incorporating the updated list of bids. During this computing state, the system blocks its bidder interface so they cannot submit new bids. Once a new winner determination has been computed, the system state changes to the announcing results state, in which the system announces the result of the new provisional allocation to the customer agents.

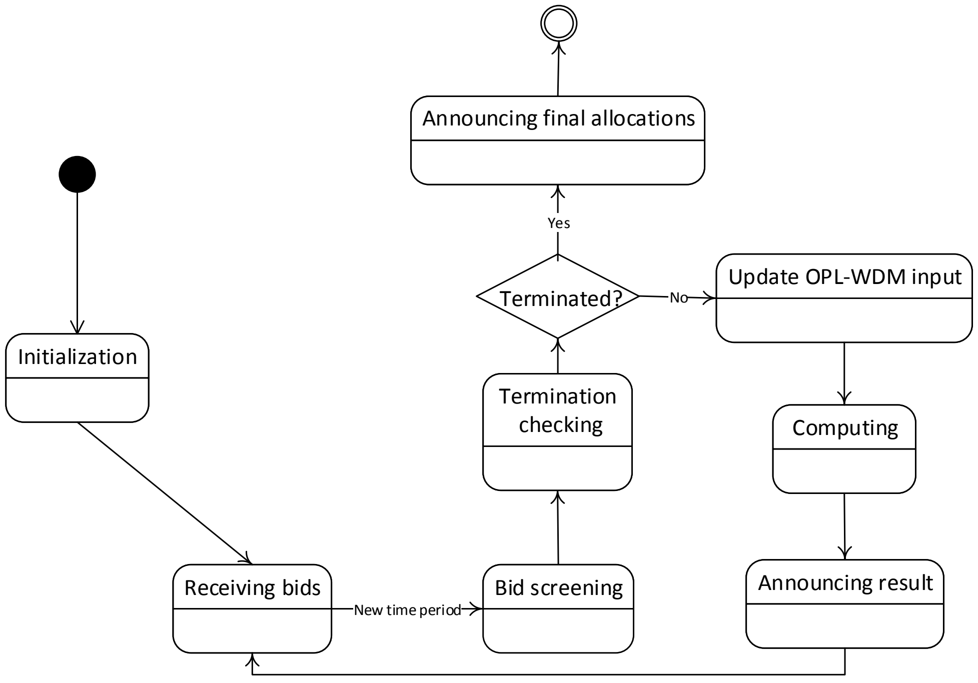


Figure 7-5 State diagram of the scheduling system

7.4.2 Software architecture

In this section we elaborate the software architecture of the service scheduling system.

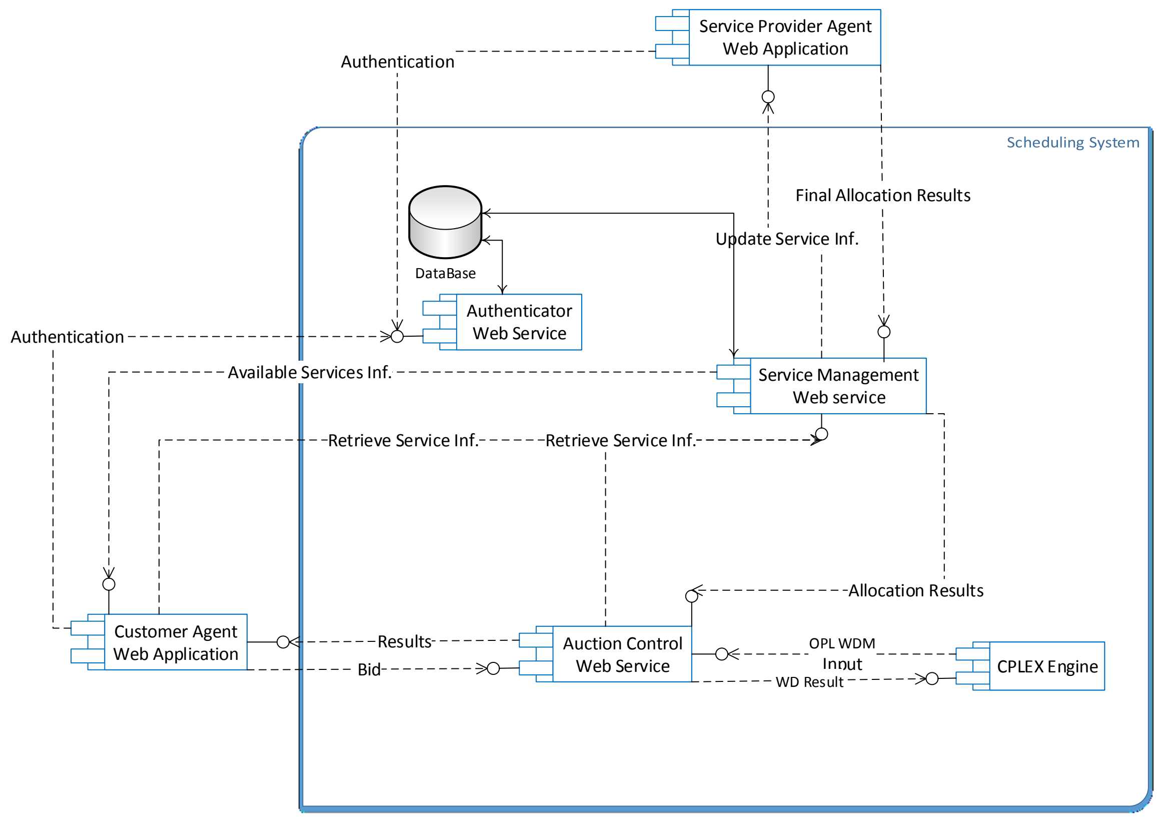
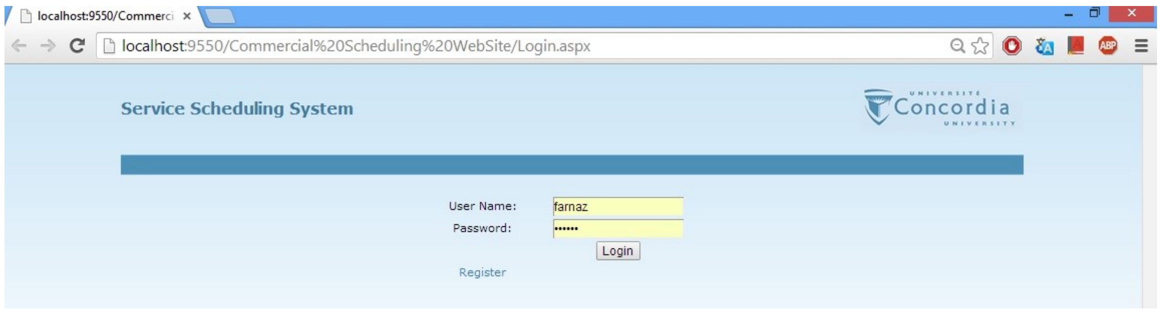


Figure 7-6 Software architecture of the service scheduling system

Figure 7-6 shows how the software components of the system interact with each other. As can be seen from the diagram, the Scheduling System consists of four major components: Authenticator web service, Service Management web service, Auction Control web service, and CPLEX Engine. Authenticator web service is responsible for registering new customers and authenticating registered customers when they login to the scheduling system. The Auction Control component provides two interfaces for interaction with customer agents and the Service Management component. The Service Management web service component is responsible for updating the list of available services. It also provide a web service interface for customer agents so that they can invoke the web service and view the available services and their properties. The Auction Control component also provides a web service interface for customer agents to receive their bids. The CPLEX engine component computes the winners of each round; to accomplish this task, CPLEX requires the updated list of bids from the customer agents. The Auction Control component provides an interface for a winner determination component to receive the updated list of bids in the OPL- WDM input format.

7.4.3 Graphic user interfaces

The prototype has two types of user interfaces: customer agent and provider agent. The provider agent interface implements the functionalities of the service provider agent. The customer agent interface implements the functionalities of customer agents and displays the scheduling results. For the demonstration, when the system is initialized, the login interface (Figure 7-7) is presented to the customer. If the customer is a new customer, registration is required (Figure 7-8).


localhost:9550/Commerci
localhost:9550/Commercial%20Scheduling%20WebSite/Login.aspx
Service Scheduling System
Concordia University
Login
Register

User Name: farnaz
Password: ••••••

Figure 7-7 Customer login interface

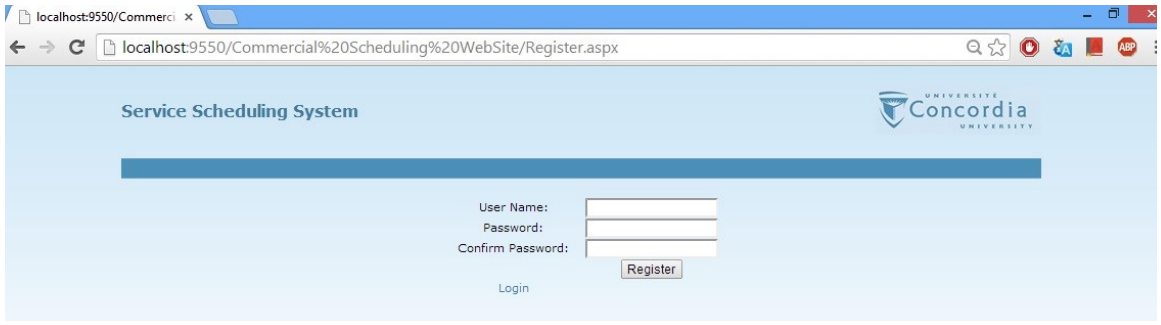
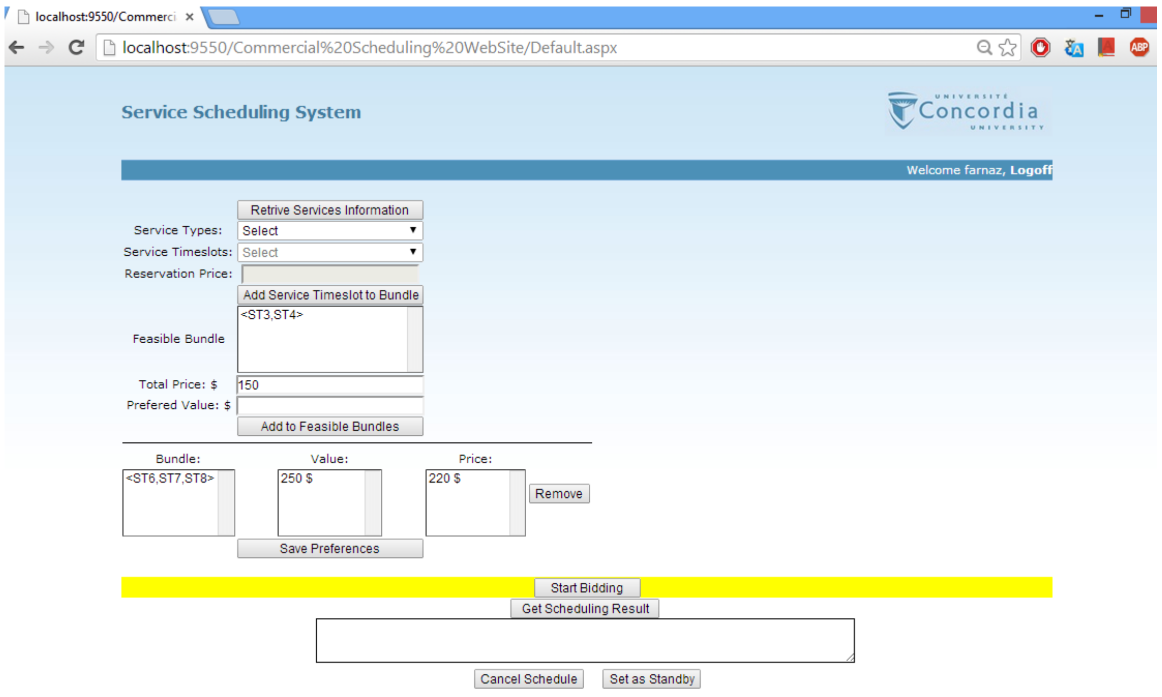


Figure 7-8 Customer registration interface

After login has been completed, the customer agent user interface (Figure 7-9) will be presented to the customer.



Service Types:	Select
Service Timeslots:	Select
Reservation Price:	
Feasible Bundle	<ST3,ST4>
Total Price: $	150
Prefered Value: $	

Bundle:	Value:	Price:
<ST6,ST7,ST8>	250 $	220 $

Figure 7-9 Customer agent interface

Customers can view the list of available service time slots by clicking on the Available Services Information button. Service types and available time slots for each service are retrieved from the Service Management web service and loaded to the dropdown lists. After loading the current available service time slots, customers can select their preferred services and their preferred time slots and add them to their Feasible Bundles. The customer agent starts its negotiation with the Auction control web service when the customer clicks the start bidding button. Once the bids are received, at each specific time period the Auction Control component will generate a new WD-OPLM input and invoke the CPLEX engine to solve the winner determination model, and the result of the winner determination will be announced to the customer agents. Upon receiving the results, the customer agents should make their decisions about updating their bids. This procedure repeats until the Auction terminates. Customers can be informed about the final schedule by clicking the Get Result button.

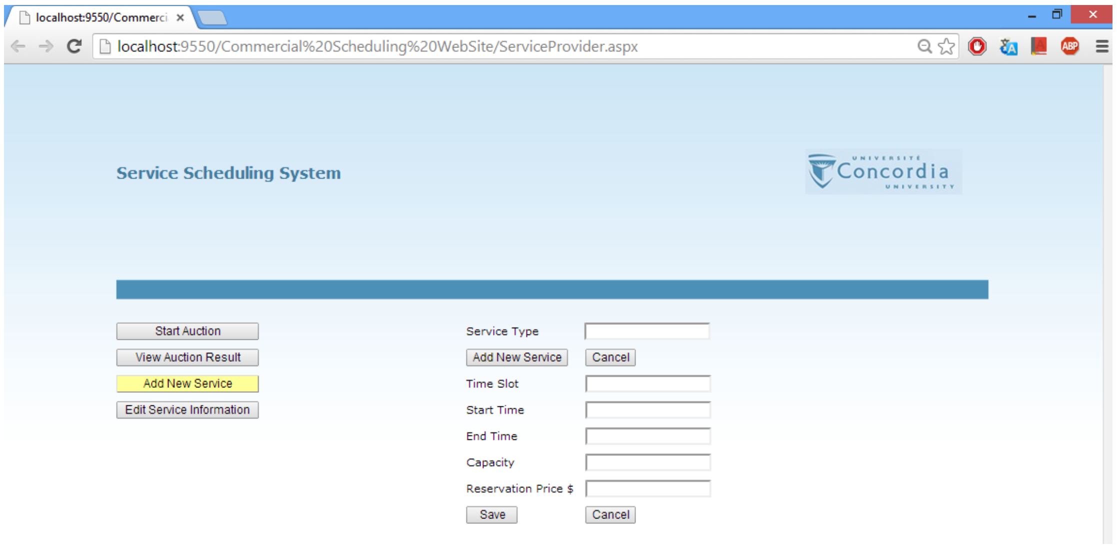


Figure 7-10 Provider agent user interface for adding a new service

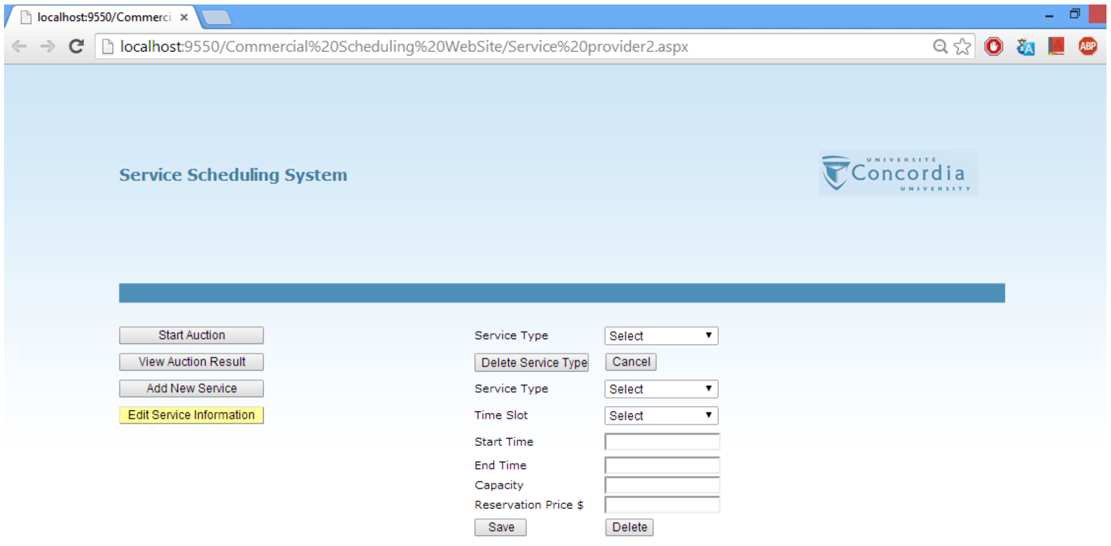
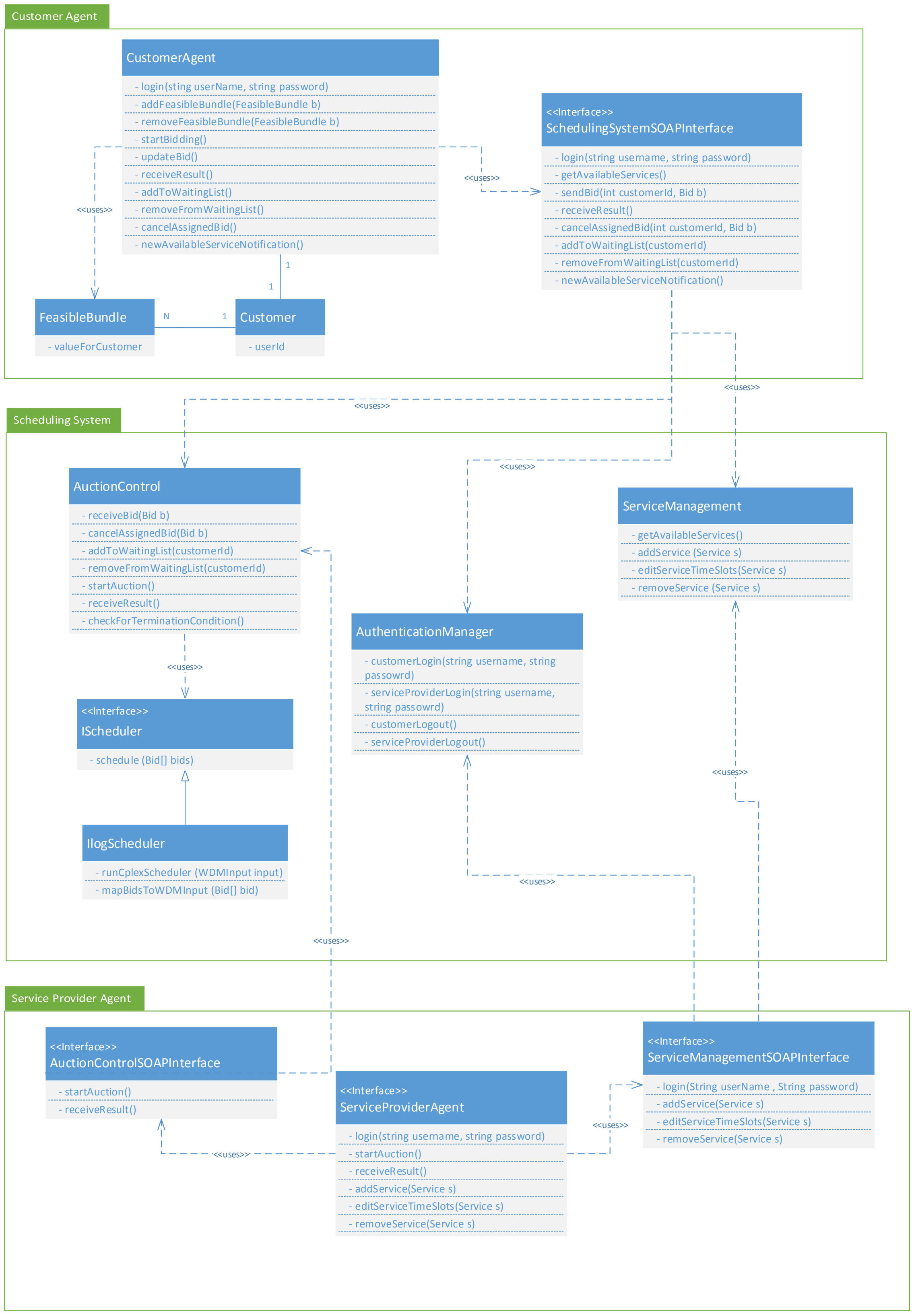


Figure 7-11 Provider agent user interface for editing service information

Service providers can submit their available service time slots and their properties to the service management web service by using the service provider agent interface (Figure 7-10). Service providers can also edit or remove a service by using the editing interface shown above (Figure 7-11).

7.4.4 Class Diagram


Customer Agent
CustomerAgent
- login(sting userName, string password)
- addFeasibleBundle(FeasibleBundle b)
- removeFeasibleBundle(FeasibleBundle b)
- startBidding()
- updateBid()
- receiveResult()
- addToWaitingList()
- removeFromWaitingList()
- cancelAssignedBid()
- newAvailableServiceNotification()
<<Interface>>
SchedulingSystemSOAPInterface
- login(string username, string password)
- getAvailableServices()
- sendBid(int customerId, Bid b)
- receiveResult()
- cancelAssignedBid(int customerId, Bid b)
- addToWaitingList(customerId)
- removeFromWaitingList(customerId)
- newAvailableServiceNotification()
FeasibleBundle
- valueForCustomer
Customer
- userId
N
1
1
1
<<uses>>
Scheduling System
AuctionControl
- receiveBid(Bid b)
- cancelAssignedBid(Bid b)
- addToWaitingList(customerId)
- removeFromWaitingList(customerId)
- startAuction()
- receiveResult()
- checkForTerminationCondition()
<<Interface>>
IScheduler
- schedule (Bid[] bids)
IlogScheduler
- runCplexScheduler (WDMInput input)
- mapBidsToWDMInput (Bid[] bid)
AuthenticationManager
- customerLogin(string username, string passowrd)
- serviceProviderLogin(string username, string passowrd)
- customerLogout()
- serviceProviderLogout()
ServiceManagement
- getAvailableServices()
- addService (Service s)
- editServiceTimeSlots(Service s)
- removeService (Service s)
Service Provider Agent
<<Interface>>
AuctionControlSOAPInterface
- startAuction()
- receiveResult()
<<Interface>>
ServiceProviderAgent
- login(string username, string password)
- startAuction()
- receiveResult()
- addService(Service s)
- editServiceTimeSlots(Service s)
- removeService(Service s)
<<Interface>>
ServiceManagementSOAPInterface
- login(String userName , String password)
- addService(Service s)
- editServiceTimeSlots(Service s)
- removeService(Service s)

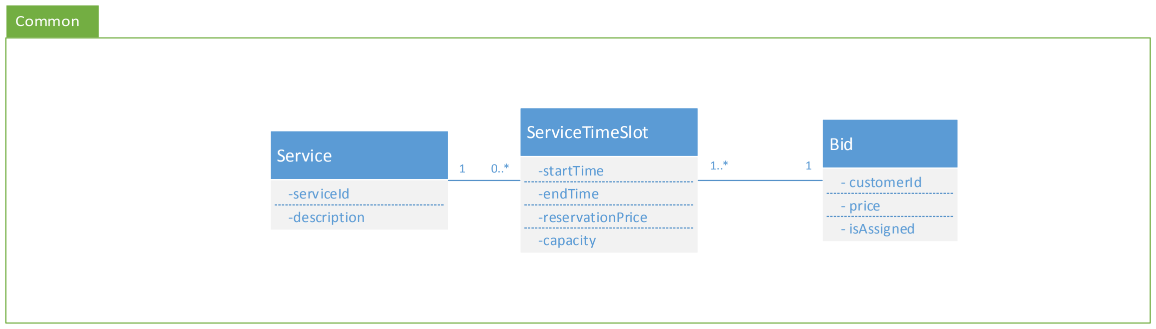


Figure 7-12 Class diagram for service scheduling system

Figure 7-12 shows the class diagram for the prototype system. The prototype system is divided into four packages: customer agent, scheduling system, common and service provider. Each package represents a specific part of the system.

7.4.5 Sequence Diagram

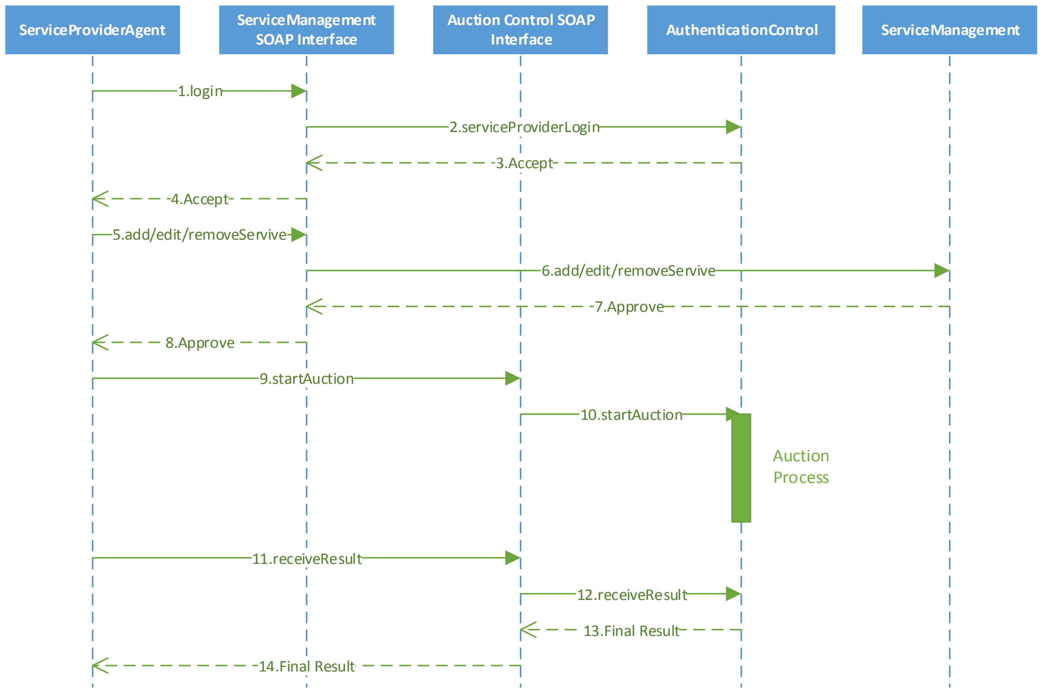


Figure 7-13 Sequence diagram for service provider functionalities

Figure 7-13 shows the message passing sequence for the service provider agent functionalities.

a) The service provider agent logs into the scheduling system (Steps1-4).
b) The service provider agent adds to and/or edits the available services and their properties and adds them to the scheduling system (Steps 5-8).
c) The service provider starts the auction (Steps 9-10).
d) The service provider agent requests the result of the final allocations (Steps 11-14).

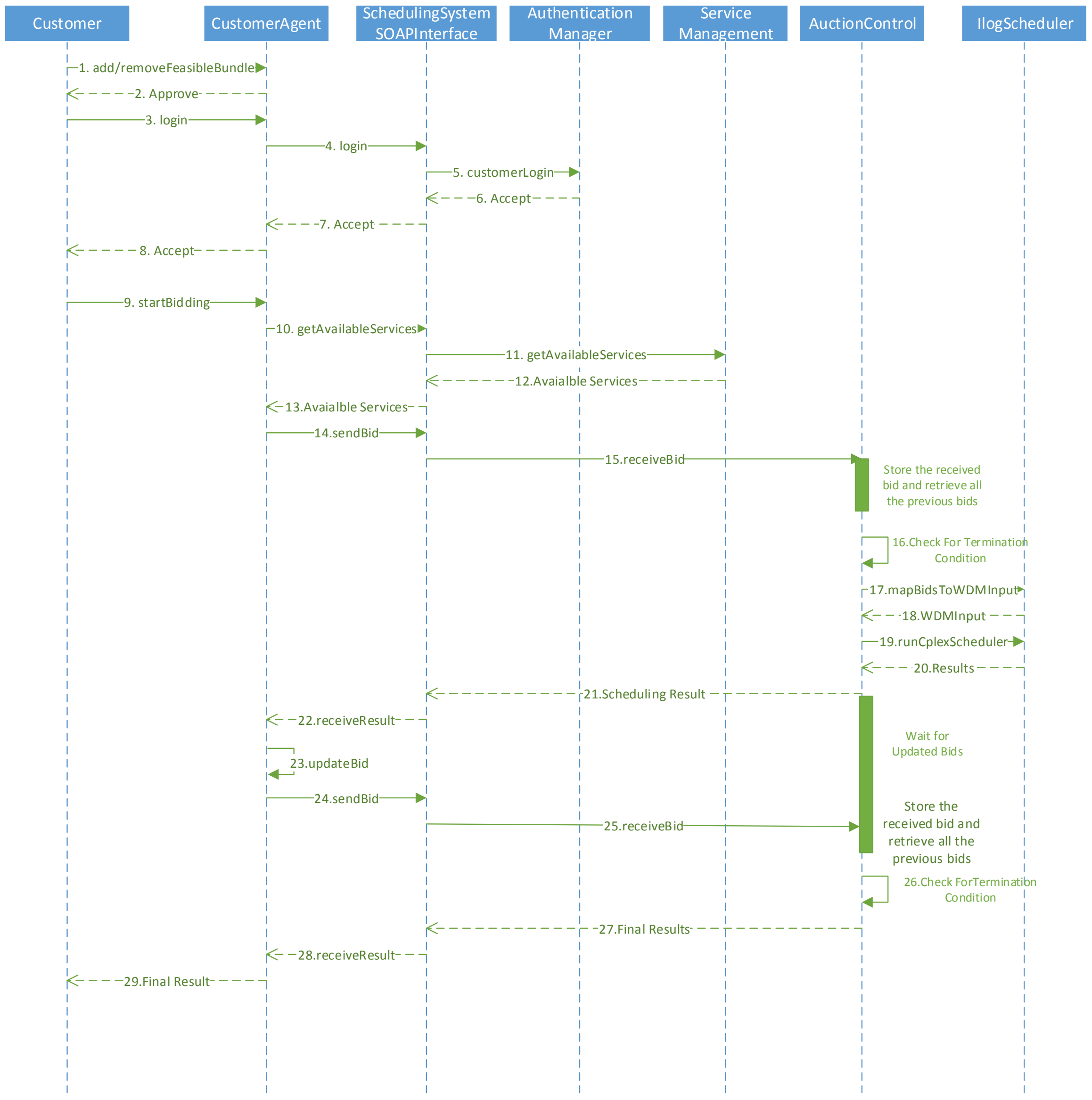


Figure 7-14 Sequence diagram for bidding process

Figure 7-14 shows the messages as they flow during the service scheduling system's bidding process.

a) The customer submits his/her feasible bundles to the customer agent (Steps1- 2).

b) The customer agent logs into the scheduling system (Steps 3-8).

c) The customer agent retrieves the available service information (Steps10-13).

d) The customer agent submits its bid to the SOPA Service Provider Interface (Step 14).

e) The Auction control receives the bids and updates its current bid list (Step 15).

f) Updated bids are sent to ILOG scheduler Interface (Steps 16-17).

g) CPLEX Solver computes the solution and returns it to the SOPA Service Provider Interface (Steps 20-21).

h) The SOPA Service Provider Interface sends the scheduling result to the customer agents (Step 22).

Figure 7-15 shows the messages for dynamic change management in service scheduling system as a sequence diagram.

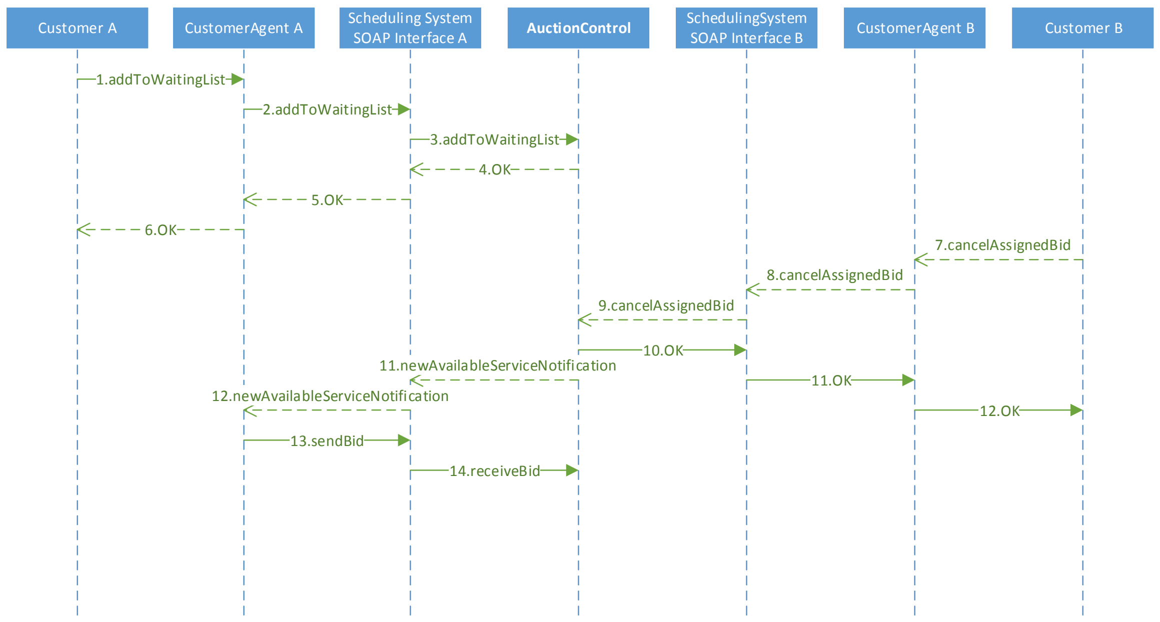


Figure 7-15 Sequence diagram for dynamic change management

7.5 Simulation Result

In this section we have used the developed prototype to evaluate our approach in terms of its scalability. The simulation platform is based on a client-server architecture. The auctioneer web services are hosted in a web server and the agents use SOAP to access those web services. The auctioneer web server runs on a desktop PC with 2.4GHz Intel CPU and 8 GB memory. All the agents are equally distributed over two laptop systems, each of which has an Intel 2GHz CPU and 4 GB of memory. Every agent generates a random set of feasible bundles and their values. When an auction starts, each agent will use the generated feasible bundles to bid on. The agent web application and the auctioneer web services are implemented using a Visual Studio 2013. Module to generate random problem sets, coded in Visual Studio 2013.

7.5.1 Metrics

Communication Cost and Response Time, are used as the scalability measures in the evaluation.

Communication Cost is the number and the size of messages sent between agents and the auctioneer during an auction, expressed as:

$$Load\ t = \sum_{g\, \in N^t} \left(\left|Req_g\right| + \left|Resp_g\right|\right)$$

$$Communication\ Cost = \sum\nolimits_{t\, \in Total\ \#\ of\ Iterations} Load\ t$$

where $load\ t$ is the load of communication in round t, N^t is the set of customers submitted their bids at round t, $\left|Req_g\right|$ is the size of requested message by agent , and $\left|Resp_g\right|$ is the size of respond message from auctioneer to agent g.

Response Time , is considered to be a measure of the computational complexity of the proposed approach. The Response Time (in seconds) is the sum of the time required for the winner-determinations, price-updates and communication over all if the auction rounds; in other words, the total time it takes for all agents to receive the final schedule from the moment an auction starts.

To evaluate the scalability, ten groups of problem instances are randomly generated with different sizes and structures. The configuration of the test problem sets and their corresponding response times, communication costs and memory usage are summarized in Table 7-1 and Table 7-2.

Table 7-1 Configuration of testing problems

Problem		Number of agents	# of service time slots	# of feasible bundles per agent	Number of Instance
#	Name				
1	Group 1	200	30	Random(5,10)	10
2	Group 2	400	50	Random(5,10)	10
3	Group 3	600	70	Random(6,11)	10
4	Group 4	800	90	Random(6,11)	10
5	Group 5	1000	100	Random(6,15)	10
6	Group 6	1200	100	Random(6,15)	10
7	Group 7	1400	110	Random(6,15)	10
8	Group 8	1600	120	Random(7,20)	10
9	Group 9	1800	125	Random(7,20)	10
10	Group 10	2000	130	Random(8,25)	10

Table 7-2 Computational results

Groups	Response Time(S)	Communication Cost (KB)	Memory Usage (MB)
Group 1	284.1725	64.99	482.31
Group 2	363.4935	277.36	701.91
Group 3	494.61	290.33	937.97
Group 4	530.189	336.47	1074.42
Group 5	728.514	376.97	1780.03
Group 6	782.2695	389.86	1866.46
Group 7	906.3825	495.19	1929.51

Group 8	1141.7535	570.33	2486.09
Group 9	1220.89	603.002	2732.97
Group 10	1513.5115	684.37	2944.76

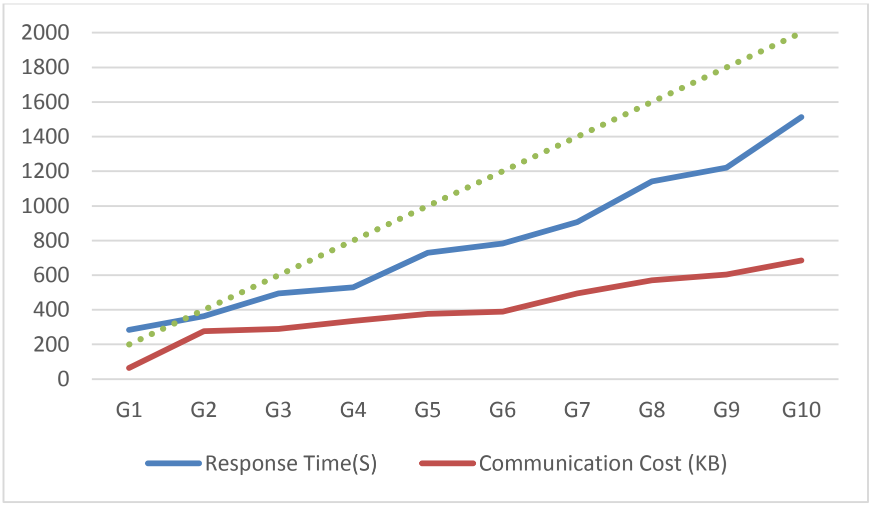


Figure 7-16 Response time and communication cost when the problem complexity increases

It can be seen from Figure 7-16 that the response time curve is sub-linear on the value axis as the problem complexity increases, indicating polynomial run times, and so it is clear that the I*b*SCHF can be applied to large-scale DSSPs.

7.6 Summary

This chapter presents the design and implementation of a prototype environment for the I*b*SCHF proposed in chapter 4. The first step was defining the requirements, followed by a detailed illustration of the system architecture and of how the software components communicate with each other. The developed prototype was evaluated in terms of its communication costs and system responsiveness.

Chapter 8 Conclusion and Future Work

8.1 Conclusion

This thesis investigates modeling and computational issues in developing solution approaches to the Distributed Service Scheduling Problem (DSSP). Compared to traditional manufacturing scheduling, service scheduling poses additional challenges due to the significant customer involvement in service processes. The first challenge is that service scheduling should be conducted in a distributed environment. The second challenge is that service scheduling has to be robust at accommodating contingencies caused by the customer involvement in service production. Uncertainty in customer demand, customer cancelations and no-shows make the service scheduling a complex dynamic process. The third challenge is the customer's private information. To compute optimal schedules, ideally, the scheduler should know the complete customer availability and preference information within the scheduling horizon. However, customers often act strategically to protect their private information. Therefore, service scheduling systems should be designed so that they are able to elicit the customer's private information required to compute high quality schedules. The fourth challenge is that the objectives in a service scheduling environment are complicated and may conflict each other. The distributed service scheduling environment enables each agent to have their own scheduling objectives. In addition to multiple objectives, since agents are self-interested, they are expected to behave strategically to achieve their own objectives without considering the global objectives of the system.

Our objective is to design a framework capable of addressing the challenges in the DSSP. An iterative bidding scheduling framework (I*b*SCHF) that can address the challenges of DSSPs concurrently is proposed. I*b*SCHF uses the price mechanism for service scheduling, which may not be applicable for non-commercial services. An adapted I*b*SCHF is also proposed for scheduling non-commercial services without using a price system or payment transfers. Our main contributions can be summarized as follows.

I*b*SCHF for DSSPs

By applying I*b*SCHF in an agent-based architecture, the challenges of DSSPs can be addressed effectively. I*b*SCHF provides a structure for the agents and the service provider to interact in a systematic way and eventually evolve the provisional solutions towards an optimal one. In I*b*SCHF, agents are not required to reveal all their private information; they reveal their private information only as it becomes necessary. It also has the potential of accommodating dynamic changes. I*b*SCHF can be applied to efficiently allocate the newly-available service time slots created by dynamic events.

The framework has been evaluated experimentally. The results indicate that, compared with the one-shot VCG auction system, the I*b*SCHF requires less information revelation, improves on the computational properties, and its computed solutions are very close to optimal. As a demonstration of the applicability of the framework, it was applied to the Service Customization under Capacity Constraints (SCCC) problem. By applying the I*b*SCHF to the SCCC problem, customer's customization decision making are integrated with the allocation of the service provider's capacity through multilateral negotiations between the service provider and its customers.

Adapted I*b*SCHF for non-commercial service scheduling problems

The I*b*SCHF uses price mechanism to allocate service time slots to customers. However, in non-commercial service scheduling environments, service providers cannot use a price mechanism to schedule customers along the service timelines. Therefore, I*b*SCHF has been adapted for non-commercial service scheduling problems. The service provider needs customers' availability information to improve resource utilization. On the other hand, customers may be of "two minds" about communicating their private information. While communicating certain amount of availability might be necessary in order to obtain their preferred schedules, too much communication implies a potential cost. To address this challenge, an adapted I*b*SCHF has been developed, designed to generate high-quality schedules while protecting customers' private information. The efficiency and information revelation performance of this adapted framework is evaluated through theoretical analysis and computational experiments. It was shown that, under the proposed mechanism, myopic bidding is the dominant strategy for customers. The privacy and efficiency performance of this proposed adapted mechanism was also evaluated, through a computational study. As a demonstration of the realistic applicability of this framework, it was applied to the appointment scheduling problem in health care system.

Design and implementation of a web-based service scheduling prototype

A web-based service scheduling prototype is designed and implemented using .Net technology and web services. The purpose of developing the prototype is first, to demonstrate how to implement the I*b*SCHF in a real-world environment, and second, to evaluate the scalability of the approach in a real environment. Scalability is measured in terms of response time, communication cost, and memory usage.

8.2 Directions for Future Research

Three directions can be outlined in terms of expanding the current work from the perspective of improving its applicability to real-world scale applications.

First, to continue improving the I*b*SCHF to accommodate more and more dynamic changes. The current framework supports customer's cancelations and uncertainty in customers' arrival time. Other reasons for dynamic changes could be incorporated, for example, service durations may be subject to change, and/or certain resources can become unavailable.

Second, the winner determination model could be extended to different application domains. The current winner determination model is a general model that can be applied to different service application domains. Each application domain has its own constraints that will need to be considered.

At the current stage, I have considered the situation where customer agents have a service request from one service provider. However, in some application domains, agents may be prefer to receive different services from multiple service providers. In the situation where each service provider uses the I*b*SCHF, exploring agents' bidding policies in order to coordinate separate scheduling system, would be another step towards practical approaches to real world distributed service scheduling applications.

References

Abdennadher, S., & Schlenker, H. (1999). INTERDIP-an interactive constraint based nurse scheduler. *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence, Menlo Park, CA,* 838-843

Aickelin, U., & Dowsland, K. A. (2001). Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling 3(3)*, 139-153

An, B., Lesser, V., Irwin, D., & Zink, M. (2010). Automated negotiation with decommitment for dynamic resource allocation in cloud computing. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 981–988

Anderson, A., Tenhunen, M., & Ygge, F. (2000). Integer programming for combinatorial auction winner determination. *In Proceedings of the Fourth Internal Conference on Multi-Agent Systems (ICMAS) Boston, MA: IEEE computer society,* 39–46

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M. (2009) , Above the Clouds: A Berkeley View of cloud computing, *University of California at Berkeley*

Azaiez, M. N., & Sharif, S. (2005). A 0-1 goal programming model for nurses cheduling. *Computers and Operations Research 32(3)*, 491-507

Babayan, A. and He, D., (2004). Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach. *International Journal of Production Research, 42 (4)*, 777–799

Bailey, R. N., Garner, K. M., & Hobbs, M. F. (1997). Using Simulated Annealing and Genetic Algorithms to Solve Staff Scheduling Problems. *Asia-Pacific Journal of Operational Research 14(2)*, 27-43

Bannock, G., Baxter, R. E., & Reese, R. (1982). *The Penguin Dictionary of Economics*. Penguin Books, Ltd., Harmondsworth, Middlesex England

Barron, W. M. (1980). Failed appointments Who misses them, why they are missed, and what can be done Primary Care *7(4)* 563–574

Becker, M., & Hans, C. (2006). Artificial Software Agents as Representatives of Their Human Principals in Operating-Room-Team-Forming. *Multi-agnet Engineering International Handbooks on Information Systems,* 221-237

Berger, S., & Bierwirth, C. (2010). Solutions to the request reassignment problem in collaborative carrier networks. *Transportation research Part E,Volume 46,No.5*, 627-638

Bikhchandani, S. and Ostroy, J. M.(2006). Ascending price Vickrey auctions, *Games and Economic Behavior, vol. 55, No. 2*, 215-241

Bowman, E. H. (1959).The Schedule Sequencing Problem, *Operations Research*, *vol.7, No.5*, 621-624

Brucker, P. (2004). *Scheduling Algorithms* (4th ed). Springer, Berlin

Burke, E. K., Elliman, D. G., & Weare, R. F. (1995). A hybrid genetic algorithm for highly constrained timetabling problems. *Proceedings of the 6th International Conference on Genetic Algorithms,* Pittsburgh, USA,Morgan Kaufmann, Los Altos, CA, 605-610

Burke, E. K., Newall, J. P., & Weare, R. F. (1996). A memetic algorithm for University exam timetabling. *Burke and Ross,* 241-250

Burke, E. K., Newall, J. P., & Weare, R. F. (1998). Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation 6 (1)*, 81-103 (special issue on Scheduling)

Burke, P. and Prosser, P., (1991). A distributed asynchronous system for predictive and reactive scheduling. *International Journal for Artificial Intelligence in Engineering, 6 (3),* 106–124

Caridi, M. and Cavalieri, S., (2004). Multi-agent systems in production planning and control: an overview. *Production Planning and Control, 15 (2)*, 106–118

Chiussi, F.M. and Francini, A., (2000). A distributed scheduling architecture for scalable packet switches. *IEEE Journal on Selected Areas in Communication, 18 (12)*, 2665–2683

Chung, D., Chankwon P., Sukho K., Jinwoo P., (1996) Developing a shop floor scheduling and control software for an FMS. *Computers and Industrial Engineering, 30 (3)*, 557–568

Church, L. K., & Uzsoy, R. (1992). Analysis of periodic and eventdriven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing, 5(3),* 153–163

Clarke, E. H. (1971), Multipart pricing of public goods, *Public Choice, 11(1),*17-33

Crawford, E., & Veloso, M. (2004). Mechanism Design for Multi-Agent Meeting Scheduling Including Time Preferences, Availability, and Value of Presence. *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*

Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics, 72(1)*, 1–13

Dargahi,F., Wang, C., Bhuiyan, M. F. H., and Mehrizi, H. (2012). Agent-Based Design for Service Process Scheduling: Challenges, Approaches and Opportunities. *Transactions of the SDPS:Journal of Integrated Design and Process Science*

Davidsson, P., Henesey, L., Ramstedt, L., T¨ornquist, J., & Wernstedt, F. (2005). An analysis of agent-based approaches to transport logistics. *Transportation Research, Part C, 13,* 255–271

Davis, L. (1985). Job shop scheduling with genetic algorithms. *Proc. 1st int. Conf. on Genetic algorithms and their Applications, Pittsburgh, PA,* 130-140

Demeester, P., Souffriau, W., De Causmaecker, P., & Vanden Berghe, G. (2010). A hybrid tabu search algorithm for automatically assigning patients to beds. *Artif. Intell. Med. 48(1)*, 61–70

Dowsland, K. (1998). Nurse scheduling with tabu search andstrategic oscillation. *European Journal of Operational Research 106 (2–3),* 393–407

Du, X., Jiao, J., & Tseng, M. M. (2001). Architecture of product family: Fundamentals and methodology. *Concurrent Engineering: Research and Application, 9(4)*, 309–325

Duffie, N.A. and Prabhu, V.V., (1994). Real-time distributed scheduling of heterarchical manufacturing systems. *Journal of Manufacturing Systems*, 13 (2), 94–107.

Fischer, K., Müller J. P. and Pischel, M. (1995). Cooperative transportation scheduling: an application domain for DAI. *Journal of Applied Artificial Intelligence*

Franzin, M. S., Freuder, E. C., & Rossi, F. (2002). Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. *American Association for Artificial Intelligence AAAI*

Frayret, J. M. (2009). A Multidisciplinary Review of Collaborative Supply Chain Planning. *Conference Proceedings IEEE International Conference on Systems, Man and Cybernetics ,San Antonio, TX*, 4414–4421

Fujishima, Y., Leyton-Brown, K., & Shoham, Y. (1999). Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. *In 16th International joint conference on artificial intelligence (IJCAI)*, 548–553

Gagliano, R. A., Fraser, M. D., & Schaefer, M. E. (1995). Auction allocation of computing resources. *Communications of the ACM, 38 (6)*, 88–102

Garg, S., and Buyya, R. (2011). Market-Oriented Resource Management and Scheduling: A Taxonomy and Survey, *Cooperative Networking 277-306, M. S. Obaidat and S. Misra (eds),* ISBN: 978-0-470-74915-9, Wiley Press, New York, USA

Ghaemi, M.,Vakili, M., & Aghagolzadeh, A. (2007). Using a genetic algorithm optimizer tool to solve university timetable scheduling problem. *9th international symposium on signal processing and its Application*

Ghenniwa, H., 1996. *Coordination in Cooperative Distributed Systems. Ph.D.* dissertation, University of Waterloo

Giret, A., and Botti, V., (2004). Holons and agents. *Journal of Intelligent Manufacturing*, 15 (5), 645–659.

Gomber, P., Schmidt, C., Weinhardt, C. (1997). Elektronische Märkte für die dezentrale Transportplanung, *Wirschaftsinformatik 39(2),*137-145

Grano, M., Medeiros, D. J., & Eitel, D. (2009). Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Manage Science, Volume 12,*228-242

Groothuis, S., & Merode, G. (2001). Simulation as decision tool for capacity planning. *Journal of Computer Methods and Programs in Biomedicine 66 ,* 139–151

Groves, T. (1973), Incentives in Teams, *Econometrica, 41(4):*617-631

Gueret, C., Jussien, N., Boizumault, P., & Prins, C. (1995). Building University Timetables Using Constraint Logic Programming. *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling,* 393- 408

Gujo, O., Schwind, M., & Vykoukal, J. (2009). A combinatorial intra-enterprise exchange for logistics services. *Information systems and e-business management,Volume 7,No 4,*447-471

Gunawan, A., Ming, K., & Poh, K. (2007). Solving the teacher assignment-course scheduling problem by hybrid Algorithm. *International journal of Computer, information and system science and engineering, 1(2),*139-141

Gupta, D., B. Denton. 2008. Appointment scheduling in health care: Challenges and opportunities. *IIE Trans. 40:* 800–819

Hadavi, K., Hsu, L., Chen, T., Lee, C.N. (1992). An architecture for real time distributed scheduling. In: Famili A, Nau DS, Kim SH (eds) *Artificial intelligence applications in manufacturing*. Cambridge USA: AAAI Press, 215–234

Hancock, W.M., & Walter, P. F. (1984). The use of admissions simulation to stabilize ancillary workloads. *Simulation journals*, 88-94

Hannebauer, M., & Muller, S. (2001). Distributed Constraint Optimization for Medical Appointment Scheduling. *Proceedings of the fifth international conference on autonomous agents,*139 -140

Harvey, J. (1998). Service quality: A tutotial. *Journal of Operations Management 16(5)*, 583-597

Hassine, A. B., Defago, X., & Ho, T. B. (2004). Agent-Based Approach to Dynamic Meeting Scheduling Problems. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems,Volume 3,* 1132 -1139

Healthcare Commission (2005) Primary care trusts: survey of patients. *London: Commission for Healthcare Audit and Inspection*

Henz, M., & Wurtz, J. (1995). Using Oz for college timetabling. *Proceedings of the 1st Int. Conference on the Practice and Theory of Automated Timetabling,* 283- 296

Hertz, A. (1991). Tabu Search for Large Scale Timetabling Problems. *European Journal of Operational Research 54,* 39-47

Hertz, A. (1992). Finding a Feasible Course Schedule Using Tabu Search. *Discrete Applied Mathematics 35(3),* 255-270

Ho, Ch., & Lau, H. (1999). Evaluating the impact of operating conditions on the performance of appointment scheduling rules in service systems. *European Journal of Operational Research 112 ,*542-553

Hohlt, B., Doherty, L., and Brewer, E., (2004). Flexible power scheduling for sensor networks. In *Proceedings of the IEEE and ACM International Symposium on Information Processing in Sensor Networks*, 205–214

Hohner, G., Rich, J., Ng, E., Reid, G., Davenport, A. J.,&Kalagnanam, J. R., et al. (2003). Combinatorial and quantity-discount procurement auctions benefit Mars, incorporated and its suppliers. *Interfaces, 33(1),* 23–35

Hosseini, H., Hoey, J., & Cohen, R. (2011). Multi-Agent Patient Scheduling Through Auctioned Decentralized MDPs. *Proceedings of the 6th InformsWorkshop on Data Mining and Health Informatics*

Hur, D., Mabert, V. A., & Bretthauer, K. M.(2004). Real-time work schedule adjustment decisions: An investigation and evaluation. *Production and Operations Management 13(4)*, 322

IntelliQuest, (1990), Conjoint Analysis: A Guide for Designing and Integrating Conjoint Studies, *Marketing Research Technique Series Studies, American Marketing Association, Market Research Division, TX*

Jack, E. P., & Powers, T. L. (2004). Volume flexible strategies in health services: A research framework. *Production and Operations Management 13(3),* 230

Jaumard, B., Semet, F., & Vovor, T. (1998). A generalized linear programming model for nurse scheduling. *European Journal of Operational Research 107(1),*1-18

Jiao, J., & Tseng, M. M. (1999). A methodology of developing product family architecture for mass customization. *Journal of Intelligent Manufacturing, 10(1)*, 3–20

Jiao, J., Simpson, T., & Siddique, Z. (2007). Product family design and platform-based product development: A state-of-the-art review. *Journal of Intelligent Manufacturing, 18(1)*, 5–29

Keil, J. M. (1992), On the complexity of scheduling tasks with discrete starting times, *Operations Research Letters, vol. 12,* 293-295

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *American Association for the Advancement of Science New Series, Vol. 220, No. 4598.,* 671-680

Kotler, P., & Keller, K. (2006). *Marketing management, Twelfth edition.* Prentice-Hall, Upper Saddle River, New Jersey

Kouiss, K., Pierreval, H., and Mebarki, N., 1997. Using multi-agent architecture in FMS for dynamic scheduling. Journal of Intelligent Manufacturing, 8 (1), 41–47

Krajewska, M. A., & Kopfer, H. (2006). Collaborating freight forwarding enterprises, request allocation and profit sharing. *OR spectrum, Volume 28, No2,* 301-317

Krishna, A., & Ünver, M. U. (2007). Improving the Efficiency of Course Bidding at Business Schools: An Experimental Study. *Marketing Science, forthcoming*

Kutanoglu, E., Wu, S. D.(1999). On combinatorial auction and Lagrangean relaxation for distributed resource scheduling, *IIE Trans. vol. 31*, 813-826

Kwon, R. H., Lee, C., & Ma, Z. (2005). An integrated combinatorial auction mechanism for truckload transportation procurement. *Technical Report, Mechanical and Industrial Engineering, the University of Toronto, Ontario, Canada*

Lang, N., Moonen, H. M., Srour, F. J., & Zuidwijk, R. A. (2008). Multi Agent Systems in Logistics: A Literature and State-of-the art Review. *ERIM Report Series,* Reference No. ERS-2008-043-LIS

Lau, J.S.K., Huang, G.Q, Mak, K.L., Liang, L. (2005a). Distributed project scheduling with information sharing in supply chains: part I– an agent-based negotiation model. *International Journal of Production Research*, 22 (15), 4813–4838.

Lau, J.S.K., Huang, G.Q, Mak, K.L., Liang, L. (2005b). Distributed project scheduling with information sharing in supply chains: part II- theoretical analysis and computational study. *International Journal of Production Research*, 23 (1), 4899–4927

Lima, R.M., Sousa, R.M., and Martins, P.J., (2006). Distributed production planning and control agent-based system. *International Journal of Production Research, 44 (18–19),* 3693–3709

Liu, J. and Sycara, K.P., (1993). Distributed constraint satisfaction through constraint partition and coordinated reaction. *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, May, Hidden Valley, PA

Liu, J. and Sycara, K.P., (1994). Distributed problem solving through coordination in a society of agents. *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, July, Seattle, WA.

Liu, J. and Sycara, K.P., (1995). Exploiting problem structure for distributed constraint optimization. *Proceedings of the First International Conference on Multiagent Systems*, June, San Francisco, California.

MacKie-Mason, J. K., Osepayshvili, A., Reeves, D. M., and Wellman, M. P.,(2004). Price prediction strategies for market-based scheduling, *Fourteenth International Conference on Automated Planning and Scheduling*, Whistler, BC, 244-252

Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). Microeconomics. Oxford: Oxford University Press

Maturana, F.P., and Norrie, D.H. (1996). Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing*, 7, 257–270

Meisels, A., & Kaplansky, E. (2003). Scheduling Agents – Distributed Timetabling Problems. *Lecture Notes in Computer Science, Practice and Theory of automated timetabling IV*,*Volume 2740/2003*, 166-177

Meyer, M. H., & Utterback, J. M. (1993). The product family and the dynamics of core capability. *Sloan Management Review, 34(3)*, 29–47

Milgrom, P. R., & Weber, R. J. (1982). A theory of auctions and competitive bidding. *Econometrica: Journal of the Econometric Society, 50*, 1089–1122.

Modi, P.,Veloso, M., Smith, S. F., & Oh, J. (2004). CMRadar: A Personal Assistant Agent for Calendar Management. *Lecture Notes in Computer Science, LNCS 3508*,169–181

Muhlemann, A. P., Lockett, G.,& Farn, C. K. (1982). Job shop scheduling heuristics and frequency of scheduling. *International Journal of Production Research, 20(2),* 227–241

National Survey of Local health Services (2006). The quality of patient engagement and involvement in primary care. *Picker Institute*

Nisan, N.(2006), Bidding languages for combinatorial auctions in Combinatorial Auctions, *Cramton, Shoham, and Steinberg, Eds.: MIT Press*

O'Hare, C. D., & Corlett, J. (2004). The outcomes of open-access scheduling. *Family Practice Management, 11(2)*, 35–38

Ovacik, I. M., & Uzsoy, R. (1994). Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequencedependent set-up times. *International Journal of Production Research, 32(6),* 1243–1263

Paechter, B., Cumming, A., & Luchian, H., (1995). The use of local search suggestion lists for improving the solution of timetabling problems with evolutionary algorithms. *Proceedings of the AISB Workshop on Evolutionary Computing, Sheffield, England*

Paechter, B., Cumming, A., Norman, M. G., & Luchian, H. (1996). Extensions to a memetic timetabling system. *The Practice and Theory of Automated Timetabling, volume 1153 of Lecture Notes in Computer Science. Springer Verlag,* 251–265

Parkes, D. C. (2001). *Iterative combinatorial auctions: Achieving economic and computational effciency, Ph.D.* dissertation, Department of Computer and Information Science, University of Pennsylvania

Parkes, D. C. (2006). Iterative combinatorial auctions. *In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, Combinatorial Auctions. MIT Press*

Parkes, D. C. and Ungar, L.(2000). Iterative Combinatorial Auctions: Theory and Practice, *In Proceedings 17th National Conference on Artificial Intelligence*, Austin, TX, 74-81

Parkes, D. C., and Kalagnanam, J.(2005). Models for Iterative Multi-attribute Procurement Auctions, *Management Science*, vol.51, no.3, 435-451

Parunak, H.V.D., (1987). Manufacturing experience with the contract Net, In: M.N. Huhns, ed. *Distributed Artificial Intelligence*. Vol. I, Los Altos, CA: Morgan Kaufmann, 285–310

Paulussen, T. O., Jennings, N. R., Decker, K. S., & Heinzl, A. (2003). Distributed patient scheduling in Hospital. *Coordination and Agent Technology in Value Networks, GITO*

Pearce, D. W. (1981). *The dictionary of modern economics.* The MIT Press, Cambridge, Massachusetts

Petrovic, D., Morshed, M., & Petrovic, S. (2011). Multi-objective genetic algorithms for scheduling of radiotherapy treatments for categorized cancer patients. *Journal of Expert Systems with Applications,38(6), 6994-7002*

Pinedo, M.L. (2008). *Scheduling Theory, Algorithms, and Systems* (3th ed).Springer ISBN: 978-0-387-78934-7

Pinedo, M.L. (2009). *Planning and scheduling in manufacturing and services* (2nd ed.). Springer, New York. doi: 10.1007/978-1-4419-0910-7

Rahimifard, S. and Newman, S.T., (1998). Reference architectures for team based distributed production planning and control. *Eureka–Factory: Project No 1629*

Rubin, G., Bate, A., George, A., Shackley, P. and Hall, N. (2006) Preferences for access to the GP: a discrete choice experiment. *British Journal of General Practice*

Sabuncuoglu, I., & Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems, 18(4),* 268 283.

Sampson, S. E. & Froehle, C. M. (2006). Foundations and implications of a proposed unified services theory. *Production and Operations Management,* 329-343

Sampson, S. E. (2001). *Understanding service businesses: Applying principles of the unified services theory (2nd ed.)* New York: Wiley

Sandholm, T. (1999). Distributed Rational Decision Making, In Multiagent Systems: *A Modern Introduction to Distributed Artificial Intelligence, G. Weiss, Ed., MIT Press*, 201-258, 19

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence, 135(1–2)*,1–54

Sandholm, T., Suri, S., Gilpin, A., & Levine, D. (2005). CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, *51(3),* 374–390

Schönberger, J. (2005). *Operational Freight Carrieer Planning*. Springer, Berlin

Schönsleben P., Hieber R. (2004). Gestaltung von effizienten Wertschöpfungspartnerschaften im Supply Chain Management. *Busch A., Dangelmaier W., Integriertes Supply Chain Management, Wiesbaden.*

Sen, S., (1997). Multiagent systems: milestones and new horizons. *Trends in Cognitive Sciences*, *1 (9)*, 334–340

Shaw, M.J. and Whinston, A.B., (1988). A distributed knowledge-based approach to flexible automation: the Contract Net framework. *International Journal of Flexible Manufacturing Systems*, *1 (1)*, 85–104

Sheffi, Y. (2004). Combinatorial Auctions in the Procurement of Transportation services. *Interfaces,Volume.34 ,* 245-252

Shen, W., (2001). Agent-based cooperative manufacturing scheduling: an overview. *COVE Newsletter*, Available online at: http://www.uninova.pt/~cove/newsletter.htm/2/Shen.pdf

Shen, W., (2002). Distributed manufacturing scheduling using intelligent agents. *IEEE Intelligent Systems*, *17 (1)*, 88–94

Shen, W., Wang, L., & Hao, Q. (2006). Agent-based distributed manufacturing process planning and scheduling : a state-of-the-art survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 36(4),* 563-577

Sim, K. M. (2012). Complex and Concurrent Negotiations for Multiple Interrelated e-Markets. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, PP(99),* DOI: 10.1109/TSMCB.2012.2204742, 1-16

Simpson, T. W. (2004). Product platform design and customization: Status and promise. *AIEDAM, 18(1)*, 3–20

Simpson, T. W., Maier, J. R. A., & Mistree, F. (2001). Product platform design: Method and application. *Research in Engineering Design, 13(1)*, 2–22

Singh, A., & Malhotra, M. (2012). Agent Based Framework for Scalability in Cloud Computing. *International Journal of Computer Science & Engineering Technology (IJCSET), 3(4),* 41-45

Smith, R.G., (1980). The Contract Net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers, C-29 (12),* 1040–1113

Smith, R.G. and Davis, R., (1981). Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-11 (1),* 61–70

Song, J., & Regan, A. C. (2003). An Auction Based Collaborative Carrier Network.*Technical report: UCI-ITS-WP-03-6, Institute of Transportation Studies, University of California, Irvine*

Sönmez, T., & Ünver, U. (2007). *Course Bidding at Business Schools.* Retrieved from http://ssrn.com/abstract=1079525 2007

Sycara, K. P., Roth, S. F., Sadeh, N., and Fox, M. S., (1991), Resource allocation in distributed factory scheduling. *IEEE Expert*, 29- 40

Tharumarajah, A., (2001). Survey of resource allocation methods for distributed manufacturing systems. *Production Planning and Control, 12 (1)*, 58–68.

Trentesaux, D., Tahon, C., and Ladet, P., (1998), Hybrid production control approach for JIT scheduling. *Artificial Intelligence in Engineering*, *12*, 49-67

Tsay, A., Nahmias, S., and Agrawal, N., (2000). Modeling supply chain contracts: a review, In: S. Tayur, R. Ganeshan and M. Magazine, eds. *Quantitative Models For Supply Chain Management*. Norwell, MA: Kluwer Academic Publishers, 299–330

Tseng, M. M., & Du, X. (1998). Design by Customers for Mass Customization Products. *CIRP Annals - Manufacturing Technology, 47(1)*, 103-106

Tseng, M. M., & Jiao, J. (1996). Design for mass customization. *Annals of the CIRP, 45(1)*, 153–156

Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy, 24(3)*, 419–440.

Vaidya, N., Dugar, A., Gupta, S., and Bahl, P., (2005). Distributed fair scheduling in a wireless LAN. *IEEE Transactions on Mobile Computing*, *4 (6),* 616–629

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance, 16(1),* 8-37

Vieira, G. E., Herrmann, J. W., & Lin, E. (2000). Analytical models to predict the performance of a single machine system under periodic and event-driven rescheduling strategies. *International Journal of Production Research, 38(8)*, 1899–1915.

Vries , S. and Vohra, R. V.,(2003). Combinatorial auctions: a survey, *INFORMS Journal on Computing, vol. 15,* 284-309

Wainer, J., Ferreira, P., & Constantino, E. R. (2007). Scheduling meetings through multi-agent negotiations. *Decision Support Systems 44,* 285–297

Wang, C., Dargahi, F., Bhuiyan, M. F. H. (2012). On the Tradeoff between Privacy and Efficiency: A Bidding Mechanism for Scheduling Non-Commercial Services. *Computers in Industry*, DOI:10.1016/j.compind.2012.01.012

Wang, C., Dargahi, F. (2012). Service Customization under Capacity Constraints: An Auction-Based Model. *Journal of Intelligent Manufacturing*, DOI: 10.1007/s10845-012-0689-7

Wang, C., Ghenniwa, H. H., & Shen, W. (2009). Constraint-based winner determination for auction-based scheduling. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* 39(3), 609–618.

Wang, W., & Gupta, D. (2011). Adaptive Appointment Systems with Patient Preferences. *Manufacturing and Service Operations Management 13(3),* 373-389

Wang,C.(2007). *Economic Models for Decentralized Scheduling , Ph.D.* dissertation. The University of Western Ontario, Canada

Wang, C., Ghenniwa, H., and Shen, W., (2009). Constraint-based winner determination for auction based scheduling. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39 (3), 609–618

Wellman, M. P., Walsh, E., Wurman, P. R. and MacKie-Mason, J. K., (2001). Auction Protocols for Decentralized Scheduling, *Games and Economic Behavior, vol.35, No.1-2*, 271-303

Wellman, M. P., MacKie-Mason, J.K., Reeves, D.M., and Swaminathan, S., (2003). Exploring bidding strategies for market-based scheduling, *Proceedings of the 4th ACM conference on Electronic commerce*, San Diego, CA, USA,115 – 124

Wemmerlov, U. (1990). A taxonomy for service processes and its implications for system design. *International Journal of Service Industry Management 1(3)*, 13–27

Wolski, R., Plank, J. S., Brevik, J., & Bryan, T. (2001). Analyzing market-based resource allocation strategies for the computational grid. *International Journal of High Performance Computing Applications, 15 (3),* 258-281

Yamamoto, M., & Nof, S. Y. (1985). Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research, 23(4),* 705–722.

Yang, E.H., Barash, M.M., and Upton, D.M., (1993). Accommodation of priority parts in a distributed computer-controlled manufacturing system with aggregate bidding schemes. *Proceedings of the 2nd Industrial Engineering Research Conference Proceedings*, 827–831

Zaman, S., & Grosu, D. (2011). Combinatorial Auction-Based Dynamic VM Provisioning and Allocation in Clouds. *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom),*107-114

Zhiming, Z. (2011). A Two-stage Scheduling Approach of Operation Rooms Considering Uncertain Operation Time. *International Conference on Information Science and Technology,Nanjing, Jiangsu, China 250.* DOI: 10.1115/DETC2011-48263