CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Distributed Coordination of Project Schedule Changes: An Agent-Based Compensatory Negotiation Approach

By

Keesoo Kim

CIFE Technical Report #130
December, 2001

STANFORD UNIVERSITY

If you would like to contact the authors, please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

SUMMARY

CIFE TECHNICAL REPORT # 130

Title: Distributed Coordination of Project Schedule Changes: An Agent-Based Compensatory Negotiation Approach

Authors: Keesoo Kim (Advisor: Boyd C. Paulson, Jr.)

Publication Date: December 20, 2001

Funding Source: CIFE Seed Project (10/1/99 to 9/30/00) – Agent-Based Project Scheduling and Control

Abstract:

In the construction industry, projects are becoming increasingly large and complex, involving multiple subcontractors. Traditional centralized coordination techniques used by the general contractors become insufficient as subcontractors perform most work and provide their own resources. When subcontractors cannot provide enough resources, they hinder their own performance as well as that of other subcontractors and ultimately the entire project. Thus, construction projects need a new distributed coordination approach wherein all of the concerned subcontractors can respond to changes and reschedule a project dynamically.

The focus of this research is rescheduling a project in a distributed manner in order to lower the sum of all participating subcontractors' extra costs associated with changes in their resource constraints, subject to the precedence relationships among project activities, without assuming that a central coordinator knows all the information needed for coordination and that subcontractors are benevolent. The challenges are to find a new distributed approach that enables subcontractors to compensate other affected subcontractors for disadvantageous agreements so that it enhances the global outcome while pursuing individual incentives; to identify schedule conflicts, consider alternatives; and resolve schedule conflicts in a tightly coupled network of related activities; and to preserve the work logic and ensure convergence of distributed computation.

To meet the challenges, I developed a new distributed coordination framework for project schedule changes (DCPSC) and a novel agent-based compensatory negotiation (ABCN) methodology to enable the framework. The DCPSC-based ABCN methodology has met challenges fully by using novel definitions of utility-based schedule-change options, by employing new multi-linked negotiation protocols based on a shared project plan, and by introducing new mechanisms of directing message-passing based on the Critical Path Method (CPM).

In addition to this theoretical work, I designed and implemented a new Java-based multi-agent prototype — distributed subcontractor agent system (DSAS) — to demonstrate the effectiveness of the DCPSC framework through a series of comparison tests, charrette tests, and measurements. DSAS solves the problems successfully. Thus, this research formalizes, implements, and tests the necessary steps to help subcontractors coordinate schedule changes in order to increase the efficiency of their resource use, which in turn enhances successful completion of whole projects.

This research describes the research completed by Keesoo Kim at Stanford University's Center for Integrated Facility Engineering (CIFE). This research supports CIFE's goals by encouraging collaboration over the Internet between distributed project teams during construction phases of a project by providing them with a novel agent-based compensatory negotiation approach for distributed coordination of project schedule changes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

My Ph.D. research began with a scenario where one subcontractor in a construction project has a problem. He cannot provide enough resources at the originally estimated cost, and thus is expecting a cost overrun. The underlying assumption is that subcontractors seek profits from their resource utilization. If making profits becomes infeasible, then a subcontractor will take actions to mitigate the impacts (O'Brien and Fischer 2000). One possibility is to re-schedule his work to match his available resources. Therefore, the subcontractor asks the general contractor whether he can do this.

However, the general contractor could respond negatively for three reasons: (1) Coordinating subcontractors' schedule changes is beyond the capability of the general contractor in cases of complex projects, involving many subcontractors; (2) the general contractor cannot access private information held by subcontractors (Choo and Tommelein 2000), which is needed for coordinating project schedules; and (3) there is a contractual relationship between the general contractor and the subcontractor. This means that the general contractor has little incentive to help the subcontractor to re-schedule (Tommelein and Ballard 1997). Therefore, the subcontractor has to negotiate with other subcontractors.

There are many interrelationships and interdependencies in project schedules. Therefore, a schedule change in a subcontractor's work might affect other subcontractors' schedules as well as the project deadline. The subcontractor should resolve schedule conflicts with other affected subcontractors. This implies that there is a need for a methodology[1] for distributed coordination of project schedule changes (DCPSC).

1.1 PRACTICAL MOTIVATION OF DCPSC

Despite the ubiquity of change in large, complex construction projects, current approaches to change coordination are mostly reactive, and therefore lead to less than optimal solutions. If, however, changes in a given schedule were coordinated prior to execution, then better solutions could be found. Previous researchers have explored the various causes of schedule changes in construction projects. Discrepancies between the needed resources for activities and the resources available to subcontractors are one major cause of change (O'Brien et al. 1995). The resource discrepancies occur when the timing of the activities is not well matched with the available resources, i.e., when subcontractors have different perspectives of scheduling. The resource focus in my work is the local resource which subcontractors provide, not the global resources which general contractors provide.

Soon after the general contractor awards subcontracts according to the master project schedule, subcontractors often want to change the master schedule because resource discrepancies cause additional costs either through over-utilizing currently available resources or importing new resources (O’Brien and Fischer 2000). Therefore, the subcontractors may try to change the project schedule in order to accommodate their wishes. Changes are likely to cause schedule conflicts among subcontractors because any move affects the activities of other subcontractors in tightly coupled construction project schedules. A schedule conflict occurs when one subcontractor tries to re-schedule his activity to the same time interval that another subcontractor has already scheduled the succeeding activity. Since all activities have precedence relationships, related activities cannot be overlapped.

[1] In this dissertation, methodology means a set or system of methods, principles, and rules used in a given discipline (Quotation from Webster’s College Dictionary, 2nd ed., Random House Inc., New York, NY, 1997, p. 825)

In most cases, these schedule conflicts cannot easily be resolved simply by delaying the succeeding activities since such delays would affect the resource profiles of succeeding subcontractors, which would cause additional costs for them. Delays could also extend the project completion beyond the deadline. Therefore, there is a need for a methodology to handle subcontractors' resource-driven schedule changes.

In current practice, the coordination of subcontractors' resource-driven schedule changes in complex projects is difficult for general contractors, because general contractors generally will not know details of each subcontractor's resource constraints. Furthermore, general contractors have little incentive to accommodate the subcontractors' wishes. This dilemma in current project schedule coordination stems from a mismatch between traditional centralized coordination techniques in the industry and current construction practices employing more and more subcontracting. Subcontractors are competitive by nature and cannot be coordinated simply by orders from general contractors, a method that has been used traditionally by general contractors when they performed most work themselves.

1.2 ASSUMPTIONS OF DCPSC

Before going further, I would like to state three assumptions made for DCPSC:

- Project schedules have fixed work logic and precedence relationships;
- Subcontractors keep their resource and cost information private; and
- Subcontractors are not benevolent in accommodating changes.

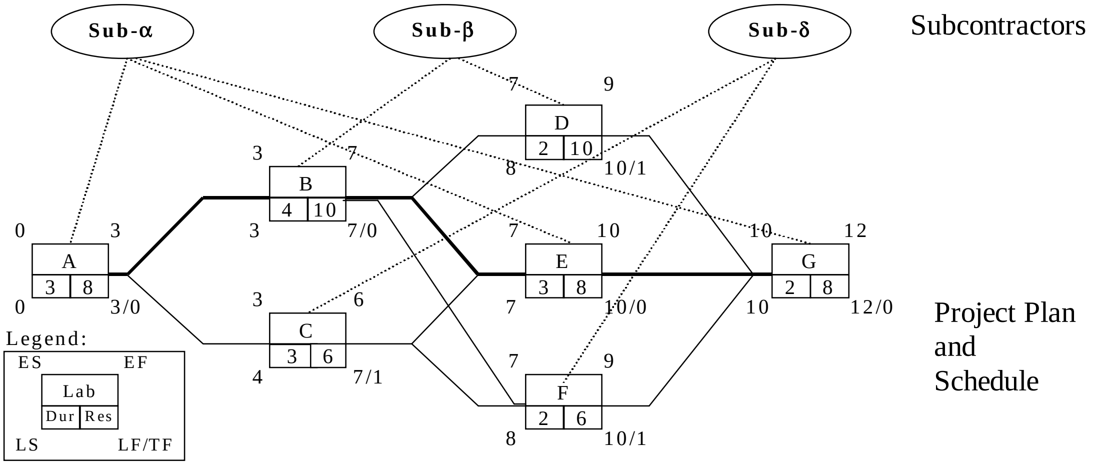
My research addressed the coordination of subcontractors' resource constraints *after* the general contractor has made the master schedule. Therefore, I assume that project schedules have fixed work logic and precedence relations among activities.

I also assume that subcontractors keep their resource information private and want to enhance their resource utilization by rescheduling their activities without violating the work logic. Since their activities are interdependent with other subcontractors' activities, subcontractors have to consult with other subcontractors when rescheduling their activities. In most construction projects schedule conflicts are common and must be negotiated among subcontractors.

Subcontractors work collaboratively to achieve a common project goal, but they are different organizations. Therefore, they cannot be regarded as “benevolent” in accommodating changes, i.e., they are unlikely to help without compensation. Rather, they are competitive, which means they will not let a subcontractor affect their work without compensation. If reasonable monetary compensation is provided, then they are likely to help.

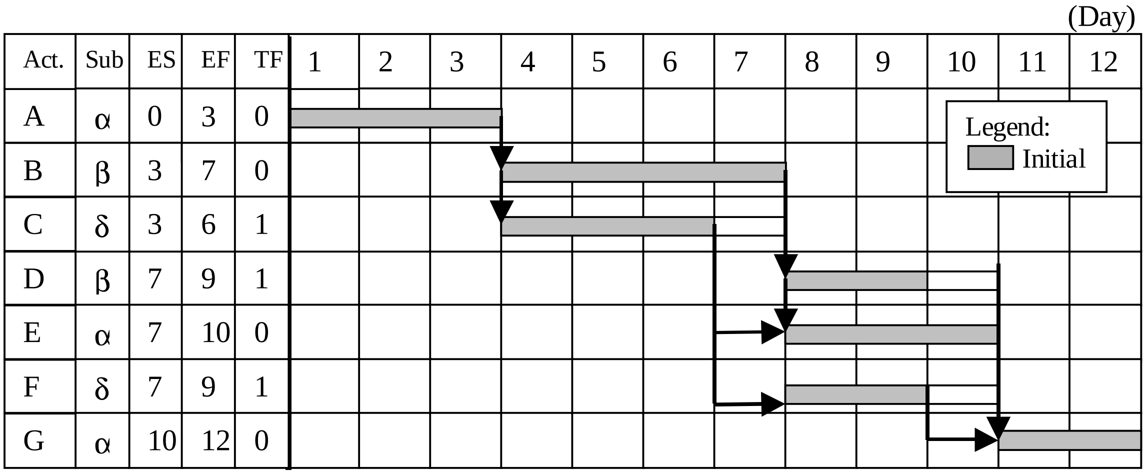
1.3 CASE EXAMPLE

I introduce a case example, which I will use throughout this dissertationto illustrate my research. Consider the example network in Figure 1-1 (a). The results of conventional CPM calculations appear on the diagram. The resource requirement for each activity appears on the diagram in Figure 1-1 (a). Activities (A, E and G) are assigned to Sub-α. Activities (B and D) are assigned to Sub-β. Activities (C and F) are assigned to Sub-δ. For simplicity, assume each subcontractor uses the same resource for its activities.



(a)

Figure 1-1. Example network and ERS schedule



Act.	Sub	ES	EF	TF
A	α	0	3	0
B	β	3	7	0
C	δ	3	6	1
D	β	7	9	1
E	α	7	10	0
F	δ	7	9	1
G	α	10	12	0

(b)

Figure 1-1. Example network and ERS schedule (Continued)

Assume that the subcontractors predicted at the time of bidding that they would have sufficient resources available for the activity to support the initial schedule. The resource histogram in Figure 1-2 indicates the initial resource requirements, based on the above schedule, for completion of the activities. The resource requirements for non-critical activities, which are not as constrained as the critical activities, are set at their latest start-finish (LS-LF) schedules. Therefore, the schedule will be feasible.

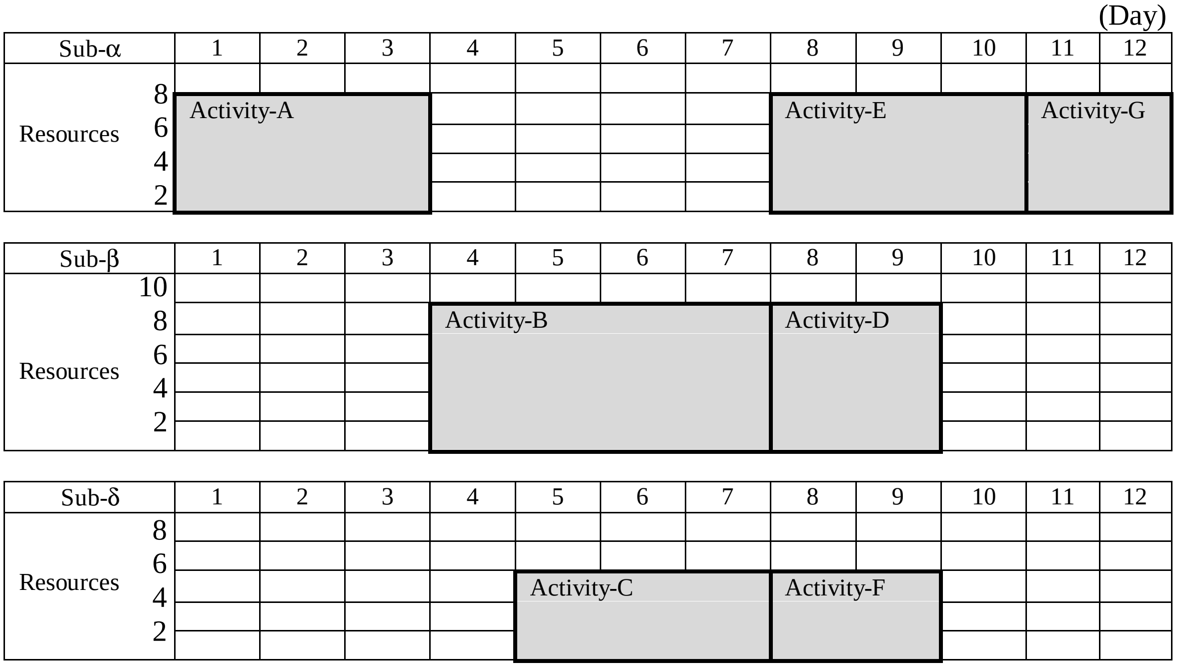


Figure 1-2. Resource requirement histogram

However, as the actual execution dates approach, the resource availability has become tighter under changing market conditions. Various reasons could explain why the resource availability would change: (1) the subcontractors experience a shortage of workers, break-downs of major equipment, shortage of major materials, etc. (Clough and Sears 1991); (2) the subcontractors might want to move the committed resources from the project to more lucrative projects (O'Brien and Fischer 2000). For whatever reasons, assume that each subcontractor now has revised available resource profiles for each activity, as shown in Figure 1-3. Bolded resource histograms (A, B, and D) differ from the required resource diagrams. Note that the resource availability is shown for explanation purposes only, even though such private information is usually kept within subcontractors and usually is not available to the general contractor.

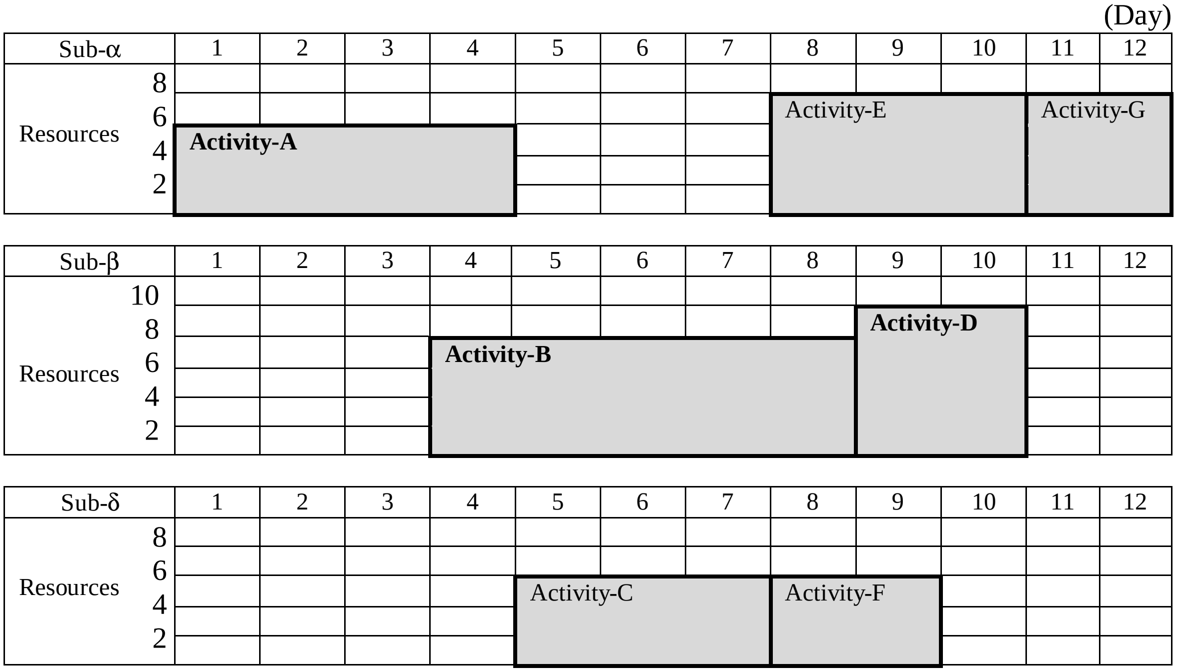
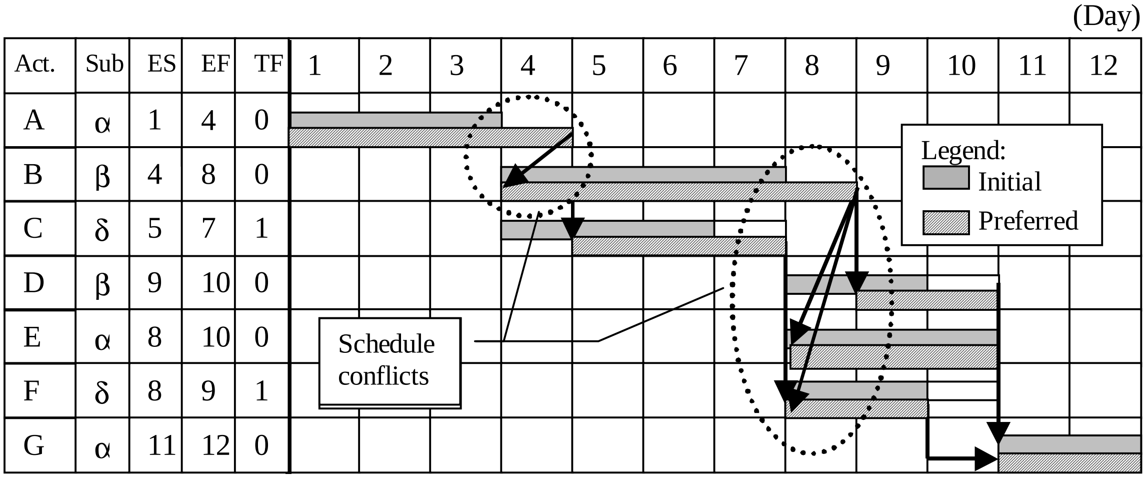


Figure 1-3. Available resource histogram

This resource histogram implies that some subcontractors' schedules differ from their original ones. For instance, Sub-α wants to finish Activity-A on Day 4 since Sub-α does not have enough resources to finish on Day 3. Sub-β wants to finish Activity-B on Day 8 since Sub-β does not have enough resources to finish on Day 7. Based on these revised resource histograms, some of the subcontractors' preferred schedule shifts are shown by the diagonally hatched bars in Figure 1-4.



Act.	Sub	ES	EF	TF
A	α	1	4	0
B	β	4	8	0
C	δ	5	7	1
D	β	9	10	0
E	α	8	10	0
F	δ	8	9	1
G	α	11	12	0

Figure 1-4. Subcontractors' preferred schedule

Note that the mere collection of subcontractors' preferred schedules is not guaranteed to produce a feasible schedule since subcontractors' resource availability is independent, unlike their schedules. Therefore, some activities might violate the network logic in the schedule. For instance, on Day 4, Activity-B starts before Activity-A has finished.

1.4 TWO CURRENT CENTRALIZED COORDINATION METHODOLOGIES

As a basis for evaluating the effectiveness of DCPSC, I introduce two centralized coordination methodologies used in current practice. Under the centralized coordination methodologies below, the general contractor (GC) is obligated to coordinate the subcontractors. It is reasonable to assume that the information needed for coordination, such as the subcontractors' preferred schedules and resource information, is not available to the GC but is kept within subcontractors (Choo et al. 2000). Assume that communication and coordination is kept among subcontractors and the general contractor under the current contractual GC-subcontractor relationships, not between subcontractors. I examine two methodologies of centralized coordination — tight and loose — that I regard as the representation of existing practice.

1.4.1 TIGHT “IRON-FIST” CENTRALIZED COORDINATION

Under tight “Iron-Fist” centralized coordination (TCC), the objective is to finish the project on time and the subcontractors are instructed to finish their activities before the latest finish date of each activity respectively. Under TCC, the GC can coordinate the subcontractors to finish the project on time. However, some subcontractors might experience cost overruns when their available resources differ from their resource requirements. For the GC, the cost overruns could be regarded as subcontractors’ faults for poor management of their resources.

Under TCC, Sub-α has to expedite Activity-A to finish on time in spite of its different resource availability. Assume that Sub-α will choose overtime to accelerate the Activity-A while paying extra costs. Other activities will have the same initial schedule. Sub-β also has to expedite Activity-B to finish on time. Assume that Sub-β will choose overtime to accelerate Activity-B while paying extra costs. Activity-D will keep the same initial schedule. Sub-δ can keep its preferred schedule because Sub-δ can finish its activities before the deadlines. The revised resource histogram following TCC is in Figure 1-5. The diagonal hatching indicates overtime.

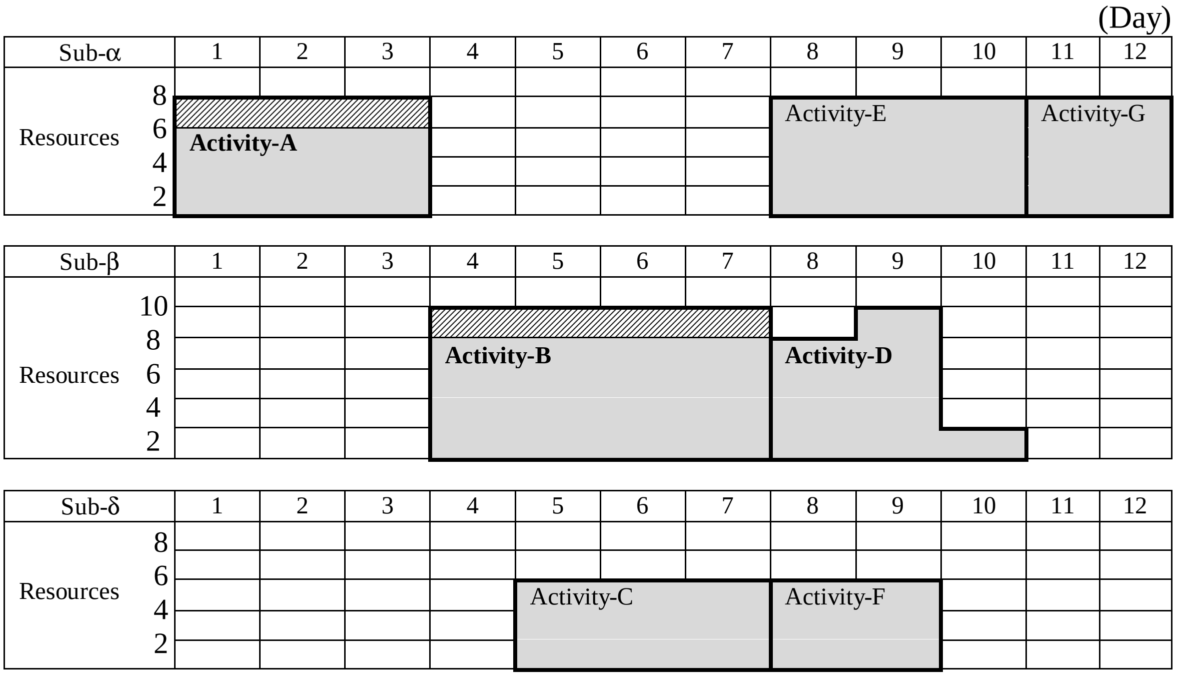
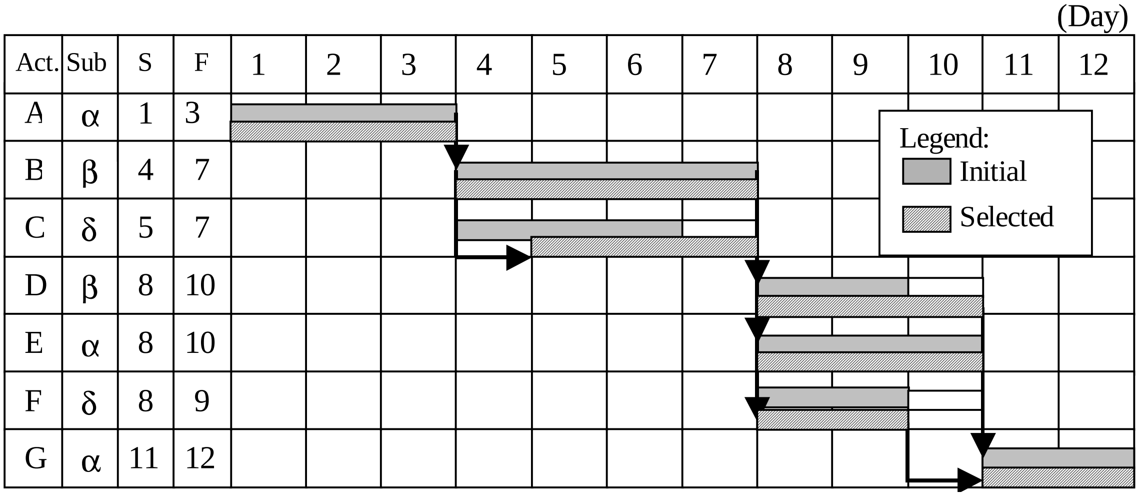


Figure 1-5. Revised resource histogram after TCC

In summary, TCC costs more for some subcontractors when they have different resource availability than their initial resource requirements. This also lowers the resource utilization, even though TCC would guarantee to finish on time. Note that the GC has no consideration of subcontractors' resource utilization in TCC. Figure 1-6 is the revised schedule after TCC. Note that subcontractors have different schedules than either their ES schedule or LS schedule.



Act.	Sub	S	F
A	α	1	3
B	β	4	7
C	δ	5	7
D	β	8	10
E	α	8	10
F	δ	8	9
G	α	11	12

Figure 1-6. Revised schedule after TCC

1.4.2 LOOSE "LAISSEZ-FAIRE" CENTRALIZED COORDINATION

Under loose "Laissez-Faire" centralized coordination (LCC), the objective is to match the resources available to produce a workable schedule. The role of the centralized coordinator is minimal in keeping the original schedule (Tommelein and Ballard 1997). Under LCC, activities are finished when enough resources are provided, like resource-driven scheduling (El-Rayes and Moselhi 1996; Choo et. al. 1999). Without knowing subcontractors' resource availability, the GC instructs subcontractors to start their activities when the preceding activities have been finished and when enough resources are available; i.e., the job is ready for it and its work can proceed unimpeded (Clough and Sears 1991). LCC usually delays the project and some subcontractors might experience cost overruns due to delays of preceding activities as well as their resource deviations. The GC also incurs liquidated damages due to the project delay. Disputes among subcontractors would follow if the causes of delays are not clearly established and the cost overruns are not reimbursed properly.

Under LCC, Sub-α can delay the finish of Activity-A, but Activity-B will be delayed by Activity-A because Activity-B is also allowed to delay its finish. Sub-δ can keep the schedule of Activity-C. Then, Sub-α, Sub-β and Sub-δ must delay the start dates of Activities D, E, F, and G while paying extra costs for importing resources. Next, Sub-α is forced to delay the start date of Activity-G. As a result, the project delays by 2 days. The GC needs to pay liquidated damages for a 2-day project delay.

Figure 1-7 shows the revised resource histogram after LCC. The diagonal hatching indicates required overtime.

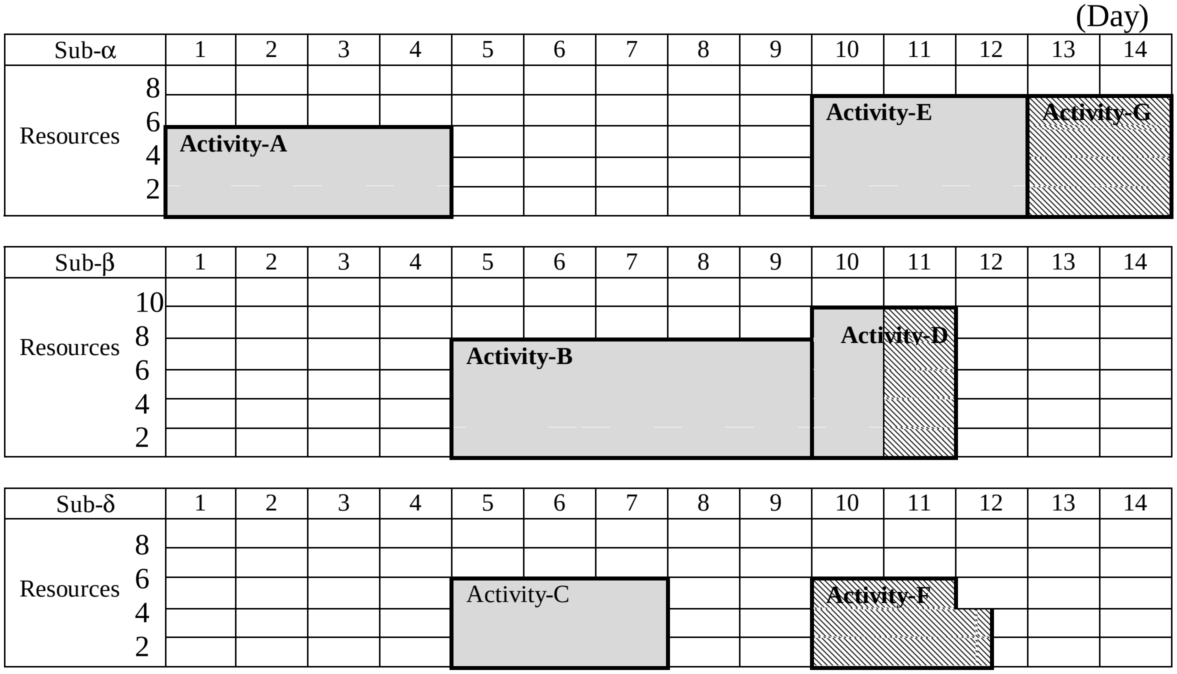
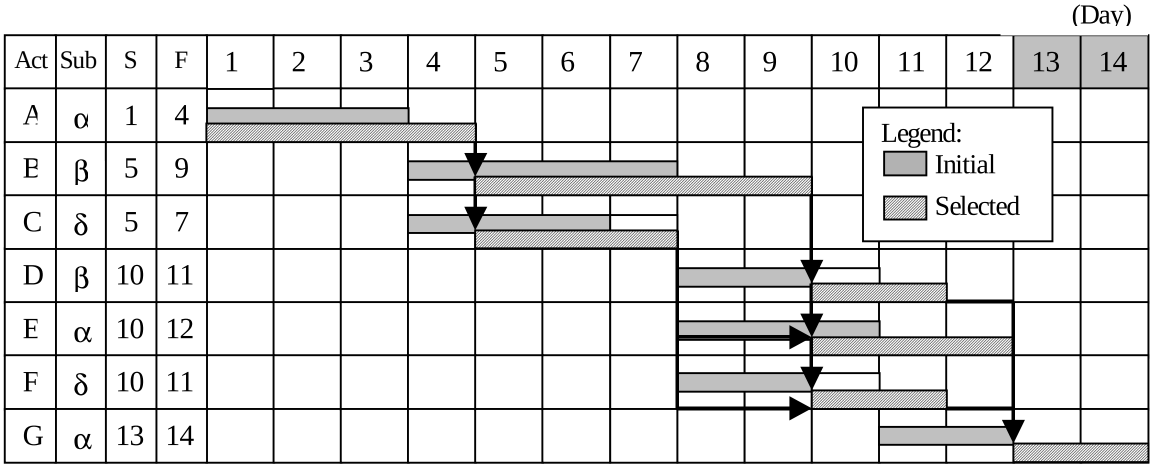


Figure 1-7. Revised resource histogram after LCC

In summary, LCC costs more for some subcontractors when they have to import new resources due to delays of preceding activities as well as their resource deviations. The project missed its completion date and the GC has to pay liquidated damages. Options to expedite the activities instead of paying liquidated damages, like time-cost tradeoff analysis (Fondahl 1961, 1991; Antill and Woodhead 1990), are limited because these need information, such as the cost slope for each activity, that usually is not available to the GC. The revised schedule after LCC appears in Figure 1-8.



Act	Sub	S	F
A	α	1	4
B	β	5	9
C	δ	5	7
D	β	10	11
E	α	10	12
F	δ	10	11
G	α	13	14

Figure 1-8. Revised schedule after LCC

1.5 CHALLENGES FOR DCPSC

From the shortcomings of two current coordination methodologies, I set practical challenges for DCPSC to set up the criteria against which I will measure goodness of a new methodology for DCPSC.

In the example, TCC is equitable in that subcontractors have to pay for their errors in planning, but LCC forces subcontractors into a worse situation by simply delaying affected activities. Some subcontractors get benefits and others lose regardless of fault. The first criterion is whether a new distributed coordination methodology can enable subcontractors to compensate the affected subcontractors for disadvantageous agreements.

In the example, neither TCC nor LCC considered alternatives because of the tightly coupled network of related activities. The second criterion is whether a new methodology can allow subcontractors to identify schedule conflicts, consider alternatives, and resolve schedule conflicts in a tightly coupled network of related activities.

In the example, neither TCC nor LCC had schedule conflicts because the GC maintained work logic during coordination. However, if subcontractors know only their schedules and relationships to other subcontractors' activities, maintaining work logic becomes a big task. The third criterion is whether a new methodology maintains work logic and ensures convergence of distributed coordination.

These challenges have been ignored in past research on the coordination of project planning and scheduling because of its dominant orientation to centralized approaches and the lack of a formal distributed coordination methodology. Current distributed coordination research in the field of cooperative distributed problem solving and multi-agent systems did not address the challenges appropriately due to lack of a formal DCPSC framework and a workable monetary conflict-resolution methodology, as I will discuss in the next chapter.

1.6 ORGANIZATION OF THE DISSERTATION

This dissertation consists of seven chapters. Chapter 1 presents practical motivation, assumptions, a case example, two current centralized coordination methodologies, and challenges for DCPSC. Chapter 2 reviews previous work on coordination of project planning and scheduling and previous work on distributed coordination in the field of cooperative distributed problem solving and multi-agent systems research. It states research objectives, research questions, and research methodology, and then gives a summary of my research.

Chapter 3 presents a new distributed coordination framework for project schedule changes (DCPSC) based on an agent-based negotiation approach wherein a project can be rescheduled dynamically by all of the concerned subcontractors with the help of software agents that evaluate the impact of changes, simulate decisions, and give advice. This chapter defines the formal DCPSC framework, introduces an agent-based negotiation approach, and then discusses relationships to previous work.

Chapter 4 presents a novel agent-based compensatory negotiation (ABCN) methodology to facilitate the distributed coordination of project schedule changes wherein a project can be rescheduled dynamically through negotiations by all of the concerned subcontractors. The methodology consists of a compensatory negotiation strategy based on utility which agents have, multi-linked negotiation protocols by which agents interact with other agents, and message-handling mechanisms for agents to evaluate alternatives and simulate the decision-making. This chapter introduces a new, simpler case example to illustrate the methodology. It also reviews previous work and states my contributions compared to the ABCN methodology.

Chapter 5 presents a multi-agent system for DCPSC wherein a project can be rescheduled dynamically through negotiations by all of the concerned subcontractors. In the multi-agent system called the Distributed Subcontractor Agent System (DSAS), subcontractors interact with their software agents to evaluate the impact of changes, simulate decisions, and get the negotiation results that they need to reschedule the project. It discusses DSAS architecture, supporting state-of-the-art technologies, and the DSAS implementation.

Chapter 6 demonstrates the significance of the ABCN methodology on DCPSC through evaluation tests. It compares two centralized coordination methodologies used in current practice to the ABCN methodology in terms of extra costs and project duration. It presents charrette test results of the DSAS, which tested the effectiveness of DSAS compared to manual centralized processes. This chapter shows the results of measurements on system performance of DSAS that in turn show that DSAS scales manageably.

Chapter 7 summarizes this dissertation with a summary of the first six chapters, contributions, practical demonstrations, limitations of the research, possible future research directions, and value to industry.

CHAPTER 2

POINTS OF DEPARTURE

This chapter reviews previous work on coordination of project planning and scheduling and previous work on distributed coordination in the field of cooperative distributed problem solving and multi-agent systems research. It sets research objectives, poses research questions, and describes research methodology.

2.1 PREVIOUS COORDINATION OF PROJECT PLANNING AND SCHEDULING

Numerous research papers have recognized a major problem in the Critical Path Method (CPM) network approach, which assumes unlimited resource supplies, and have provided frameworks to address various limited-resource issues in construction planning and scheduling, including resource leveling (Easa 1989; Harris 1990; Seibert and Evans 1991; Martinez and Ioannou 1993; Russell and Dubey 1995; Son and Skibniewski 1999; Hiyassat 2000), resource-constrained allocation (Hegazy et al. 2000), and resource-driven scheduling (Moselhi and Lorterpong 1993; Moselhi and El-Rayes 1993; El-Rayes and Moselhi 1996 & 2001). Others have applied various techniques such as linear programming (Shah and Baugh 1993; Mattila and Abraham 1998) and Genetic Algorithms (Chan et al. 1996; Leu and Yang 1999; Hegazy 1999) to the limited-resource problem. However, few current frameworks address the difficulties of gathering information in the coordination of subcontractors' resource-driven schedule changes. The existing centralized frameworks are insufficient because the information needed for centralized resource-based scheduling, such as the resource constraints, is usually kept private by subcontractors (Choo and Tommelein 2000) and is usually not available for the centralized frameworks, as stated in one of the assumptions of distributed coordination of project schedule changes (see Section 1.2).

When only some subcontractors benefit from the coordination of their resource-driven schedule changes and when general contractors are not willing to coordinate the work of subcontractors (Tommelein and Ballard 1997), subcontractors need to work together with minimal information sharing. Subcontractors need a new distributed coordination methodology that allows evaluating the impact of their changes and making appropriate decisions.

Current distributed frameworks in construction and in the broader project management and AI research literature have inadequately addressed challenges for distributed coordination of project schedule changes. They have not provided a monetary conflict-resolution mechanism, which is their main shortcoming, even though some of them have provided various conflict-resolution mechanisms for interactions between participants (Koo 1987; Khedro et al. 1993; Jin and Levitt 1993; Gomes et al. 1994). ProcessLink (Petrie et al. 1998) identifies dependencies among activities and participants but does not specify a conflict-resolution mechanism.

As this review of previous literature demonstrates, the distributed coordination with monetary conflict-resolution mechanism is a new problem domain that current research does not explore. The problem I am trying to solve is rescheduling activities through distributed coordination when subcontractors have available resources that differ from the needed resources. They need a distributed coordination framework that includes a monetary conflict-resolution methodology, wherein a project can be rescheduled dynamically by all of the concerned subcontractors, while maintaining schedule logic and keeping their information private.

2.2 PREVIOUS DISTRIBUTED AGENT-BASED COORDINATION

Many Cooperative Distributed Problem Solving (CDPS) and Multi-Agent Systems (MAS) papers have proposed various distributed coordination methodologies, but I found that the research has shortcomings in applications for distributed coordination of project schedule changes.

First, I found that none of the papers mentioned an explicit method for the transfer of utility units ("money") to compensate for disadvantageous agreements. Allowing the transfer of utility is very important for coordination of project schedule changes. When subcontractors want to reschedule their activities, the rescheduling has external effects on succeeding subcontractors' resource profiles, causing external costs for the affected subcontractors, similar to externalities in *Welfare Economics and Social Choice Theory* (Feldman, 1980). This is not to say that all of the external costs are negative. Some of them are positive. In case of negative external costs, the external effects destroy the Pareto optimality (Feldman, 1980) in a construction master schedule. By allowing the transfer of utility ("money") to compensate for the external costs, the group of subcontractors prefers the changed project schedule to the initial project schedule. Consequently, a new project schedule is "Pareto superior" (Feldman, 1980, p. 140) to the initial project schedule.

My monetary conflict-resolution methodology is more efficient than using incentives or reward mechanisms found in some market-based systems (Malone et al. 1988; Wellman 1993; Shoham and Tanaka 1997). The Clarke tax voting mechanism (Ephrati and Rosenschein 1996) collects taxes centrally, but provides no way to distribute the collected taxes. The unified negotiation protocol (Rosenschein and Zlotkin 1994) does not provide an explicit monetary conflict-resolution mechanism, so it uses an implicit method – working together after flipping a coin.

My compensatory negotiation methodology differs from payment via contracts used in other MAS research (Sandholm 1993; Sen and Durfee 1996), in which any profit-seeking bid from agents might prevent a system from reaching a better solution. Compromise via negotiation (Sycara 1989) provides a way of transferring utilities between agents through a central mediator, but it is implicit and not for compensation of disadvantageous agreements. Distributed constraint satisfaction (Yokoo et al. 1992, 1998) and distributed search (Durfee and Montgomery 1991; Sycara et al. 1991; Decker and Lesser 1992) find a satisfactory solution that needs no disadvantageous agreements for any agent.

Another finding is that most CDPS and MAS applications employ pair-wise negotiation or multi-lateral (auction) protocols, which are based on the Contract Net Protocol (Smith 1980) that would be unsuitable for coordinating the tightly coupled project schedules. Pair-wise negotiation or multi-lateral (auction) protocols cannot capture external effects of their agreements on other agents because agents do not know the consequences of their decision until getting the responses from the succeeding agents. Rather, they implicitly assume that there are no externalities.

The third finding is that most CDPS and MAS applications do not guarantee the convergence of distributed computation because protocols that cannot ensure consistency among agents' knowledge cannot ensure convergence. Some distributed search frameworks use unique ID (Yokoo et al 1992), pecking order (Durfee and Montgomery 1991), or heuristic order (Sycara et al 1991), but none of them use the Critical Path Method (CPM) (Fondahl 1961) for coordination of message passing.

2.3 RESEARCH OBJECTIVES

To overcome past research limitations for the distributed coordination of project schedule changes (DCPSC), the overall purpose of the proposed research was fourfold:

(1) To formalize and generalize a DCPSC framework;
(2) To formalize and generalize an agent-based compensatory negotiation methodology to enable the DCPSC framework;
(3) To implement a distributed subcontractor agent system to demonstrate the DCPSC framework; and
(4) To test the DCPSC framework

Figure 2-1 illustrates the components of the research. I abstracted the practical problem to three components: schedule, resource constraint, and extra cost. When a schedule does not match resource availabilities, this generates resource constraints and subcontractors incur extra costs. Subcontractors will use the extra cost information for distributed coordination to re-schedule the project. Therefore, the research goal is to provide a distributed coordination framework that reschedules projects to lower the sum of subcontractors' costs associated with their resource constraints in cases of changes in subcontractors' resource availabilities.

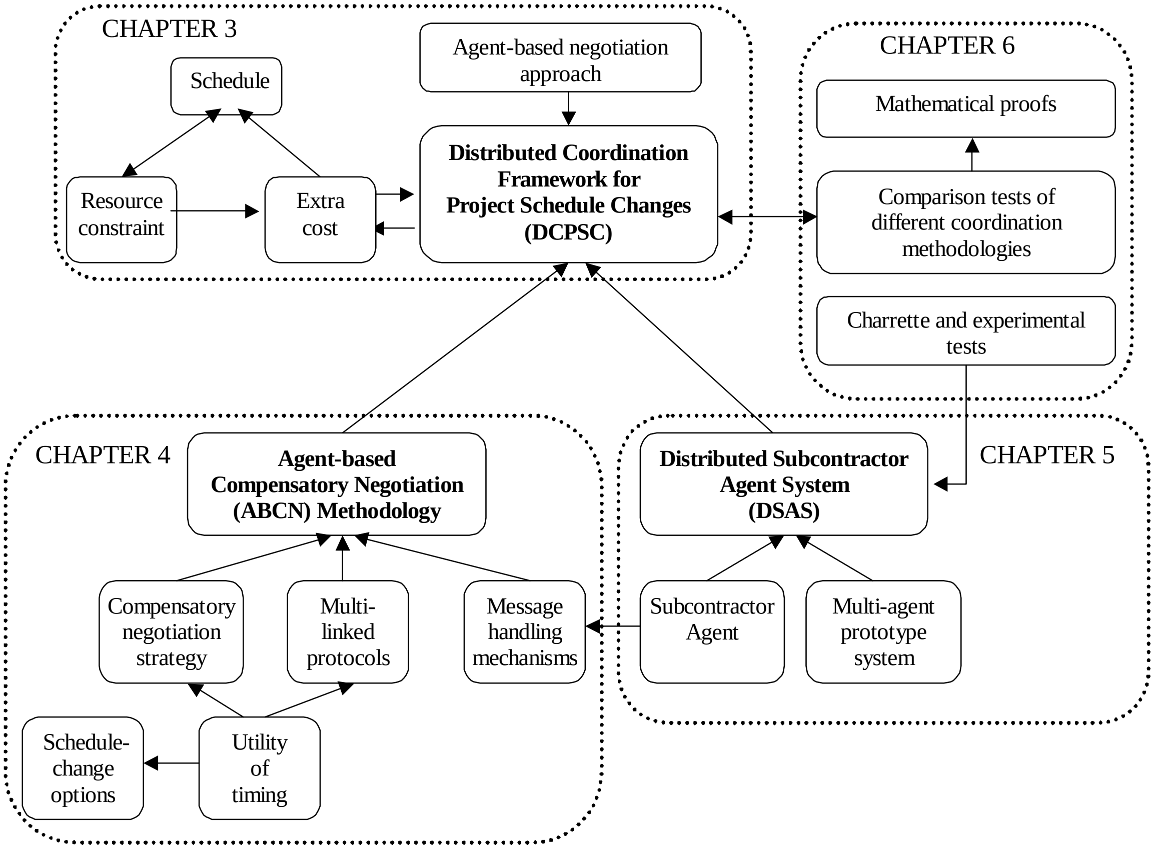

CHAPTER 3
Schedule
Resource constraint
Extra cost
Agent-based negotiation approach
Distributed Coordination Framework for Project Schedule Changes (DCPSC)
CHAPTER 6
Mathematical proofs
Comparison tests of different coordination methodologies
Charrette and experimental tests
CHAPTER 4
Agent-based Compensatory Negotiation (ABCN) Methodology
Compensatory negotiation strategy
Multi-linked protocols
Message handling mechanisms
Schedule-change options
Utility of timing
Distributed Subcontractor Agent System (DSAS)
CHAPTER 5
Subcontractor Agent
Multi-agent prototype system

Figure 2-1. Summary of research

To achieve my research goal, I proposed a distributed coordination framework for project schedule changes and an agent-based compensatory negotiation methodology to enable the framework. I implemented a distributed subcontractor agent system to demonstrate the effectiveness of the DCPSC framework. Through verification tests, the DCPSC framework demonstrates that it solves the problems successfully. Chapters 3 to 6 elaborate upon certain functionalities of these components.

2.4 RESEARCH QUESTIONS

I posed six research questions intended to fulfill the research objectives for distributed coordination of project schedule changes:

Q1: What formalism and approach can enable every subcontractor to consider its own activities, but will also enhance global outcomes?

My answer to this question presents a new DCPSC framework, wherein all of the concerned subcontractors can reschedule a project dynamically, based on the social welfare function (Varian 1978) and an agent-based negotiation approach adopting the typed-message agent (Petrie 1996). Chapter 3 presents the novel distributed coordination framework for project schedule changes and introduces an agent-based negotiation approach.

Q2: What formalism can enable agents to compensate other agents for disadvantageous agreements?

My answer to this question formalizes a new compensatory negotiation strategy for DCPSC, which is necessary for agents to compensate other agents for disadvantageous agreements based on the externalities (Feldman 1980). Chapter 4 presents the novel definitions of utility of timing and schedule change options, which allow the transfer of utility for compensation of disadvantageous agreements.

Q3: What protocols can enable agents to identify and resolve schedule conflicts in a tightly coupled network of related activities?

My answer to this question formalizes new negotiation protocols for DCPSC, which are suitable for coordinating the tightly coupled project schedules, by extending the Contract Net Protocol (Smith 1980). Chapter 4 describes novel multi-linked protocols, which can enable agents to identify and resolve schedule conflicts in a tightly coupled network of related activities.

Q4: What mechanisms can enable agents to maintain work logic and ensure convergence of distributed coordination?

My answer to this question formalizes message-handling mechanisms for DCPSC, which coordinate message passing based on the Critical Path Method (CPM) (Fondahl 1961). Chapter 4 describes novel message-handling mechanisms, which can enable agents to maintain work logic and ensure convergence of distributed coordination.

Q5: How can a multi-agent system be developed to implement the distributed agent-based coordination methodology?

My answer to this question presents a new Java-based multi-agent system for distributed coordination of project schedule change, by using the JATLite (Java Agent Template, Lite) (Jeon et al. 2000). Chapter 5 describes agent-based software engineering to produces the novel Distributed Subcontractor Agent System (DSAS), which can enable human subcontractors to interact with software agents to reschedule projects.

Q6: What are the impacts on DCPSC of a distributed agent-based coordination methodology?

My answer to this question presents the significance of the distributed agent-based coordination methodology through comparison tests, Charrette tests (Clayton et al. 1998), and experimental tests. Chapter 6 presents evaluation results: (1) The distributed agent-based coordination methodology produces a solution that is better than or equal to either of the two current centralized coordination methodologies, in terms of project cost; (2) The distributed agent-based coordination methodology produces the solution and finds the lower-cost solution faster than conventional manual processes; and (3) The distributed agent-based coordination methodology is scalable.

2.5 RESEARCH METHODOLOGY

To answer the research questions, the research methodology had four major components:

(1) Modeling,
(2) System building,
(3) Verification through testing, and
(4) Evaluation.

I modeled a framework and a methodology for distributed coordination of subcontractors' resource-driven schedule changes through the case example and a survey of background literature on AI planning, Cooperative Distributed Problem Solving (CDPS), and Multi-Agent Systems (MAS) research. Using the developed model, I built an agent-based distributed decision-making system in which I tested and analyzed several case examples with changing resource profiles to prove the concept in the theory and verify the system. Finally, I evaluated the results of my research according to its contributions.

I identified six specific tasks corresponding to the research methodology described above. The following sections show the evolutions of the specific tasks.

(1) Build up research background

The research built on four research foundations: (1) construction planning and scheduling, (2) AI planning, (3) cooperative distributed problem solving, and (4) coordination theory in multi-agent systems. This chapter and Appendixes B and C review the selected previous work related to the specific topics.

(2) Develop a distributed coordination framework for project schedule changes

A distributed coordination framework for project schedule changes (DCPSC) models the coordination of subcontractors' resource-driven schedule changes as distributed coordination processes by subcontractors to enhance the project network schedule for lowering the sum of subcontractors' costs associated with their resource constraints. It does this by rescheduling the project subject to the precedence relationships among project activities, when changes occur in subcontractors' resource availabilities.

In the new DCPSC framework, I represent the subcontractors as software agents that simulate negotiations on behalf of subcontractors. This enhances the project network schedule for lowering the sum of subcontractors' costs associated with their resource constraints. Chapter 3 defines the DCPSC framework and the agent-based negotiation approach.

(3) Develop an agent-based compensatory negotiation methodology

Development of an agent-based compensatory negotiation (ABCN) methodology for agents started with modeling subcontractors' resource-driven schedule changes on activity performance through extra costs in addition to normal costs. Then, I represent the models through the schedule-change options that consist of extra costs associated with alternative start/end dates. The compensatory negotiation methodology represents the coordination between subcontractors as negotiation between software agents that employ the compensatory negotiation strategy based on utility of timing[2].

Within the compensatory negotiation methodology, I developed the multi-linked negotiation protocols to model the schedule externalities that are natural for tightly coupled project schedules. I also developed message-handling mechanisms to model decision-making mechanisms of subcontractors in the negotiating agent for the methodology. Chapter 4 describes the compensatory negotiation strategy based on utility, multi-linked negotiation protocols, and message-handling mechanisms for the ABCN methodology.

(4) Develop and implement the distributed subcontractor agent system

To demonstrate the distributed coordination based on agent-based compensatory negotiation methodology, I developed the Distributed Subcontractor Agent System (DSAS), which is a Java-based multi-agent system. In DSAS, subcontractor agents that implement the agent-based compensatory negotiation methodology negotiate with other subcontractors based on schedule-change options for distributed coordination of project

[2] In this thesis, utility of timing is a real-valued number ("money"), which describes the difference between the cost of the initial schedule and the costs of alternatives for the activity (see Section 4.2.1 for a further discussion of this definition).

schedule changes. Through Graphic User Interfaces (GUIs), human subcontractors can interact with their subcontractor agents to provide schedule-change options and get negotiation results. The Agent Message Router (AMR) provides a robust message-passing infrastructure. Chapter 5 shows an architecture and implementation details of DSAS.

(5) Verify DCPSC, ABCN methodology, and DSAS

The verification of my research needed a number of experimental tests to demonstrate the effectiveness of the proposed distributed coordination approach. In a test case, I compared two centralized coordination methodologies used in current practice — tight "Iron-Fist" centralized coordination (TCC) and loose "Laissez-Faire" centralized coordination (LCC) — to the DCPSC. I generalized test results in mathematical proofs that show that the proposed distributed coordination approach always finds a solution that is better than or equal to those of the two centralized coordination methodologies in project performance (cost, duration, and resource utilization).

I conducted charrette tests on a test case to demonstrate that the resulting agent-based distributed scheduling system finds the lower-cost solution faster than conventional manual processes. I also measured system performance (number of messages and time taken) on several test cases to show that the proposed distributed coordination approach scales without becoming infeasible for practical applications. Chapter 6 describes test methodologies and results.

(6) Evaluate the research results

Throughout the proposed research, I allocated time for evaluating the research results in terms of their contributions to knowledge. I had a number of evaluation meetings at the end of each specific task. The evaluations confirmed that my research has distinguishable contributions because my resulting DSAS system worked as planned and my contributions clearly extend beyond previous work. My research has validity and applicability since two groups of users have verified that the resulting system produces a solution that is better than or equal to the initial solution and can be applicable to real construction projects. Chapter 7 summarizes the contributions and limitations of my research.

2.6 SUMMARY OF POINTS OF DEPARTURE

This chapter reviewed previous work on project planning and scheduling and previous work on distributed coordination in the field of cooperative distributed problem solving and multi-agent systems research. It set research objectives to overcome the current research limitations of DCPSC, posed research questions to fulfill the research objectives for DCPSC, and described the research methodology I used to answer the research questions.

CHAPTER 3

DISTRIBUTED COORDINATION OF PROJECT SCHEDULE CHANGES BASED ON AGENT-BASED NEGOTIATION APPROACH

This chapter presents a new distributed coordination framework for project schedule changes (DCPSC) based on an agent-based negotiation approach wherein a project can be rescheduled dynamically by all of the concerned subcontractors with the help of software agents that evaluate the impact of changes, simulate decisions, and give advice. This chapter defines the formal DCPSC framework, introduces an agent-based negotiation approach, and then discusses relationships to previous work.

3.1 INTRODUCTION

The distributed coordination of project schedule changes consists of three components similar to Oberlender's (1993) project definition: project schedule, resource constraints, and extra costs, as illustrated in Figure 3-1. Note that every subcontractor has its own resource constraint and extra cost information. That is, there are multiple resource constraints and extra costs for multiple subcontractors, but there is one project schedule for a project.

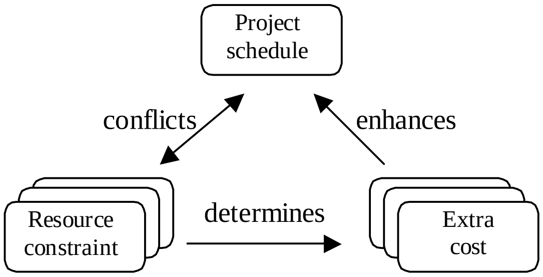


Figure 3-1. Distributed coordination of project schedule changes

Project schedule refers to the schedule that represents the logical sequencing and timing of the work to be performed. Usually, a general contractor prepares the project schedule and assigns parts of the schedule to specialty contractors, i.e., subcontractors. Nonetheless, the separately assigned project schedules interrelate with each other because they are a part of the whole project schedule. *Resource constraints* represent the differences between the needed resources and subcontractors' available resources. Therefore, the subcontractors want to change the project schedule in their favor in a way that enhances their resource utilization. Resolving resource constraints without violating the work logic is the task of subcontractors. *Extra cost* refers to the subcontractors' worst extra costs determined by their resource constraints.

When there are resource constraints that conflict with the project schedule, the resource constraints generate schedule changes and determine the extra cost information. Subcontractors will try to lower the extra costs through distributed coordination among subcontractors, and accept the schedule changes that lower the extra costs associated with the schedule changes, which will enhance the project schedule.

3.2 DISTRIBUTED COORDINATION FRAMEWORK FOR PROJECT SCHEDULE CHNAGES

In order for subcontractors to consider their own activities, and also enhance global outcomes, I developed the distributed coordination framework for project schedule changes, based on the social welfare function (Varian 1978). The social welfare function is "some sort of function that aggregates individual utility functions to come up with some sort of social utility." (Varian 1978, p. 153). Varian also states, "The social welfare function is increasing in each of its arguments – if you increase an agent's utility without decreasing anybody else's utility then society is made better off." (Varian 1978, p. 153).

I formalize the distributed coordination framework of project schedule changes as follows. A set of subcontractors A = $\{A_{1,} . . ., A_n\}$ must produce a collective decision over a set of activities $\{a_1, . . ., a_m\}$. Each subcontractor needs to know a utility for its alternatives of its own activity, by taking into consideration the extra costs to others of its course of action and weighing it against its own extra costs. I define a social choice function, *E*, such that *E* represents the group choice, as follows:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{m} Cost_{(i,j)}$$

> Where *E* and $Cost_{(i,\,j)}$ = the sum of the subcontractors' extra costs for all *m* activities of *n* subcontractors and the j^{th} activity, which belongs to the i^{th} subcontractor.

Unlike the ideals of social choice function, which is the summation of individual utility, my social choice function is the summation of individual subcontractor's extra costs, which is indeed the negative utility for subcontractors and social welfare. In my social choice function, the utility for subcontractors and social welfare increases when the extra costs decrease.

To increase individual utility and social welfare together, therefore, I set the objective of distributed coordination of project schedule changes so as to lower *E*, i.e., the sum of subcontractors' costs associated with their resource constraints, subject to the precedence relationship among project activities:

$$lower \qquad E = \sum_{i=1}^{n} \sum_{j=1}^{m} Cost_{(i,j)}$$

subject to

$$\forall x, Finish_x < \min_{y \in S_x} \{ Start_y \}$$

> Where *E* and $Cost_{(i,\,j)}$ = the sum of the subcontractors' extra costs for all *m* activities of *n* subcontractors and the j^{th} activity, which belongs to the i^{th} subcontractor, respectively; $Finish_x$ = finish date of activity *x*; $Start_y$ = start date of activity *y*; and S_x = set of activities which must succeed activity *x*.

One distinguishing feature of the problem definition is that the problem will be solved through collaboration of all the concerned subcontractors in a distributed manner, without assuming that a central coordinator knows all the information needed for coordination and that subcontractors are benevolent. Another notable feature is that every subcontractor only considers its own activities, but it will also enhance global outcomes. The third feature is that distributed coordination maintains the logical sequencing of the work to be performed, subject to the precedence relationship among project activities, even though it might change the timing of the work. These features of the distributed coordination of project schedule changes overcome shortcomings of the current centralized coordination approaches.

3.3 AGENT-BASED NEGOTIATION APPROACH

The previous section reveals three important issues of distributed coordination for project schedule changes: distributed coordination by competitive subcontractors, socially rational decision-making, and maintaining the logical sequence of the work. In this section, I discuss the adoption of the agent-based negotiation approach, which is needed to overcome the difficulties stemming from these issues.

3.3.1 AGENT

Because of the huge number of interactions among subcontractors throughout the distributed coordination for project schedule change, each subcontractor should have a software agent, which is a program capable of communicating with other software agents using an Agent Communication Language (Khedro et al. 1993). The agent exchanges messages with other agents to evaluate changes, and advises its human subcontractor to make a decision.

For this research, I adopted the so-called “Typed-Message Agent (TMA)” (Petrie 1996). Agent communities define TMA as those agents that must exchange messages to accomplish a task. For agent characteristics, TMA stresses peer-to-peer communications and message passing based on shared, typed protocols and semantics to which the agent communities have committed, along with other important features for distributed coordination of project schedule changes (Jeon 2000):

- Performing a task that would normally be performed by a person
- Using peer-to-peer communication, which differs from the traditional client/server model
- Communicating with a common agent language, such as Knowledge Query and Manipulation Language (KQML) (Finin et al. 1994; Labrou and Finin 1997) and the FIPA Agent Communication Language
- Obeying agent negotiation protocols, the semantics of which the agent community shares
- Maintaining state information, and acting on it, which distinguishes agents from other object-oriented software programs

3.3.2 AGENT-BASED NEGOTIATION

There are many definitions of agent-based negotiation in the cooperative distributed problem solving (CDPS) and multi-agent systems (MAS) research communities. In the context of this thesis, I define agent-based negotiation as the process of resolving conflicts among affected agents by increasing knowledge about others' intentions through the structured exchange of relevant information. The objective of agent-based negotiation is to improve mutual agreements for conflicting agents so that the results of negotiations should not make any conflicting agent worse off than before the negotiations. My definition of agent-based negotiation is limited in the sense of capturing human negotiations, but parallels definitions by previous researchers (Davis and Smith 1988; Sycara 1989; Adler et al. 1989; Durfee et al. 1989; Rosenschein and Zlotkin 1994; Khedro 1996; Glossary 1999) in CDPS and MAS research communities, as quoted below. The readers who are familiar with these definitions might want to skip these and jump to Section 3.3.3.

In “Negotiation as a metaphor for distributed problem solving,” Davis and Smith (1988) state:

> The central element in our approach to a problem solving protocol is the concept of negotiation. By negotiation, I mean a discussion in which the interested parties exchange information and come to an agreement. For our purpose, negotiation has three important components: (a) there is two-way exchange of information, (b) each party to the negotiation evaluates the information from its own perspective, and (c) final agreement is achieved by mutual selection (p. 337).

In ”Multiagent compromise via negotiation,” Sycara (1989) writes:

> The negotiation process involves identifying potential interactions either through communications or by reasoning about the current states and intentions of other agents in the system and modifying the intentions of these agents so as to avoid harmful interactions or create cooperative situations (p. 120). Negotiation is a process in which the parties iteratively propose compromises and argue with each other until a settlement is reached (p. 122).

In “Conflict-resolution strategies for nonhierarchical distributed agents,” Adler et al. (1989) say:

> Negotiation is a process of communication established between two conflicting agents in which they try to develop or refine their plans jointly so that the goals of each are satisfied. Agents exchange representations of their goals, look for conflicts in realizing them, develop understanding of the motivations behind those goals, look for actions they can take jointly to meet their own goals while at the same time helping other agents achieve their goals. Negotiation is engaged when a conflict is obvious to the various parties and no predefined mechanism exists for resolving it (p. 147).

In “Trends in cooperative distributed problem solving,” Durfee, Lesser, and Corkill (1989) state:

> We define negotiation as the process of improving agreement (reducing inconsistency and uncertainty) on common viewpoints or plans through the structured exchange of relevant information. Although these descriptions of negotiation capture many of our intuition about human negotiation, they are too vague to provide blueprints for how to get AI systems to negotiate (p. 68).

In *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, Rosenschein and Zlotkin (1994) say:

> Negotiation denotes the process of several agents searching for an agreement. Agreement can be about price, about military arrangements, about a meeting place, about joint actions, or about a joint objective. The search process may involve the exchange of information, the relaxations of initial goals, mutual concessions, lies, or threats. The way we use it, the term negotiation is closely related to the idea of reaching consensus. Separate agents, with potentially disparate interests, attempt to make a group choice over well-defined alternatives (p. 19).

In his dissertation, “A distributed problem-solving approach to collaborative facility engineering,” Khedro (1996) writes:

> Negotiation is a process of resolving conflicts between two conflicting, intelligent systems in which these systems attempt to develop or refine their plans jointly so that the goals of each are satisfied. In the course of negotiation, intelligent, interacting systems typically develop understanding of each other at the goal level, and thus can find complex solutions involving trade-offs and novel approaches to solving shared problems that neither could have recognized independently [Adler 1989].

In Glossary of *Multiagent Systems*: *A Modern Approach to Distributed Artificial Intelligence*, Huhns, Stephens, and Weiss (1999) state:

> Negotiation – interaction among agents based on communication for the purpose of coming to an agreement. Negotiation has much to do with distributed conflict resolution and decision making, and requires that the agents use a common language. In the course of negotiation an agent makes a proposal which then is commented by other agents. Negotiation may be interpreted as coordination among competitive or simply self-interested agents. Another common interpretation of negotiation is that of a distributed, communication-based search through a space of possible solutions (p. 598).

3.3.3 ADVANTAGES OF AN AGENT-BASED NEGOTIAITON APPROACH

TMAs provide several advantages over recent object-oriented approaches and other agent systems. The key idea of TMAs is their ability to model distributed coordination among subcontractors throughout the conflict-resolution negotiation processes. TMAs can model the subcontractors as software agents performing a task on behalf of human subcontractors, while modeling the interactions among subcontractors as agent negotiation protocols based on agent communication language.

With the ability of modeling distributed coordination, the agent-based negotiation approach based on TMA is a powerful tool to overcome the difficulties by the distributed approach. Software agents can communicate rapidly with each other over the Internet, which allows subcontractors to coordinate project schedule changes in a distributed manner with the agent-based compensatory negotiation methodology to be introduced in the next chapter.

3.4 RELATION TO PREVIOUS WORK ON COORDINATION OF PROJECT PLANNING AND SCHEDULING

In this section, I review previous work on coordination of project planning and scheduling and compare in detail my DCPSC framework and agent-based negotiation approach to previous work, in terms of three important issues: distributed coordination by competitive subcontractors, socially rational decision-making, and maintaining the logical sequence of the work.

3.4.1 AGENT-BASED SOFTWARE INTEGRATION

Khedro and others (1993) proposed a framework for collaborative distributed facility engineering based on Cooperative Distributed Problem Solving and Agent Software Engineering. Their framework allows the integration of existing design software applications through the implementation of agent programs in the Federation Architecture. They presented the framework for collaboration of designers in the integrated distributed environment by providing effective methods for coordinated exchange of information and protocols and a strategy for collaboration.

Their distributed-problem-solving approach to the facility design domain inspired me to apply a similar approach to the DCPSC framework. While their agents represent the cooperative design software, my agents represent competitive subcontractors who execute the activities according to the given schedule. Therefore, an addition to their agent model will be an agent-based negotiation approach for resolving conflicts between agents to make socially rational decisions, rather than resolving conflicts with authority. The organization structure in their framework is the Federal Architecture that involves facilitators, which are system programs for facilitating and coordinating the interaction of agents in an environment. In my DCPSC framework, there is no central facilitator. Agents themselves coordinate the interaction of agents.

3.4.2 COOPERATIVE DISTRIBUTED PLANNING MODEL

Koo (1987) proposed a distributed model for synchronizing and monitoring plans made independently by intelligent agents via communication during performance cycles. The proposed model allows agents to plan autonomously and then synchronize their plan via a commitment-based communication vehicle, while maintaining the logical sequence of the work. He formulated his agent model based on these two assumptions: rationality and willingness to compromise. His thesis included an autonomous nonlinear planning algorithm that functions incrementally in a multiagent environment.

The considerations of resource constraints of subcontractors are very important in the DCPSC framework, but the resource constraints were not addressed comprehensively in his cooperative distributed planning model. An agent-based negotiation approach for resolving schedule conflicts among agents that have resource constraints is an addition to his framework. The DCPSC framework and an agent-based negotiation approach facilitate cooperation by employing monetary compensation as the motivation to cooperate with other agents, rather than a simple negotiation process based on an agent's willingness to compromise.

3.4.3 DISTRIBUTED JOB-SHOP SCHEDULING

Gomes et al. (1994) proposed a distributed scheduling framework by applying a distributed-problem-solving approach to job-shop scheduling. They viewed the distributed scheduling system as a hierarchical organization with three main levels: the strategic level, the tactical level and operational level. This framework is very suitable for resolving schedule conflicts by a central coordinator, such as their strategic agent, in the manufacturing industry.

My DCPSC framework employs a one-level organization with all agents at the tactical level. The main difference is that agents at the tactical level interact with each other with their interests and resolve conflicts by themselves with an agent-based negotiation approach. Therefore, my DCPSC framework does not need another higher-level of coordination to resolve conflicts, such as the strategic level. A new addition to their system is an agent-based negotiation approach that resolves conflicts at the same level, while making socially rational decisions.

3.4.4 i-AGENT

Jin and Levitt (1993) proposed the i-AGENTS framework based on organization theory and Distributed Artificial Intelligence. i-AGENTS is a computerized framework for studying organizational problem solving in multi-agent teams. i-AGENTS consists of high-level concepts: tasks, agents, organization, and communication. i-AGENTS can simulate and analyze the organizational behavior of teams in an engineering domain. Their approach considers how to organize the communication structure among the set of agents, given a set of tasks and agents.

Since my DCPSC framework focuses on distributed coordination of schedule changes among subcontractors, and my objective is to explore the relationship between the methodology and project performance, my agent model builds expert systems coupled with communication capability rather than computer systems based on knowledge and mental states like their agent model. Unlike i-AGENTS, which uses agents to study how to organize human actors using parameters for task and agent specifications, my agent-based negotiation approach uses agents to coordinate project schedules using real data for task and agent specifications. Therefore, my work is not an organizational design approach. I model the organization structure of agents as a virtual organization encompassing multiple organizations that participate in a project. Within the virtual organization, there is no formal organization structure for agents; rather, agents organize dynamically based on their interrelationships with activities.

3.4.5 PM IN PROCESSLINK

ProcessLink (Petrie, et al., 1998) provided an agent-based framework supporting the distributed task interactions of modern enterprise, especially for integrated project management, which interleaves design and construction planning. Change notification needs to maintain dependency information among plan and design tasks. When distributed between designer and planners, no one may have all of the information to perform such notification. Since the central problem of distributed interleaved planning is change propagation, they proposed a coordination model as a set of dependencies among tasks and a computer system to manage the dependency information in order to coordinate a distributed project, while maintaining the logical sequence of the work. Their ProcessLink system consists of Redux', which is a general model of design, Constraint Manager (CM), which manages Constraints Solvers, Plan Manager (PM), which performs global tracking of plan elements, and the JATLite agent infrastructure as a general “bus” for the exchange of messages.

PM is a domain-independent centralized agent of the ProcessLink system for planning and scheduling design activities. The PM model consists of scheduling goals and tasks. Therefore, it integrates design with planning and scheduling through the PM that performs global tracking of the plan elements, using the Redux’ and the CM. The PM allows planning and scheduling to be distributed among project members according to their responsibilities and expertise. However, their PM only suggests or makes changes when the schedule can be improved in one instance without making it longer in another way, i.e., only in cases where the schedule is not Pareto optimal.

My DCPSC framework addresses the issue where one agent wants to make a change that will adversely impact others, but is willing to pay for it, i.e., making socially rational decisions. That is the main difference between the two systems. In my DCPSC framework, the subcontractor agents can interact and negotiate with each other to make a better schedule through the agent-based negotiation approach. The agent-based negotiation approach is a crucial supplement to the PM in the ProcessLink system, which otherwise only provides coordination of schedule changes due to design changes, but provides no distributed mechanism for agreeing on change options.

3.4.6 RESOURCE-DRIVEN SCHEDULING MODEL

El-Rayes and Moselhi (1996) developed a resource-driven scheduling algorithm for repetitive activities. Their algorithm produces a schedule that complies with precedence relationships, crew availability, and crew work continuity constraints. Their algorithm works in two stages: the first achieves compliance with logical precedence relationships and crew availability constraints, and the second achieves compliance with the crew work continuity constraint. Their focus was to maintain work continuity in repetitive activities in a way that enables timely movement of crews from one unit to the next, avoiding crew idle time. They developed a computer model, utilizing object-oriented programming, where objects represent the activities and their relationships.

Their computer model satisfies the issue of maintaining the logical sequence of the work using a simpler protocol. Their model initiates generating the schedule by sending a message to the first activity. The activity schedules resources using the resource-driven scheduling algorithm and then sends messages to all the succeeding activities throughout the entire project network. However, they did not consider the extra costs associated with resource constraints, nor did they provide any conflict-resolution methodology for resolving schedule conflicts. In their centralized framework, activities are simply scheduled in the order in the network schedule such that resource availability complies with resource requirements.

In my DCPSC framework, activities are re-scheduled only if all the succeeding activities can be scheduled in a way that is better than or equal to the initial schedule in terms of cost for making socially rational decisions. In contrast to their simple resource-driven scheduling algorithm, my agent-based negotiation approach allows subcontractors to evaluate the impacts of their changes quantitatively, resolve schedule conflicts, and make better schedules.

3.4.7 WORKPLAN

Choo, Tommelein, Ballard, and Zabelle (1999) presented a crew-level planning system as the last planner, based on resource availability and other factors. They focused on crew work continuity, which is the main objective of job-shop scheduling. Indeed, they were trying to apply the techniques of job-shop scheduling to subcontractor planning in the construction industry.

Their centralized work plan produces a workable schedule from a “workable backlog,” where they check and satisfy all constraints and, therefore, maintain the logical sequence of the work. When they considered resource availability, which is one of the constraints in their framework, they do not evaluate the impact of the work plan on project performance when making a schedule, nor do they present any mechanism for matching resource availability to resource requirements, like the resource-driven scheduling model (El-Rayes and Moselhi 1996). As a result, their method schedules activities in the order of the network schedule when resource availability matches with resource requirements. Lately, they have identified the necessity of interactive coordination of distributed work plans in order to better coordinate work (Choo and Tommelein 2000), but do not provide a conflict-resolution methodology for coordinating distributed work plans.

In my DCPSC framework, a subcontractor first makes its preferred schedule for its activities, then negotiates with other subcontractors. Therefore, in addition to their work, my DCPSC framework integrates individual subcontractor's preferred schedules into a project-wide workable schedule for socially rational decision-making. I only constrain resource availability in my DCPSC framework, but other constraints would also need agent-based negotiation if the schedule decision affects other subcontractors. My agent-based negotiation approach is an extension to their interactive coordination of distributed work plans for resolving conflicts in distributed work plans.

3.4.8 CAPACITY CONSTRAINTS

O'Brien and Fischer (2000) discuss the importance of capacity constraints to construction cost and schedule. Based on results from case studies, they conclude that capacity constraints affect the cost of subcontractors and suppliers. They also show that it is necessary to quantitatively model the relationship between capacity allocation and cost, but do not provide a quantitative cost model of capacity constraints. Earlier, O'Brien (1998) presented a centralized coordination model detailing the interactions between resource allocations and productivity for the activities of a subcontractor working on a particular project. His research provides a foundation to help subcontractors make decisions on how to allocate their resources across projects while subject to capacity constraints.

My DCPSC framework employs a distributed coordination paradigm, unlike their centralized coordination model. The reason for employing the distributed coordination paradigm is that consideration of each agent's resource constraints and extra costs will be beyond general contractors' capability in cases involving many agents in complex projects. Based on their finding of the importance of capacity constraints to construction cost and schedule, I have introduced an agent-based negotiation approach by which subcontractors can use the quantitative capacity constraints, i.e., extra cost information, for project schedule coordination in a distributed manner.

3.5 SUMMARY OF DCPSC FRAMEWORK BASED ON AGENT-BASED NEGOTIATION APPROACH

This chapter presented a new distributed coordination framework for project schedule changes (DCPSC) based on an agent-based negotiation approach wherein a project can be rescheduled dynamically by all of the concerned subcontractors with the help of software agents that evaluate the impact of changes, simulate decisions, and give advice. This chapter formalized the DCPSC framework and revealed three important issues: distributed coordination by competitive subcontractors, socially rational decision-making, and maintaining the logical sequence of the work. This chapter introduced an agent-based negotiation approach to overcome the difficulties stemming from these issues. This chapter reviewed previous work on coordination of project planning and scheduling and related in detail the DCPSC framework and agent-based negotiation approach to previous work, in terms of these issues.

CHAPTER 4

AGENT-BASED COMPENSATORY NEGOTIATION METHODOLOGY TO FACILITATE DISTRIBUTED COORDINATION OF PROJECT SCHEDULE CHANGES

This chapter presents a novel agent-based compensatory negotiation (ABCN) methodology to facilitate the distributed coordination of project schedule changes wherein a project can be rescheduled dynamically through negotiations by all of the concerned subcontractors. The methodology consists of a compensatory negotiation strategy based on utility which agents have, multi-linked negotiation protocols by which agents interact with other agents, and message-handling mechanisms for agents to evaluate alternatives and simulate the decision-making. This chapter introduces a new, simpler case example to illustrate the methodology. It also reviews previous work and states my contributions compared to the ABCN methodology.

4.1 INTRODUCTION

Subcontractors can reallocate their initially assigned resources whenever timing of the activities does not match well with the timing of available resources, which means that there are discrepancies between resource requirements and resource availabilities. However, this resource reallocation causes extra costs. When they try to change the timing of their activities instead of reallocating resources, the changes cause external costs to succeeding subcontractors. Therefore, subcontractors have to evaluate the extra costs associated with the reallocation of their resources and the external costs for changing the timing of activities, and then they can make decisions within the distributed coordination framework for the project schedule changes (DCPSC).

In the DCPSC framework, activities differ in costs so that extra costs of resource reallocation and external costs of changed timing vary greatly with activities. Therefore, without an explicit method for transferring utility units ("money"), they cannot find fair deals with other agents. Imagine that a subcontractor would need to pay one million dollars for failing to meet his/her schedule, i.e., the extra cost is one million dollars. If the subcontractor finds that it costs only ten thousand dollars for delaying succeeding activities, i.e., the external cost is ten thousand dollars, he/she will be happy to pay ten thousand dollars for the delay. This is an extreme case, but it shows how a subcontractor can transfer utility to other subcontractors for compensation of disadvantageous agreements. Also, such potential great disparities between overall costs make the split option (Sandholm, 1993) and coin tossing (Rosenschein and Zlotkin, 1994) inappropriate in the DCPSC framework and make compensatory negotiation the appropriate approach.

The problem of finding external costs in a distributed manner is a major challenge when the number of activities is huge. For example, a typical building project has several thousand activities to be rescheduled. Furthermore, the activities are tightly linked. No subcontractor has complete knowledge of the whole schedule and it is not feasible to send private information, such as resource and cost information, to one central coordinator**.** Therefore, I need new negotiation protocols, which subcontractors can use to receive cost responses before making decisions.

Since a coordination methodology is needed for subcontractors to interact with other subcontractors in the distributed coordination of project schedule changes, and because of the huge number of messages to be exchanged among subcontractors for negotiation processes, I adopted the agent-based approach to develop a novel coordination methodology. By adopting the agent-based approach, I represented the subcontractors in the distributed coordination of project schedule changes as agents, which exchange messages with other agents to evaluate changes to simulate negotiation processes. Therefore, agents need message-handling mechanisms.

In this chapter, I formalize three main aspects of the agent-based compensatory negotiation methodology: (1) the compensatory negotiation strategy based on utility to the agents; (2) the multi-linked negotiation protocols by which agents interact with other agents; and (3) message-handling mechanisms for agents to evaluate alternatives and simulate the decision-making.

4.2 COMPENSATORY NEGOTIATION STRATEGY BASED ON UTILITY

Each agent calculates utility of timing to evaluate the impacts of its schedule changes and to compensate other agents for disadvantageous agreements through a utility transfer scheme. My utility transfer scheme differs from the monetary transfer schemes developed in market-based systems, where agents transfer money in return for goods or services. In my research, utility captures the value of timing, which market-based systems have not considered as a transferable good or service. Because of the difficulty of capturing the value of timing, they assumed that the timing is a risk that all agents should bear. The representation of the utility of timing as "transferable money" is one of my key contributions.

4.2.1 UTILITY OF TIMING

In my research, I adopted the definition of utility by Rosenschein and Zlotkin (1998) to quantify utility of timing[3]:

> The utility of a deal for an agent is defined as the cost of its original work minus the cost of its new work given the deal. The difference is how much it has gained from the deal (p. 361).

Therefore, I represent utility of timing of as a real-valued number ("money"), which describes the difference between the cost of the initial schedule and the costs of alternatives for the activity as units of money based on resource utilization. A reasonable

[3] In this dissertation, timing represents timing of the work to be performed. I represent timing of work as a tuple of (start date, end date) of the work, but it is different from the duration of the work.

assumption about resource utilization is that any discrepancy between the resource requirements and the available resources causes a subcontractor to incur extra costs of either over-utilizing current resources, e.g., paying overtime, or importing new resources. These extra costs motivate the search for a better solution. The utility units are common for all agents, and agents can transfer their utility units to other agents for compensation.

Each agent uses the following utility function for each activity *k*

$$Utility_k = AC_k - \sum_{x \in all\_suceeding\_activities_k} DC_x$$

Where AC_k is the extra "acceleration cost" for accelerating the k^{th} activity; DC_x is the extra "delay cost" for delaying the succeeding activity *x*.

In my research, I consider only direct costs of the activity and liquidated damage for project delays and ignore other overhead costs or indirect costs. Since the liquidated damage for project delays is included in the calculation of *AC* and *DC*, the overhead costs or indirect costs can be treated as the same way.

Note that the agent, which has activity k, knows AC_k, but does not know the summation of DC_x until getting *DC*s from the succeeding activities. I discuss the methods for the agent to get *DC*s from the succeeding activities in Section 4.3. The following two sections explain the Acceleration Cost (AC_k) and the Delay Cost (DC_x) in detail.

4.2.1.1 Acceleration Cost

If an agent cannot meet its schedule with its available resources, the agent has two choices. One choice is to complete the activity with the same schedule and a higher cost per day. It might incur extra costs to accelerate the k^{th} activity, in addition to the original cost (C_{k0}). The second choice is to extend the schedule with a lower cost per day. There might also be extra costs to extend the k^{th} activity. Therefore, agents have to consider two kinds of costs for the k^{th} activity to calculate the acceleration cost (AC_k): C_{k1}, the total cost for the k^{th} activity with the same schedule and a higher cost per day; and C_{k2}, the total cost for the k^{th} activity with an extended schedule and a lower cost per day, as shown in Figure 4-1.

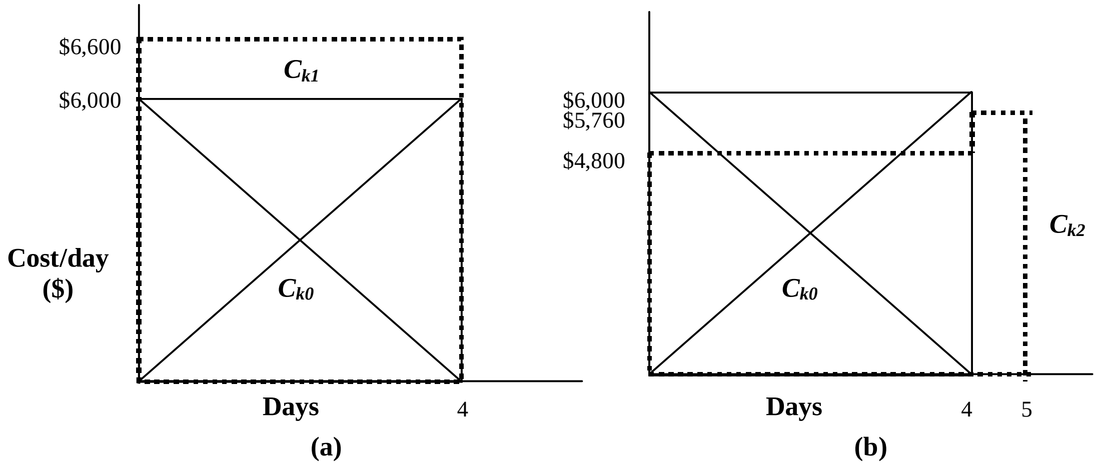


Figure 4-1. Two kinds of acceleration costs: (a) C_{k1} ; (b) C_{k2}

Then, I calculate AC_k as follows:

If C_{k1} is bigger than C_{k2}:

$AC_k = C_{k1} - C_{k2}$

Else:

$AC_k = 0$

Note that C_{k2} is not always equal to C_{k0}, even though the same numbers of resource-days are used in the original and extended schedule. There are many reasons why the cost per resource-day would be different day by day, such as bad weather, increase of labor costs after collective bargaining, or delays of scheduled move-out, as shown in the example below.

Example-1: Suppose an activity *k* needs 10 resources for 4 days (from Day 1 to Day 4), which are expected to be $600 per resource-day at bid, but its agent has only 8 resources at execution. Then the agent either has to work overtime, which would cost $900 per resource-day for 8-resource-day shortages (assuming the overtime rate is 150%), or delay the work by one extra day while spending $720 per resource-day (assuming the extra rate is 120%). If the delay incurs liquidated damages, the agent should include the liquidated damages into the extra costs.

In this case, C_{k0} is {10 resources*4 days*$600/(resource-day)} = $24,000;

C_{k1} is {{8 resources*4 days*$600/(resource-day)} + {8 resource-day * $900/(resource-day)} = $26,400;

C_{k2} is {(8 resources*4 days*$600/(resource-day)} + {8 resources*1 day*$720/(resource-day)} = $24,960.

Since C_{k1} is bigger than C_{k2}, $AC_k = C_{k1} – C_{k2}$ = $26,400 - $24,960 = $1,440, this means that the activity *K* can save $1,440 if the activity would be delayed by one extra day.

4.2.1.2 Delay Cost

If an agent has to change its schedule due to the delays of preceding activities, the agent has two choices. One choice is to do the work with the shorter duration and a higher cost per day. The agent might incur extra cost to accelerate the x^{th} activity, in addition to the original cost (C_{x0}). The second choice is to do the work with the longer duration and a lower cost per day. There might be extra costs to extend the x^{th} activity. Therefore, two kinds of costs for the x^{th} activity are considered to calculate a “delay cost” (*DC*): C_{x3}, the total cost for the x^{th} activity with the shorter duration and a higher cost per day; and C_{x4} , the total cost for the x^{th} activity with the longer duration and a lower cost per day, as shown in Figure 4-2.

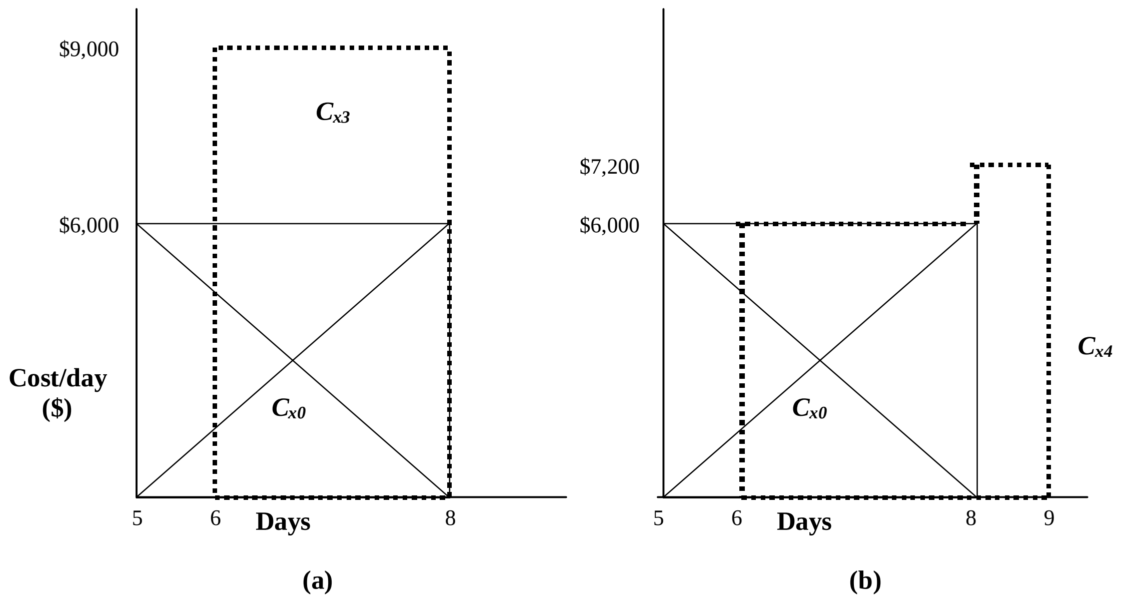


Figure 4-2. Two kinds of delay costs: (a) C_{x3} ; (b) C_{x4}

Then, I calculate DC_x as follows:

If the start of the activity is delayed:

$$DC_x = \min\left(\left(C_{x3} - C_{x0}\right), \left(\left(C_{x4} - C_{x0}\right) + \sum_{y \in all\_suc\_activity_x} DC_y \right) \right)$$

$$Else : DC_x = 0$$

Example-2: Suppose activity x also needs 10 resources for 4 days (from Day 5 to Day 8) at $400 per resource-day. Also assume the start of activity x is delayed due to the preceding activity. Then the agent has to work overtime to complete the activity on time, which would cost $600 per resource-day for 10 resource-day shortages (assuming the overtime rate is 150%), or delay the work by one extra day while bringing in 10 resource-day shortages at $480 per resource-day (assuming the import rate is 120%).

In this case, C_{x0} is {10 resources*4 days*$400/(resource-day)} = $16,000;
C_{x3} is {{10 resources*3 days*$400/(resource-day)} + {10 resource-days * $600/(resource-day)}} = $18,000;

C_{X4} is {{10 resources*3 days*$400/(resource-day)} + {10 resource-days * $480/(resource-day)}} = $16,800.

Since *DCx* cannot be calculated without considering other *DC*s from succeeding activities, let us assume that the activity is the last activity that has no succeeding activity and has no liquidated damages. Since the activity is delayed, *DCx* = min $((C_{X3} - C_{X0})$, $((C_{X4} - C_{X0}))$
= min (($18,000 - $16,000), ($16,800 - $16,000))
= min ($2,000, $800) = $800. This means that the activity *x* cost $800 more if the activity is delayed by one extra day.

4.2.1.3 Schedule-Change Options

Agents can calculate the acceleration cost (*AC*) and the delay cost (*DC*) by using given rates, such as overtime rates, extra rates, or import rates, as shown in the two previous examples. However, the assumptions vary by the parameters for characteristics of resources, such as units, fixed/variable, timing, and upper limits. Therefore, instead of calculating *AC* and *DC* based on the parameters, which cannot cover all the different situations that should be considered, the software agents are given schedule-change options by their "clients" — subcontractors. Agents can calculate the utility of timing for their activities from the given schedule-change options.

I represent a schedule-change option as a tuple of the form[4]:

(*startDate endDate extraCost*)

The *startDate* is the possible start date of the activity. The *endDate* is the possible finish date of the activity. The *extraCost* is the extra cost associated with the timing of the activity. An activity has one or more schedule-change options.

[4] The representation of schedule-change options looks very much like the time-cost trade-off (TCT) formulation, but it represents the timing of the activity, which TCT ignores, since TCT assumes cost is invariant with the temporal position of the activity (O'Brien and Fischer 2000, pp. 367-368).

For Example 1, the schedule-change options for activity *k* are {(1 4 \$2,400)(1 5 \$960)}. The schedule-change options mean that the activity *k* costs \$2,400 more when it starts at Day 1 and finish at Day 4, but it costs \$960 more when it delays the finish to Day 5. Then the acceleration cost for the activity k for going from Day 1 to Day 4 (AC_k) = (\$2,400 - \$960) = \$1,440. In other words, the activity can save \$1,440 if it delays the finish by one day.

For Example 2, the schedule-change options for activity *x* are {(5 8 \$0)(6 8 \$2,000)(6 9 \$800)}. The schedule-change options mean that the activity *x* costs \$0 more when it starts on Day 5 and finishes on Day 8, but it costs \$2,000 more when it delays the start by one day, but finishes on Day 8. The third option means it costs \$800 more when it delays the start and finish by one day with the same duration. When the start of activity *x* is delayed by one day, the possible schedule-change options are {(6 8 \$2,000)(6 9 \$800)}. Then the delay cost (DC_x) = min {\$2,000, \$800}= \$800 if the activity *x* is the last activity.

4.2.2 UTILITY TRANSFER FOR COMPENSATORY NEGOTIATION

The key concept of the compensatory negotiation strategy is to transfer utility ("money") to compensate agents for "playing along" in a situation that, to them, is otherwise locally suboptimal. After getting *DC* through negotiation, if *AC* is more than *DC*, i.e., there is positive utility, the agent decides to make an extension, and transfers the DC portion of the utility to other agents for compensation of disadvantageous agreements.

The compensatory negotiation consists of inner and outer cycles: Inner cycles[5] are used for an activity to get *DC*s from succeeding activities, i.e., forward and backward. One outer cycle[6] is used when an activity finishes its negotiation through the inner cycles. The direction of the outer cycle is only forward, i.e., from the start activity to the end activity on the project schedule, while each activity negotiates using inner cycles. Since schedule-change options are defined on the basis of activity, the negotiation is based on activity.

[5] The state of inner cycle is tracked by agents updating the "flag" state on their activities as further discussed in Section 4.3.1

[6] The state of outer cycle is tracked by agents updating the "active" state on their activities as further discussed in Section 4.3.1

Therefore, agents, which have multiple activities, could experience many cycles of negotiations.

Agents will change schedule-change options during the negotiation process to reflect compensations among agents. For instance, agent-A can change the schedule-change options for activity *k* from {(1 4 $2,400)(1 5 $960)} to {(1 4 $2,400)(1 5 $1,760)}, if agentA compensates agentB for delaying the start of activity *x* at the cost of $800. This will cause agentB to change its schedule-change options from {(5 8 $0)(6 8 $2,000)(6 9 $800)} to {(5 8 $0)(6 8 $2,000)(6 9 $0)}. After compensating Agent-B's loss, Agent-A still can save $2,400 - $1,760 = $640 and Agent-B has no loss at all.

Except the aforementioned changes by agents, I exclude situations where the human subcontractors change their schedule-change options while their agents are engaging in the negotiation process, although it is quite common in real-world negotiation situations. In the compensatory negotiation, agents evaluate the impacts of changes and simulate negotiation based on the given schedule-change options. Therefore, updating change options throughout the negotiation process cannot ensure consistency of the negotiation.

After agents finish all negotiation processes based on the given schedule-change options, human subcontractors can update their schedule-change options. As it is unknown to various subcontractors what kinds of delays they might have to respond to, the schedule-change options are progressive, which means subcontractors update their schedule-change options when they notice a delay.

4.3 MULTI-LINKED NEGOTIATION PROTOCOLS

Negotiation protocols govern the interaction among agents by constraining the way the agents interact. In this research, agents need negotiation protocols to get *DC*s from succeeding agents and to transfer utility for compensation of disadvantageous agreements.

Compensatory negotiation starts with a baseline project schedule and has an agent propose to compensate other agents for costs imposed on the latter for the former's change of the plan. This is simple when agents can reschedule their activities without affecting others or when the counterpart agent is one, which is the case of pairwise negotiation. In a more complicated case, an agent needs to negotiate with another agent, which in turn needs to negotiate with a third, and so on, until the last agent. I call it “multi-linked” negotiations. This multi-linked negotiation is inspired by the work of Neiman and others (1994), but it is acyclic and therefore more straightforward than their protocol.

Multi-linked negotiation protocols are needed because of the tightly coupled nature of construction project schedules. My multi-linked negotiation differs from multilateral negotiation (auction) protocols because multi-linked negotiation allows agents to negotiate with other agents within precedence relationships rather than restricting them to negotiate solely with an auctioneer.

The negotiation protocols provide the performatives, which are shared primitive message types for agents to use in negotiation (Finin et al. 1994), and conversation sequence, which shows the structured message exchanges between agents while changing states. The next two sections formalize the performatives and conversation sequence for the multi-linked negotiation protocols.

4.3.1 MULTI-LINKED NEGOTIATION PERFORMATIVES

My multi-linked negotiation performatives fall into three classes: *human interaction*, *negotiation*, and *negotiation control*. The *human interaction* performatives allow a human subcontractor to provide input data to its agent and an agent to inform its subcontractor of the current status of negotiation. The *negotiation* performatives facilitate the actual compensatory negotiation processes. The *negotiation control* performatives manage the states of negotiation processes. For my multi-linked negotiation protocol, performatives should be recursive because of the tightly coupled network-like precedence relationships, as shown in Table 4-1.

Performative	Start/Active/Flag	Middle activity(s)	End activity
▼ input	sorry ▲	sorry ▲	sorry ▲
▼ ◀ ready	ask-cost|hand-over ▶	ready ◀	final ▲
▶ ask-cost		ask-cost ▶	reply-cost ◀
◀ reply-cost	accept-all|reject-all ▶	accept|reject ▶	
▶ accept		accept|reject ▶	confirm|renege ◀
▶ reject		reject ▶	renege ◀
◀ confirm	reply-cost ◀	confirm ◀	
◀ renege	reply-cost ◀	renege ◀	
▶ accept-all		accept-all|reject-all ▶	confirm-all|renege-
▶ reject-all		reject-all ▶	renege-all ◀
◀ confirm-all	hand-over ▶ ◀	confirm-all ◀	
◀ renege-all	hand-over ▶	renege-all ◀	
▶ hand-over		ask-cost|hand-over ▶	done ◀ & final ▲
◀ done	final ▲	done ◀ & final ▲	

Legend:

▼ Performative	Receiving a message from its subcontractor
◀ Performative	Receiving a message from its succeeding activity
▶ Performative	Receiving a message from its preceding activity
Performative ▲	Sending a message to its subcontractor
Performative ▶	Sending a message to its succeeding activity
Performative ◀	Sending a message to its preceding activity
Performative ▶ ◀	Sending a message to its own agent

Table 4-1. Multi-linked negotiation performatives and expected responses

Note that agents choose the performative based on the position in the project schedule, such as start, middle, or end, and the state of negotiation, such as active or flag (see Section 4.3.1.2). This section addresses the semantics of the performatives. Appendix-B shows details of the syntax of the performatives.

4.3.1.1 Human Interaction Performatives

These performatives are formal interfaces needed for a human subcontractor to provide input data to its agent and for an agent to inform the subcontractor of the results of negotiation. The message senders are human subcontractors or agents. The receivers are agents or human subcontractors.

input

This performative allows a subcontractor to provide its agent with agent information and activity information, including precedence and schedule-change options.

ready

This performative allows a subcontractor to inform its agent that input is finished on the specific activity.

final

This performative allows an agent to inform the subcontractor of the final result of negotiation on the specific activity.

4.3.1.2 Negotiation Performatives

The following performatives are used for facilitating compensatory negotiation processes. The senders and receivers are agents[7].

Since distributed agents have no knowledge about the whole state of negotiation, I introduce two states of negotiations marked on activities to promote structured communication and ensure consistency among agents during negotiation. One is the "active" state, which means the activity is "active" in initiating negotiation. First, the start activity becomes "active" and other activities become "active" when the preceding activity passes control. When an activity finishes an inner cycle of negotiation, the activity changes the state from "active" to NULL. The second is the "flag" state, which

[7] For simplicity of explaining performatives, I assume that each agent has one activity and the schedule is linear. Therefore, an activity means an agent in this section. However, the performatives can be used for multiple activities for an agent and multiple succeeding activities for an activity.

means the activity is not “active”, but calculates *DC* based on the cost reply from the succeeding activity. An activity becomes “flag” when the preceding activity sends an “ask-cost” message and changes the activity’s state from “flag” to NULL after sending a “reply-cost” message.

ask-cost

This performative allows an activity to ask the succeeding activity to find out any cost which is incurred by the delay of the proposed start date. This performative makes the receiving activity “flag.”

reply-cost

This performative allows an activity to reply to the preceding activity with the cost, which is incurred by the delay. Note that the cost can be its own cost or a sum of succeeding activities’ cost so that no agent will be able to figure out how much a delay costs for a particular agent.

accept

This performative allows the “flag” activity to accept the cost response from the succeeding activity. This means that the activity chooses to delay rather than accelerate.

reject

This performative allows the “flag” activity to reject the cost response from the succeeding activity. This means that the activity chooses to accelerate rather than delay.

confirm

This performative allows an activity to confirm the “accept” from the preceding activity. This means that its cost reply is accepted, but the final contract is pending. This performative makes the receiving activity change the state from “flag” to NULL if the receiving activity is the “flag” activity.

renege

This performative allows an activity to renege the “accept” or “reject” from the preceding activity. This means that its cost reply is rejected and the activity has to keep the original schedule. This performative makes the receiving activity change the state from “flag” to NULL if the receiving activity is the “flag” activity.

accept-all

This performative allows the “active” activity to accept the cost reply from the succeeding activity. This means that the activity chooses to delay rather than accelerate.

reject-all

This performative allows the “active” activity to reject the cost reply from the succeeding activity. This means that the activity chooses to accelerate rather than delay.

confirm-all

This performative allows an activity to confirm the “accept-all” from the preceding activity. This means that its cost reply is accepted and the contract is binding.

renege-all

This performative allows an activity to renege the “accept-all” or “reject-all” from the preceding activity. This means that the activity has to keep the original schedule.

4.3.1.3 Negotiation Control Performatives

These performatives are used for managing the states of negotiation processes. The senders and receivers are agents.

ready

This performative allows an activity to inform the preceding activity that the activity is ready for negotiation. Note that the “ready” performative between activities differs from the “ready” performative between a human and its agent. This performative makes the start activity “active.”

hand-over

This performative allows the "active" activity to pass control to the succeeding activity, informing it of starting a negotiation. This performative makes the receiving activity "active."

done

This performative allows an activity to inform the preceding activity that the activity has finished the negotiation. This performative makes the receiving activity change its state from "active" to NULL.

4.3.1.4 Comparison to Pair-Wise Negotiation Performatives

In this section, I compare my negotiation performatives to the "pair-wise negotiation performatives" developed by Chen, et al. (1999). The "pair-wise negotiation performatives" consist of *CFP* (*Call For Proposal*), *proposal*, *accept-proposal*, *reject-proposal*, and *terminate*, as shown in Table 4-2.

CFP	the action of calling for a proposal to perform a given action
proposal	the action of submitting a proposal to perform a certain action, given certain preconditions
accept-proposal	the action of accepting a previously submitted proposal to perform an action
reject-proposal	the action of rejecting a previously submitted proposal to perform an action
terminate	the action to finish the negotiation process

Table 4-2. Pair-wise negotiation performatives (excerpt from Chen et al (1999))

The "pair-wise negotiation performatives" are similar to the "announcing-bidding-awarding" processes in the Contract Net Protocol (CNP) (Smith 1980; Smith and Davis 1981; Davis and Smith 1988). In fact, CNP is heavily adopted for many "pair-wise" or "multi-lateral (auction)" negotiation protocols (Malone et al. 1988; Sandholm 1993; Sandholm and Lesser 1995; Sen and Durfee 1996). The CNP is a very general protocol for distributing activities in a network of agents. However, the "awarding" process requires contract binding. In my research, where multiple agents concurrently try to reach consensus agreements, I need another performative, “confirm/renege,” for binding contracts.

Suppose there are three agents who are trying to solve schedule conflicts. Agents A and B should precede AgentC, but have no relationship. Therefore, AgentC needs two ”accept-proposal” messages from Agents A and B to resolve schedule conflicts after sending “proposal” messages to Agents A and B according to their CFPs. In a case where AgentA sends an “accept-proposal” message to AgentC, but AgentB sends a “reject-proposal” message to AgentC, the “awarding” stage should not automatically bind the contract, as shown in Figure 4-3. Otherwise, the contract cannot ensure the consistency of the project schedule.

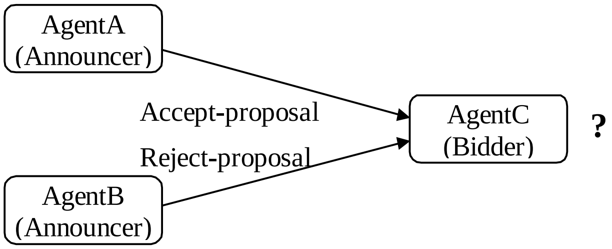


Figure 4-3. “Awarding” stage of negotiation

I must allow AgentC to use another performative, “confirm/renege,” to inform Agents A and B of the inconsistent messages. AgentC uses the “confirm” performative for contract binding when all the messages are “accept-proposal” and uses the “renege” performative otherwise. In this example, AgentC sends the “renege” message, which means the “awarding” is not valid, as shown in Figure 4-4. Agents A and B need to start a new negotiation.

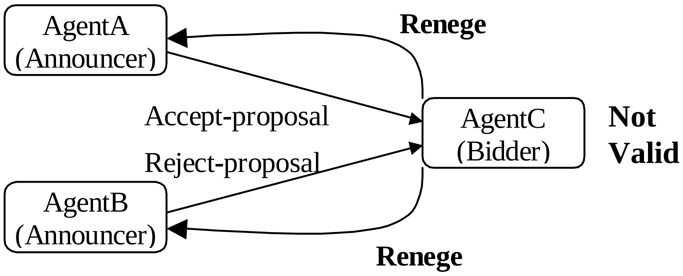


Figure 4-4. “Confirming” stage of negotiation

As shown in the above simple example, the new performatives, “confirm/renege,” are necessary where multiple agents are concurrently trying to reach mutual agreements in the new “confirming” stage. This is the addition to CNP.

4.3.2 CONVERSATION SEQUENCE OF MULTI-LINKED NEGOTIATION

Conversation sequence shows the structured message exchanges between agents based on states. It is important to note that the project activity precedence relationships govern the agent message exchange. Since I developed the multi-linked negotiation protocols for the distributed coordination of project schedule changes, agents should exchange messages according to the project schedule. I can guarantee that my protocols will converge, which means my protocols will not enter infinite loops of refinement, because agents use static activity precedence relationships in the project schedule for message passing among agents. The following two sections describe the conversation sequence for the multi-linked negotiation and compare it to the conversation sequence of pair-wise negotiation.

4.3.2.1 Conversation Sequence Diagram

A conversation sequence diagram that includes the multi-linked negotiation performatives in Section 4.3.1 represents my negotiation process. For my multi-linked negotiations, loops represent recursive sequences of conversation, as shown in Figure 4-5.

Since agents should exchange messages according to the project schedule, the multi-linked negotiation process is synchronous, which means agents are allowed to work only on replanning a task under the control of a single “active” agent. Neiman and Lesser (1996) assert that a synchronous negotiation process is superior to an asynchronous process in their cooperative schedule repair method, in which all agents suspend their current problem-solving activities and collaboratively search for some reassignment of resources that would allow the current scheduling goal to be satisfied without a constraint relaxation.

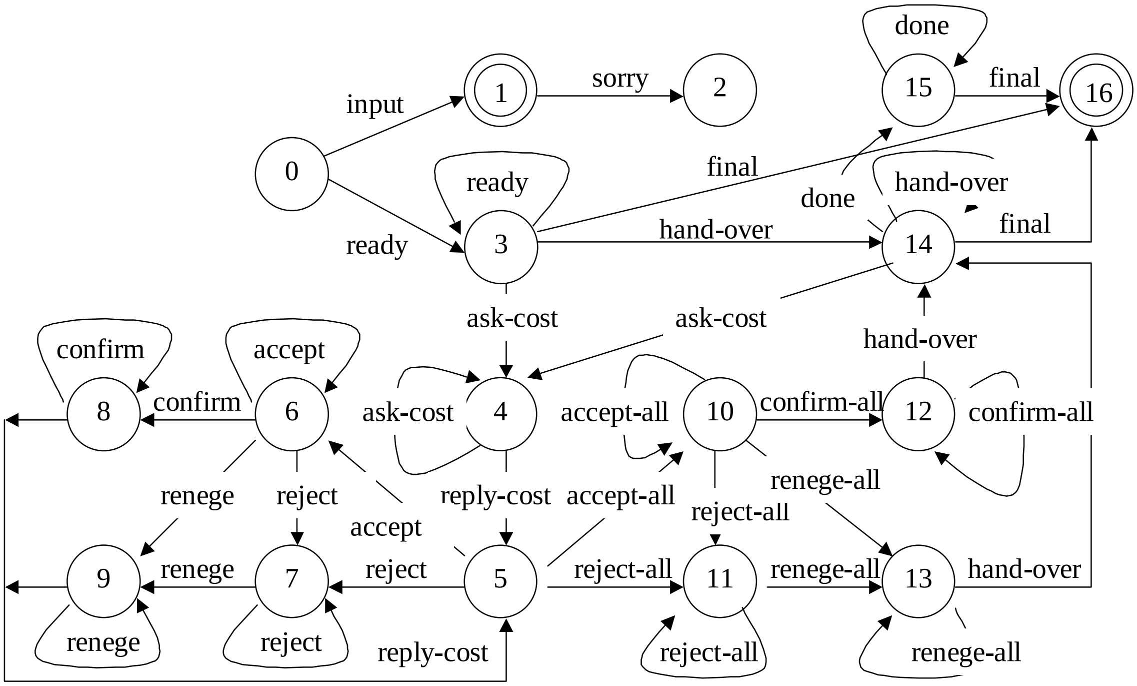


Figure 4-5. Conversation sequence diagram of multi-linked negotiation protocols

4.3.2.2 Comparison to Conversation Sequence Diagram of Pair-Wise Negotiation

In this section, I compare my conversation sequence diagram of multi-linked negotiation performatives to the conversation sequence diagram of the "pair-wise negotiation" developed by Mudgal and Vassileva (1999).

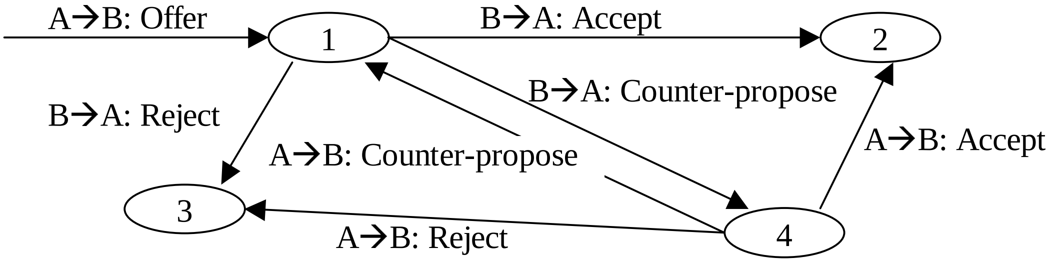


Figure 4-6. Conversation sequence diagram of pair-wise negotiation
(Mudgal and Vassileva, 1999)

The conversation sequence diagram in Figure 4-6 represents possible sequences of conversation that can occur during pair-wise negotiations. Even though their conversation sequence is richer than mine in terms of allowing repetitive negotiation using “counter-propose” messages, their conversation sequence diagram is missing the recursive sequences of conversation because they only consider two agents. The recursive sequences of conversation are necessary when multiple agents engage in negotiations in a tightly coupled schedule.

4.3.3 EXAMPLE OF MULTI-LINKED NEGOTIATION

This section illustrates the multi-linked negotiation protocols. This simple example consists of three agents, in which each has one activity, as shown in Figure 4-7.

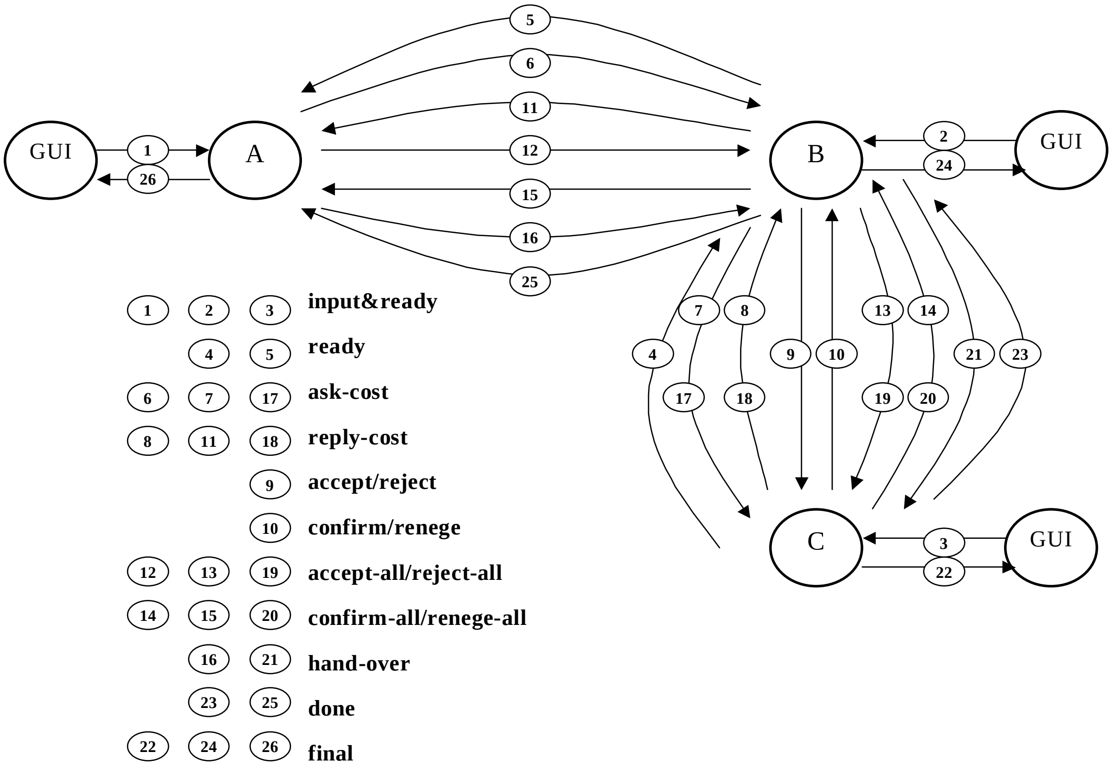


Figure 4-7. Example conversation sequence diagram

A negotiation starts when the subcontractors send “input” and “ready” messages (#1~3) to their respective agents. Then agents send “ready” messages (#4~5) to agents backward up to the ‘start’ agent, which has the start activity.

Upon receiving a “ready” message, the ‘start’ agent sends an “ask-cost” message (#6) forward to the ‘middle’ agent, which will also send an “ask-cost” message (#7) forward to the ‘end’ agent. Then the ‘end’ agent sends a “reply-cost” message (#8) backward to the ‘middle’ agent, which will send an “accept” or a “reject” message (#9) back to the ‘end’ agent. Then the ‘end’ agent sends a “confirm” or a “renege” message (#10) backward to the ‘middle’ agent, which will send a “reply-cost” message (#11) backward to the ‘start’ agent.

The ‘start’ agent sends an “accept-all” or a “reject-all” message (#12) forward to the ‘middle’ agent, which will also send an “accept-all” or a “reject-all” message (#13) forward to the ‘end’ agent. The ‘end’ agent will send a “confirm-all” or a “renege-all” message (#14~15) backward up to the ‘start’ agent, which will send a “hand-over” message (#16) forward to the ‘middle’ agent.

The conversation sequence will repeat (#17~20) until the ‘middle’ agent sends a “hand-over” message (#21) to the ‘end’ agent. A cycle of negotiation finishes when the ‘end’ agent receives a “done” message (#25) and sends a “final” message (#26) to its subcontractor.

4.4 MESSAGE-HANDLING MECHANISMS

The agent reacts according to what message it gets. Therefore, the agent should have the functionality of handling messages for each type of multi-linked message protocol. When the agent handles a message, it should also make a decision accordingly. Table 4-3 summarizes the message handling mechanisms. Appendix-C shows details of the message handling mechanisms.

Incoming message	Message handling mechanisms
1. “input”	All activities: • If AGENT status is ‘lock,’ send a “sorry” message • Else parse and store AGENT or ACTIVITY information
2. “ready”	All activities: • Check if all “ready” messages are received • If received all, update its AGENT status as ‘lock’ • Else wait Start activity: • Updates ACTIVITY status as ‘active’ • Selects AC • If it is the ‘end’ activity, sends a “final” message • Else if AC = 0, sends “hand-over” messages • Else sends “ask-cost” messages Middle activity(ies): • Forward “ready” messages
3. “ask-cost”	All activities: • Update critical activities • Check if all “ask-cost” messages are received • If received all, select C4 • Else wait Middle activity(ies): • Update ACTIVITY status as ‘flag’ • Forward “ask-cost” messages End activity: • Sends “reply-cost” messages according to criticality
4. “reply-cost”	All activities: • Accumulate DC • Check if all “reply-cost” messages are received • If not, wait Flag activity(ies): • Select C3 • If C3 > DC, send “reject” messages • Else send “accept” messages Active activity: • Compares AC and DC • If AC >= DC, sends “accept-all” messages • Else sends “reject-all” messages

Table 4-3. Summary of message handling mechanisms

Incoming message	Message handling mechanisms
5. “accept”	All activities: • Update ACTIVITY check as ‘accept’ • Check if all “accept” or “reject” messages are received • If not, wait Middle activity(ies): • If any ACTIVITY check is ‘reject,’ send “reject” messages • Else send “accept” messages End activity: • If any ACTIVITY check is ‘reject,’ sends “renege” messages • Else sends “confirm” messages
6. “reject”	All activities: • Update ACTIVITY check as ‘reject’ • Check if all “accept” or “reject” messages are received • If not, wait Middle activity(ies): • Forward “reject” messages End activity: • Sends “renege” messages
7. “confirm”	All activities: • Check if all “confirm” messages are received • If not, wait Middle activity(ies): • Forward “confirm” messages Flag activity(ies): • Update ACTIVITY status as ‘null’ • Send “reply-cost” messages according to criticality
8. “renege”	All activities: • Check if all “renege” messages are received • If not, wait Middle activity(ies): • Forward “renege” messages Flag activity(ies): • Update ACTIVITY status as ‘null’ • Send “reply-cost” messages according to criticality
9. “accept-all”	All activities: • Update ACTIVITY check as ‘accept-all’ • Check if all “accept-all” or “reject-all” messages are received • If not, wait Middle activity(ies): • If any ACTIVITY check is ‘reject-all,’ send “reject-all” messages • Else send “accept-all” messages End activity(ies): • If any ACTIVITY check is ‘reject-all,’ send “renege-all” messages • Else send “confirm-all” messages

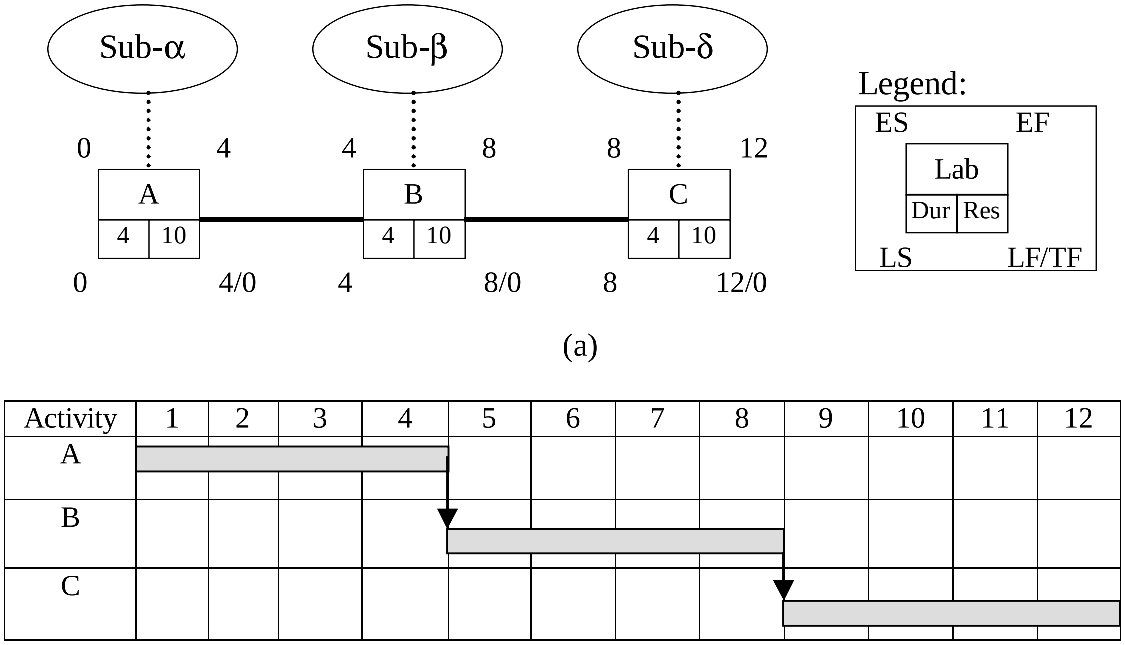
Table 4-3. Summary of message handling mechanisms (Continued)

Incoming message	Message handling mechanisms
10. “reject-all”	All activities: • Update ACTIVITY check as ‘reject-all’ • Check if all “accept-all” or “reject-all” messages are received • If not, wait Middle activity(ies): • Forward “reject-all” messages End activity: • Sends “renege-all” messages
11. “confirm-all”	All activities: • Check if all “confirm-all” messages are received • If not, wait Middle activity(ies): • Forward “confirm-all” messages Active activity(ies): • Update ACTIVITY status as ‘null’ • Send “hand-over” messages
12. “renege-all”	All activities: • Check if all “renege-all” messages are received • If not, wait Middle activity(ies): • Forward “renege-all” messages Flag activity(ies): • Update ACTIVITY status as ‘null’ • Send “hand-over” messages according to criticality
13. “hand-over”	All activities: • Check if all “hand-over” messages are received • If received all, update ACTIVITY status as ‘active’ • Else wait Middle activity(ies): • Select AC • If AC = 0, send “hand-over” messages • Else send “ask-cost” messages End activity: • Sends a “final” • Sends “done” messages
14. “done”	All activities: • Check if all “done” messages are received • If received all, update AGENT status as ‘unlock’ • Else wait Middle activity(ies): • Forward “done” messages • Send a “final” message Start activity: • Sends a “final” message

Table 4-3. Summary of message handling mechanisms (Continued)

4.5 A SIMPLE CASE EXAMPLE OF AGENT-BASED COMPENSATORY NEGOTIATION

This section will demonstrate the agent-based compensatory negotiation methodology using a simple case example. Consider the example network[8] shown in Figure 4-8(a). The results of conventional CPM calculations appear directly on the diagram. For simplicity, assume that each activity, which was subcontracted to one of three subcontractors, requires just one type of resource. The resource requirement for each activity appears on the diagram in Figure 4-8(a). Assume each subcontractor uses the same resource for its activities.



(a)

(b)

Figure 4-8. Example schedule: (a) CPM network schedule; (b) Gantt chart schedule

Assume that the subcontractors predicted at the time of bidding that their activities have sufficient resources available to support the initial schedule, as shown in Figure 4-9. The

[8] It is too complex to show the detailed agent-based compensatory negotiation steps using the case example in Section 1.3. Therefore, I use a simpler example here.

dotted rectangles indicate the initial resource requirements, based on the above schedule, for completion of the activities. Therefore, the above schedule will be feasible. However, as the actual execution dates approaches, the resource availability has become tighter under changing market conditions. Assume that each subcontractor has revised the available resource profile before the actual execution date, as shown in Figure 4-9. The shaded boxes indicate the new resource availability for each activity.

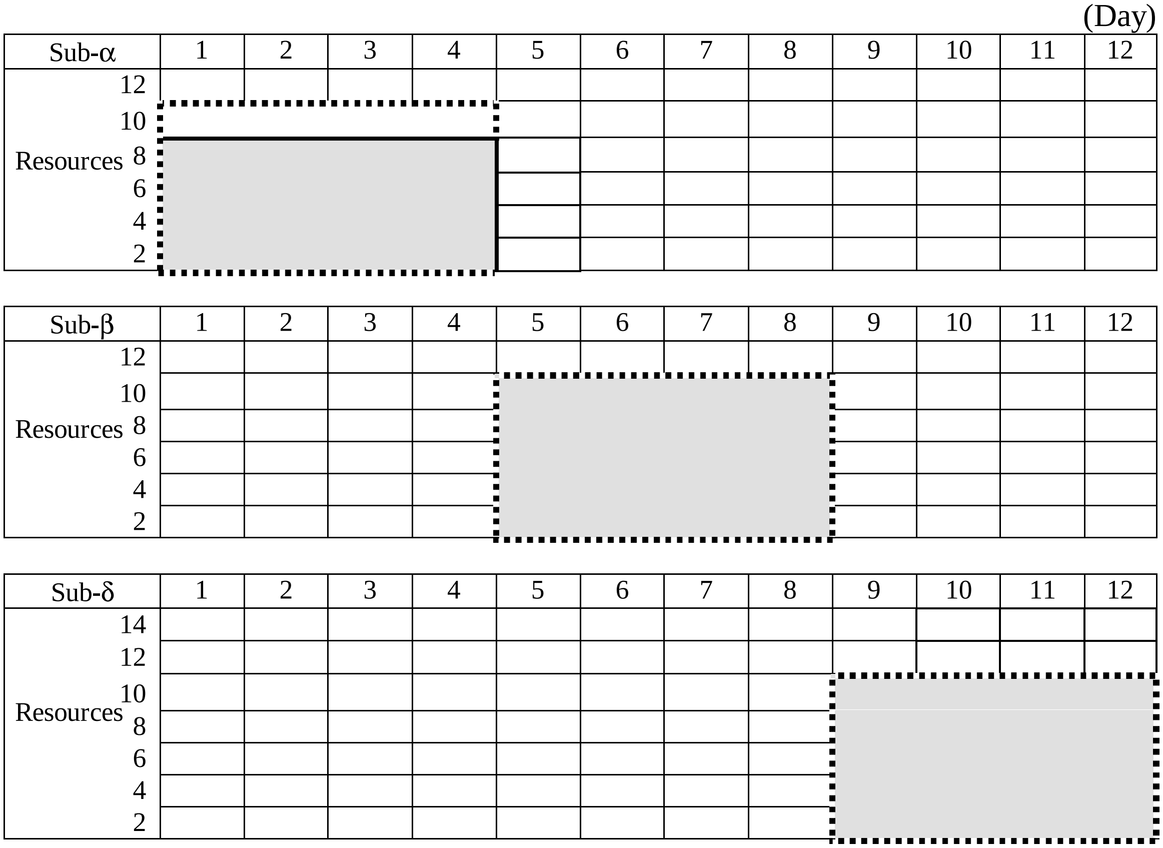


Figure 4-9. Resource histograms of Sub-α, Sub-β, and Sub-δ

The above resource histogram implies that some subcontractors have different preferred schedules than the original schedule. For instance, Sub-α wants to finish Activity-A on Day 5 since Sub-α does not have enough resources to finish on Day 4. Based on the above resource histograms, the diagonally hatched bars in Figure 4-10 summarize the subcontractors’ preferred schedules. Note that Activity-A and Activity-B have a schedule conflict on Day 5.

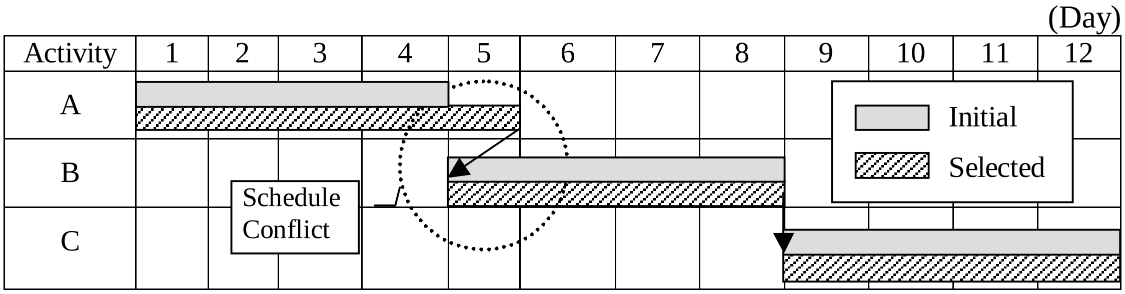


Figure 4-10. Preferred schedule

Based on the resource histogram, assume that each subcontractor prepares schedule-change options for its activity respectively, as shown in Table 4-4. Schedule-change options of activities A and B remain the same as the earlier examples and schedule-change options of activity-C are constructed to demonstrate the multi-linked negotiations. The subcontractors provide their agents with these predefined schedule-change options.

(startDate endDate extraCost)

Activity	ES-LF	Option-1	Option-2	Option-3
A	(1 4)	(1 4 2400)	(1 5 960)	
B	(5 8)	(5 8 0)	(6 8 2000)	(6 9 800)
C	(9 12)	(9 12 0)	(10 12 200)	(11 13 2000)

Table 4-4. Schedule change options of Sub-α, Sub-β, and Sub-δ

The final schedule appears in Figure 4-11, and Figure 4-12 shows the conversation sequences during the negotiation.

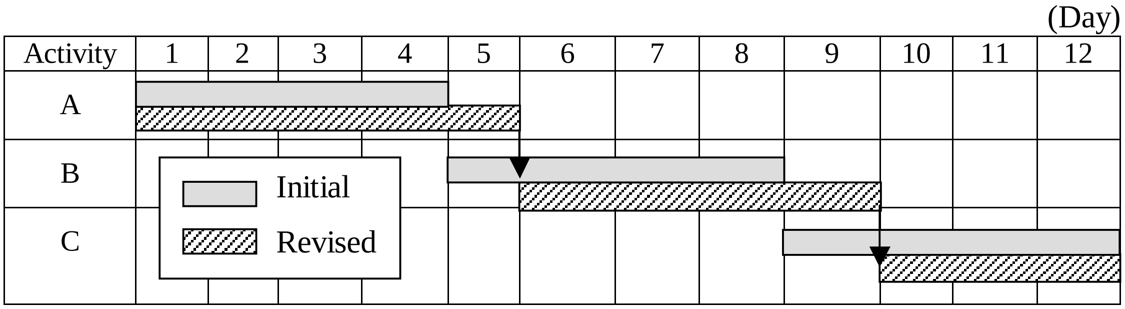


Figure 4-11. Revised schedule after negotiation

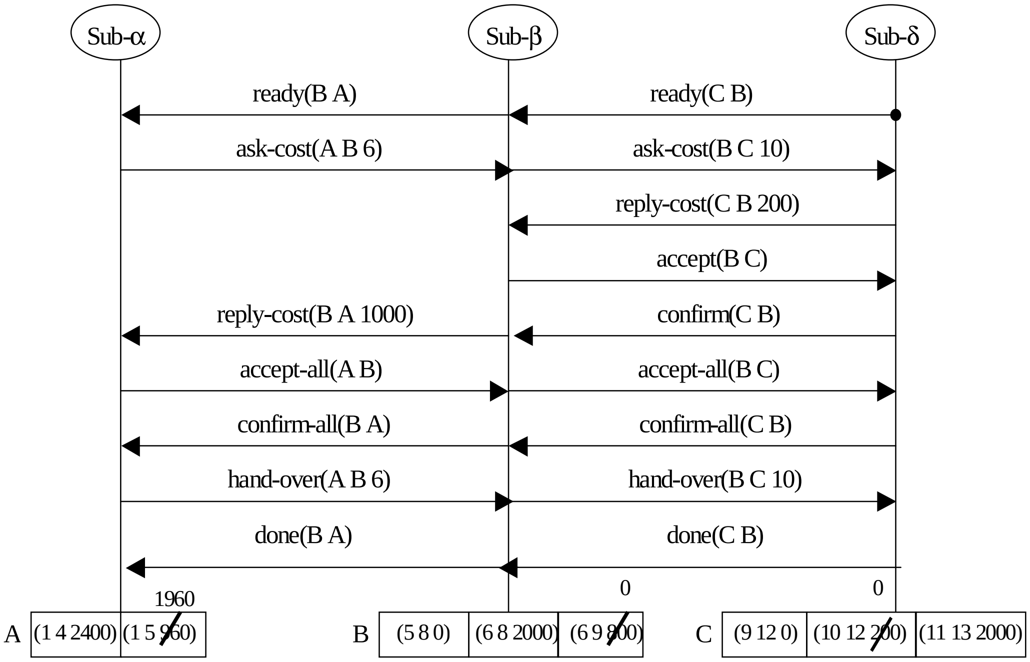


Figure 4-12. Conversation sequences during the negotiation

To summarize, Activity-A of Sub-α will change to its preferred schedule. Then, Sub-α transfers $1,000 to Sub-β for its loss due to the schedule change. Sub-β also transfers $200 to Sub-δ for the same reason. Sub-α still gains a profit of $440. Note that Sub-β and Sub-δ would not cooperate with Sub-α without compensation for their losses. Sometimes Sub-β and Sub-δ have to bear losses due to the schedule change of Activity-A. The case example shows that the agent-based compensatory negotiation methodology facilitates the distributed coordination of project schedule changes.

4.6 RELATION TO PREVIOUS WORK ON DISTRIBUTED AGENT-BASED COORDINATION

In this section, I relate my ABCN methodology to previous work on distributed agent-based coordination in terms of coordination strategy, interaction protocols, and coordination mechanisms (see Table 4-5).

.

No.	Previous work (Researcher(s))	Coordination Strategy	Interaction protocols	Coordination mechanisms
1	Generic partial global planning (Decker and Lesser 1992)	Non-compensatory	Pair-wise negotiation	Random
2	Automated contracting: TRACONET (Sandholm and Lesser 1995)	Compensatory via profit-seeking bidding	Multi-lateral negotiation	Random
3	Agent-based distributed meeting scheduling (Sen and Durfee 1996)	Non-compensatory	Multi-lateral negotiation and voting	Random
4	Distributed constraint-satisfaction problem (Yokoo et al 1992)	- -	Backtracking	Unique ID
5	Coordination as distributed search (Durfee and Montgomery 1991)	- -	Backtracking	Pecking order
6	Rules of encounter: Unified negotiation protocol (Rosenschein and Zlotkin 1994)	Non-compensatory via coin flip	Pair-wise negotiation	Random
7	Clarke tax voting mechanism (Ephrati and Rosenschein 1996)	Non-compensatory and no distribution of collected taxes	Voting	Random
8	Multiagent compromise via negotiation (Sycara 1989)	Non-compensatory, but implicit utility transfer	Mediation	Random
9	Distributed constrained heuristic search (Sycara et al 1991)	- -	Backtracking	Heuristic order
10	Market-oriented programming (Wellman 1993)	Compensatory via profit-seeking bidding	Multi-lateral negotiation	Dependency
11	Enterprise: A market-like task scheduler (Malone et al 1988)	Compensatory via profit-seeking bidding	Multi-lateral negotiation	Priority
12	A dynamic theory of incentives in multi-agent systems (Shoham and Tanaka 1997)	Compensatory via incentive or reward mechanisms	- -	- -
	Agent-based compensatory negotiation	Compensatory via explicit and direct utility transfer	Multi-linked negotiation	Use of Sequence logic in CPM

Table 4-5. Summary of selected previous work

4.6.1 GENERIC PARTIAL GLOBAL PLANNING

Decker and Lesser (1992) presented the Generic Partial Global Planning (GPGP) that extends the Partial Global Planning (Durfee and Lesser 1991). The GPGP is a domain-independent framework for coordinating the real-time activities of small teams of agents. In the GPGP framework, each agent constructs its own local view of activities using TAEMS (Task Analysis, Environment Modeling and Simulation) task structure (Decker 1996). The local views of agents are coordinated into a schedule using a family of coordination algorithms (Decker et al. 1995) based on the criteria given by a client. The objective of GPGP is to find the best schedule of activities based on hard and soft constraints while maximizing the total payoff.

GPGP uses pair-wise negotiations for resolving conflicts between two agents, which occurs due to direct consequences of heterogeneous, dynamic, and real-time agents. Since GPGP assumes that the agents are cooperative, agents exchange their local views with other agents to find a better joint schedule, while relaxing their soft constraints. Even though they use utilities for evaluating the solutions toward the total payoff, there is no monetary compensation for disadvantageous agreements between agents.

4.6.2 AUTOMATED CONTRACTING

Sandholm and Lesser (1995) explored automated negotiations among agents that try to maximize payoff without concern for the global good (self-interested) in settings where computational limitations preclude enumerating and evaluating all possible outcomes.

For automated contracting, they extended the contract-net protocol, which was developed for cooperative agents, for self-interested, computationally limited agents. They augmented a formal model for announcing, bidding, and awarding decisions based on marginal-cost calculations, which was their early work on TRACONET (Sandholm 1993). The most closely related work is TRACONET, in which agents having very different local criteria can interact to distribute tasks so that the network as a whole functions more effectively.

TRACONET uses a multi-lateral (auction) protocol for trading surplus tasks among agents. TRACONET assumes agents to be self-interested so that they will not accept disadvantageous agreements without compensation. Furthermore, agents will seek any possible profit from the announcing-bidding-awarding protocols. Any profit-seeking bid from agents might prevent a system from reaching a better global solution.

4.6.3 AGENT-BASED DISTRIBUTED MEETING SCHEDULING

Sen and Durfee were working toward developing intelligent “surrogate” agent systems that automate meeting tasks of their associated humans. Their approach viewed meeting scheduling as a distributed search. However, they were not trying to derive any closed-form solution to the dynamic meeting scheduling problem because they believed any unique optimal solution to this problem does not exist. Rather, they focused on predicting the expected efficiency of different reasonable scheduling heuristics under a variety of resource constraints, based on a formal model of the distributed meeting problem and process (Sen and Durfee 1998), developing a cancellation/rescheduling mechanism (Sen and Durfee 1996), and representing and reasoning with preference and bias of associated users (Sen et al. 1997). The cancellation/rescheduling mechanism in his work is the focus of my methodology, which tries to resolves conflicts when conflict avoidance is not possible.

In the Distributed Meeting Scheduling System, a host uses a multi-lateral (auction) protocol to schedule a meeting among agents based on user-input preferences or priorities. When conflicts occur when scheduling a meeting, the host applies a voting mechanism to arrive at consensus choices for the meeting time while balancing different preferences. The Distributed Meeting Scheduling System assumes agents to be cooperative so that agents will not compensate others for disadvantageous meeting times.

4.6.4 DISTRIBUTED CONSTRAINT-SATISFACTION PROBLEM

Yokoo and others (1992) proposed a distributed constraint-satisfaction problem (DCSP) framework to systemize cooperative distributed problem solving and methods by extending traditional constraint-satisfaction methods. For methods for DCSP, they introduced and compared three backtracking algorithms: centralized backtracking, synchronous backtracking, and their newly developed “asynchronous backtracking.” The experimental results showed that their asynchronous backtracking outperformed the

synchronous backtracking due to their additional parallelism. In their recent paper (Yokoo, et al. 1998), they modified the asynchronous backtracking algorithm into an asynchronous weak-commitment search. Their experimental results showed that the new algorithm was more efficient than the asynchronous backtracking algorithm.
DCSP uses asynchronous backtracking or search mechanism to find a set of values for all variables such that all constraints are satisfied. Since DCSP assumes that all constraints cannot be relaxed, no conflict will occur between agents and, therefore, they did not develop a conflict-resolution mechanism. In other words, DCSP finds a satisfactory solution that produces no disadvantageous agreement for any agent.

4.6.5 COORDINATION AS DISTRIBUTED SEARCH

Durfee and Montgomery (1991) identified five key components of the theory of coordination: hierarchical behavior representation, metrics, distributed search protocol, local search algorithms, and control knowledge and heuristics. In their theory, agents form their behavior hierarchies, but do not know with whom they might interact. The superior agent, therefore, broadcasts their abstract-level goals according to the given authority value and the inferior agents resolve conflicts either by delaying their behaviors or by searching for non-conflicting behavior at a more detailed level through interactions with other agents. When the superior knows that no conflict exists, it passes control to the next agent in the pecking order. This process repeats to the lowest agent. Agents also use control knowledge and heuristics for search reduction. They used the conflict avoidance metrics for evaluating collective behaviors.

Their Coordination as Distributed Search method employs pairwise-interaction protocol between agents even though agents' alternative behaviors might affect other agents' behaviors. Also, agents might find difficulties in choosing which strategy to use to resolve conflicts because they lack monetary metrics. Agents would need the monetary metrics for choosing a strategy when agents are in the tightly coupled networks where conflicts apparently occur. Like the distributed constraint-satisfaction problem, Coordination as Distributed Search finds a satisfactory solution that produces no disadvantageous agreement for any agent.

4.6.6 RULES OF ENCOUNTER: UNIFIED NEGOTIATION PROTOCOL

For multiagent systems (MAS) that consist of self-interested heterogeneous agents, Rosenschein and Zlotkin (1994) analyzed the attributes of the domain in which the agents are operating and discussed the available interaction protocols to satisfy the efficiency, stability, simplicity, distribution, and symmetry conditions. They presented the Unified Negotiation Protocol to resolve the conflicts as well as to reach cooperative agreements between agents. With the Unified Negotiation Protocol, conflicting agents flip a coin to decide who is going to achieve one of their goals and, no matter who wins, commit themselves to work together in a joint plan.

One of their assumptions is that the Unified Negotiation Protocol does not allow explicit utility transfer between agents. Because of this assumption, their agents need a non-compensatory conflict-resolution strategy to resolve conflicts and to solicit agents to be cooperative for reaching an agreement that would be disadvantageous to one of them. The Unified Negotiation Protocol is a pairwise-interaction protocol that cannot be used for cases where agents’ alternative behaviors might affect other agents' behaviors.

4.6.7 CLARKE TAX VOTING MECHANISM

Ephrati and Rosenschein (1996) used the Clarke tax voting procedure as a method for reaching consensus without negotiation. The Clarke tax voting procedure is non-manipulative so that using it ensures stability of the system. According to this procedure, all agents vote their preferences over a set of alternatives and an alternative that gets the highest votes gets selected as a consensus.

Even though their voting procedure assumes an explicit utility transferability from self-interested agents to the central controller, it does allow a way to distribute the collected tax among agents. Distribution of collected tax will undermine the stability of the system. This means that their procedure only ensures the stability of the system, and does not compensate agents for disadvantageous agreements.

4.6.8 MULTIAGENT COMPROMISE VIA NEGOTIATION

Sycara (1989) presented a general negotiation model based on integration of case-based reasoning and multi-attribute utility theory. She implemented the negotiation model in the PERSUADER system to resolve labor-management disputes. The PERSUADER system, which modeled human mediators, uses negotiation to find a compromise that is acceptable to the agents in conflict.

Humans use pairwise-negotiation protocols via the central PERSUADER system for mediating conflicting labor issues in practice. Therefore, her negotiation model cannot be automated and must involve real human entities. The PERSUADER system provides a way of transferring utility between agents, but it is implicit and does not provide compensation for disadvantageous agreements.

4.6.9 DISTRIBUTED CONSTRAINED HEURISTIC SEARCH

Sycara and others (1991) presented a distributed problem-solving technique that is called Distributed Constrained Heuristic Search (DCHS). This model views problem solving as constraint optimization, incorporates heuristic search, and extends constraint satisfaction to optimization problems. Since the general constraint satisfaction problem (CSP) is an NP-complete problem, they devised a set of heuristics to reduce the search space, which are variable-ordering heuristics to decide which variable to initiate next and value-ordering heuristics to decide which value to assign to a variable. Another way to reduce distributed search space is through distributed asynchronous back jumping, a type of distributed dependency-directed backtracking.

DCHS uses an asynchronous search mechanism to find a set of values for all variables such that all constraints are satisfied. Since DCHS assumed that all constraints would not be relaxed, no conflict will occur between agents and, therefore, no conflict-resolution mechanism is presented. In other words, DCHS finds the best solution among satisfactory solutions that produce no disadvantageous agreement for any agent.

4.6.10 MARKET-ORIENTED PROGRAMMING

Wellman (1993) presented a market-oriented programming approach to distributed problem solving as a way to allocate tasks and resources for a set of computational agents by computing the competitive equilibrium of an artificial economy. His market-price system focused on effective decentralization of decision making with minimal communication overhead. The general equilibrium theory regards agents as consumers and producers and defines their tasks in terms of production and consumption of commodities. Consumers can buy, sell, and consume goods, and specify their preferences by their utility function. Producers can transform some sorts of goods into some others according to their technology that specifies the feasible combination of inputs and outputs for the producers. They reach competitive equilibrium when the total amount consumed equals the total amount produced, plus the total amount the economy started out with. Interactions between agents are exchanges, the terms of which are mediated by the underlying economic mechanism, or protocol.

Walsh and Wellman (1998) presented a decentralized market protocol for allocating tasks among agents that contend for scarce resources. Through a series of experiments with profit-maximizing bidding policies by agents, they verified that the decentralized market protocol would converge to a solution when one exits.

While Wellman's market price system provides an efficient way to allocate tasks or resources, its auction mechanism substitutes for a direct negotiation protocol between agents. Another difference is that his market protocol needs agents' inputs and outputs to be explicitly defined for auctions. It assumes agents in a market-price system to be self-interested so that they will not accept disadvantageous agreements without compensation. Furthermore, agents will seek any possible profit from the auction mechanism. Any profit-seeking bid from agents might prevent a system from reaching a better global solution.

4.6.11 ENTERPRISE: A MARKET-LIKE TASK SCHEDULER

Malone and others (1988) presented the Enterprise system for sharing tasks among personal workstations connected by a local area network (LAN). The system includes a distributed scheduling protocol (DSP) that assigns tasks to the best machines available at run-time, based on the metaphor of a market. According to DSP, the client sends out a “request for bids” that includes the numerical priority of tasks and contractors respond with “bids” giving their estimated completion times. After evaluation of bids, the client assigns the task to the best bidder. If a later bid is “significantly better” than the best early one, the client cancels the task on the early bidder and sends the task to the later bidder.

DSP in the Enterprise system provides an efficient way to assigns tasks to the best machines available at run-time, but its auction mechanism substitutes direct negotiation protocols between agents. Like other market systems, the Enterprise system assumes agents to be self-interested so that agents will not accept disadvantageous agreements without compensation.

4.6.12 A DYNAMIC THEORY OF INCENTIVES IN MULTI-AGENT SYSTEMS

Shoham and Tanaka (1997) proposed a dynamic model of incentives in multi-agent systems by investigating the role of incentives in “public goods” settings, that is, settings in which the system’s members supply the value of the system. For the dynamic model that is based on decision theory and economics (including game theory), they defined a growth function, a reward function, a disutility function, and a utility function, which will set the reward mechanism so as to ensure that by optimizing their own objectives the agents will also optimize the global objectives.

In their dynamic model of incentives, there is no discussion about interaction protocols. The coordination strategy they adopted uses an incentives and reward mechanism to optimize global objectives in multi-agent systems. Their indirect control principle by the central controller coordinates self-interested agents, but it has to bear overhead costs using the indirect control principle.

In summary, previous work on distributed agent-based coordination has inadequately provided any of the compensatory negotiation strategy, multi-linked interaction protocols, and coordination mechanisms based on CPM, which are necessary for the agent-based compensatory negotiation methodology, as shown in Table 4-5.

4.7 SUMMARY OF ABCN METHODOLOGY

This chapter presented a novel agent-based compensatory negotiation (ABCN) methodology to facilitate the distributed coordination of project schedule changes. The methodology consists of a compensatory negotiation strategy based on utility, multi-linked negotiation protocols, and message-handling mechanisms. This chapter illustrated the methodology using a simple case example. It also reviewed previous work and stated my contributions.

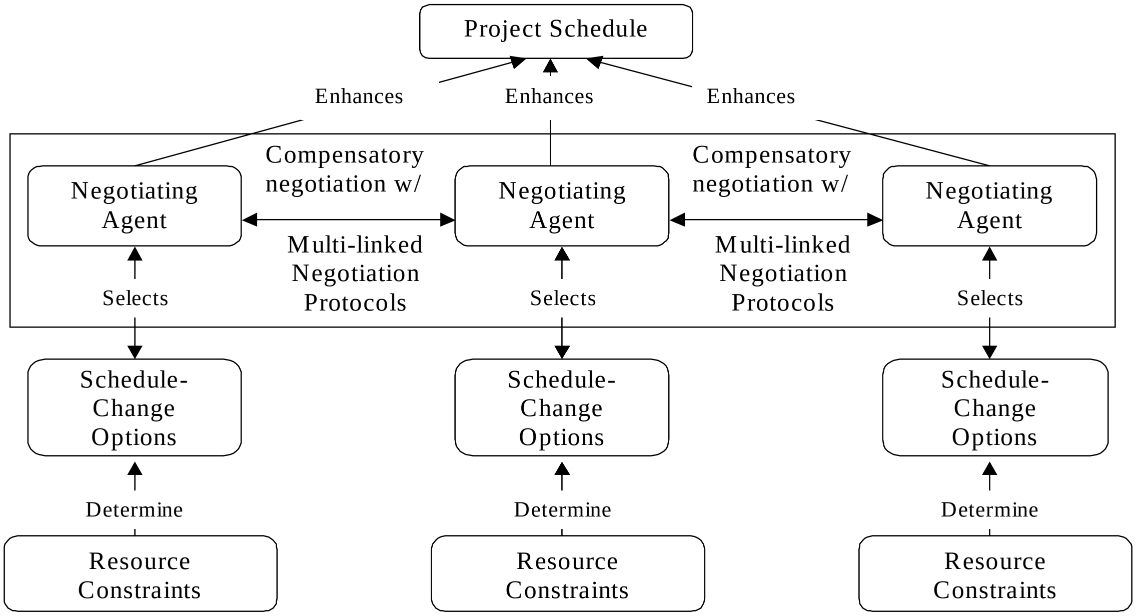


Figure 4-13. Conceptual diagram of agent-based compensatory negotiation methodology

Using the aforementioned agent-based compensatory negotiation methodology, agents can accomplish four crucial procedures: (1) calculating utility with a utility function from the predefined schedule-change options; (2) exploring feasible alternatives by collaboration with other agents using multi-linked negotiation protocols; (3) evaluating the impact of their alternatives; and (4) making appropriate decisions based on the evaluation. When the decisions affect other agents' activities, agents transfer utility to compensate other agents that are forced to make disadvantageous agreements, and, as a result, agents cooperatively enhance the overall project schedule in a distributed manner (see above Figure 4-13).

I conclude that the proposed agent-based compensatory negotiation methodology facilitates the distributed coordination of project schedule changes by meeting the practical challenges stated in Section 1.5, as follows:

- By using schedule-change options based on utility of timing, agents on behalf of subcontractors can compensate other agents for disadvantageous agreements
- By employing multi-linked negotiation protocols, agents on behalf of subcontractors can identify schedule conflicts, consider alternatives, and resolve schedule conflicts in a tightly coupled network of related activities
- By directing message-passing based on the CPM-based schedule, agents on behalf of subcontractors can maintain work logic and ensure convergence of the distributed coordination

In the next chapter, I will describe a multi-agent system that implements the ABCN methodology for distributed coordination of project schedule changes.

CHAPTER 5

DISTRIBUTED SUBCONTRACTOR AGENT SYSTEM: A MULTI-AGENT SYSTEM FOR DISTRIBUTED COORDINATION OF PROJECT SCHEDULE CHANGES

This chapter presents a multi-agent system for distributed coordination of project schedule changes (DCPSC) wherein a project can be rescheduled dynamically through negotiations by all of the concerned subcontractors. In the multi-agent system called the Distributed Subcontractor Agent System (DSAS), subcontractors interact with their software agents to evaluate the impact of changes, simulate decisions, and get the negotiation results that they need to reschedule the project.

5.1 INTRODUCTION

This research produced a distributed coordination framework for project schedule changes (DCPSC), wherein concerned subcontractors can reschedule a project dynamically through negotiations. To enable the DCPSC framework, I developed an agent-based compensatory negotiation (ABCN) methodology that allows software agents to evaluate the impact of changes and simulate decisions on behalf of human subcontractors. I needed to implement a prototype of a multi-agent system in order to demonstrate that the DCPSC and the ABCN are formalized enough to develop the DSAS. Chapters 3 and 4 provided details of the DCPSC and ABCN methodology, respectively.

The requirements for developing the multi-agent system called Distributed Subcontractor Agent System (DSAS) are as follows:

- Subcontractor agents should have the functionalities of negotiating agents modeled in the ABCN methodology.
- Human subcontractors can interact with their agents to provide them with the needed information for negotiations and to get the negotiation results that they needed to reschedule the project, which is the objective of the DCPSC.

5.2 DISTRIBUTED SUBCONTRACTOR AGENT SYSTEM

According to the aforementioned requirements, we designed and implemented a multi-agent system called the Distributed Subcontractor Agent System (DSAS). This section describes the DSAS architecture, subcontractor agents, graphic user interfaces, and agent message router.

5.2.1 DSAS ARCHITECTURE

DSAS consists of multiple subcontractor agents that have functionalities of the ABCN methodology, multiple Graphic User Interfaces (GUIs) for human subcontractors to interact with their subcontractor agents, and the Agent Message Router (AMR), which routes messages between agents over the Internet, as shown in Figure 5-1. The following sections describe details of the subcontractor agents, GUI, and AMR.

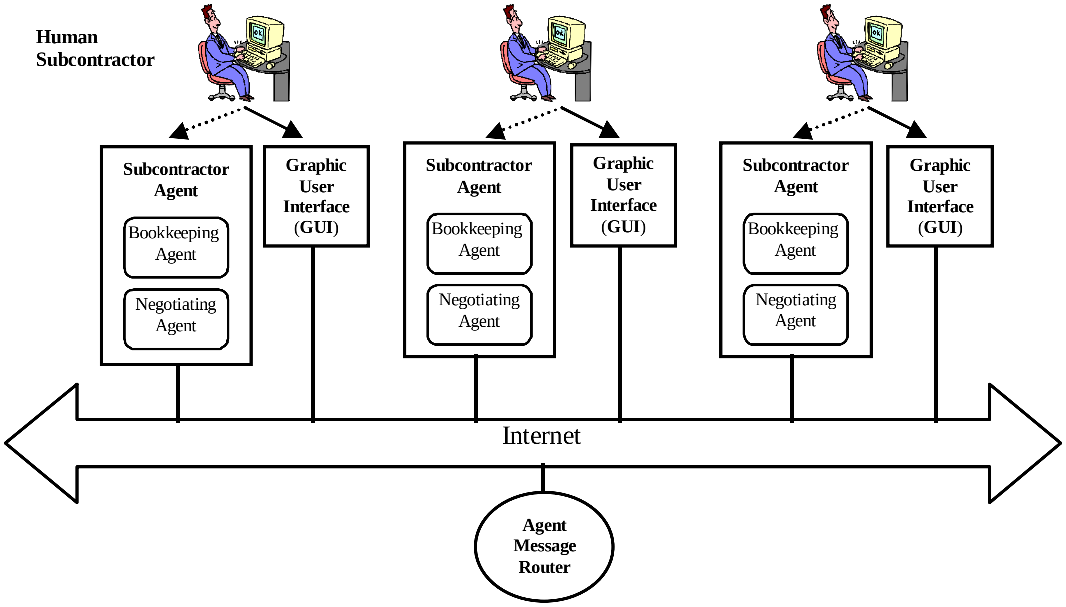


Figure 5-1. DSAS architecture

5.2.2 SUBCONTRACTOR AGENTS

Subcontractor agents, on the basis of the schedule-change options input by the human subcontractors, simulate decision-making on behalf of human subcontractors. The subcontractor agents consist of three important classes, the *Subcontractor* class, the *BookkeepingAgent* class, and the *NegotiatingAgent* class, as well as of other helper classes, as shown in Figure 5-2.

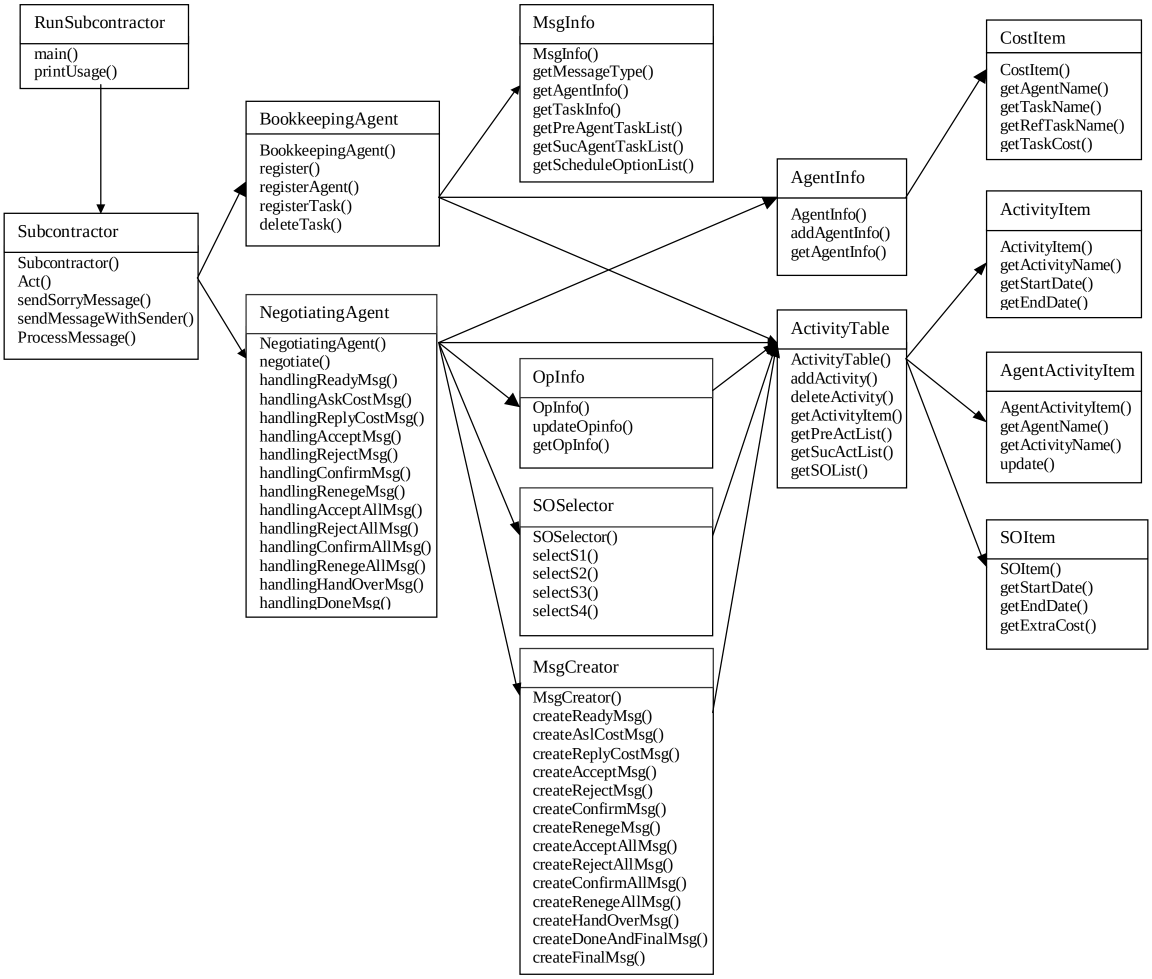


Figure 5-2. DSAS classes

5.2.2.1 Subcontractor Class

The *Subcontractor* class is the body of the subcontractor agent. It invokes the *BookkeepingAgent* class when the subcontractor agent receives messages from human subcontractors. It invokes the *NegotiatingAgent* class when the subcontractor agent receives any other messages from subcontractor agents. The *NegotiatingAgent* class conducts actual compensatory negotiations. Only the *Subcontractor* class sends and receives the messages because it has the necessary name and password.

5.2.2.2 BookkeepingAgent Class

The *BookkeepingAgent* class handles “input” messages that contain agent or activity information. It parses the received message and stores the parsed information appropriately.

It uses the *MsgInfo* class to parse the received message and stores the parsed information using the *AgentInfo* class and the *ActivityTable* class. The *AgentInfo* class uses the *CostItem* class to store cost information after negotiation. The *ActivityTable* class uses the *ActivityItem* class, the *AgentActivityItem* class, and the *SOItem* class to store activity information.

5.2.2.3 NegotiatingAgent Class

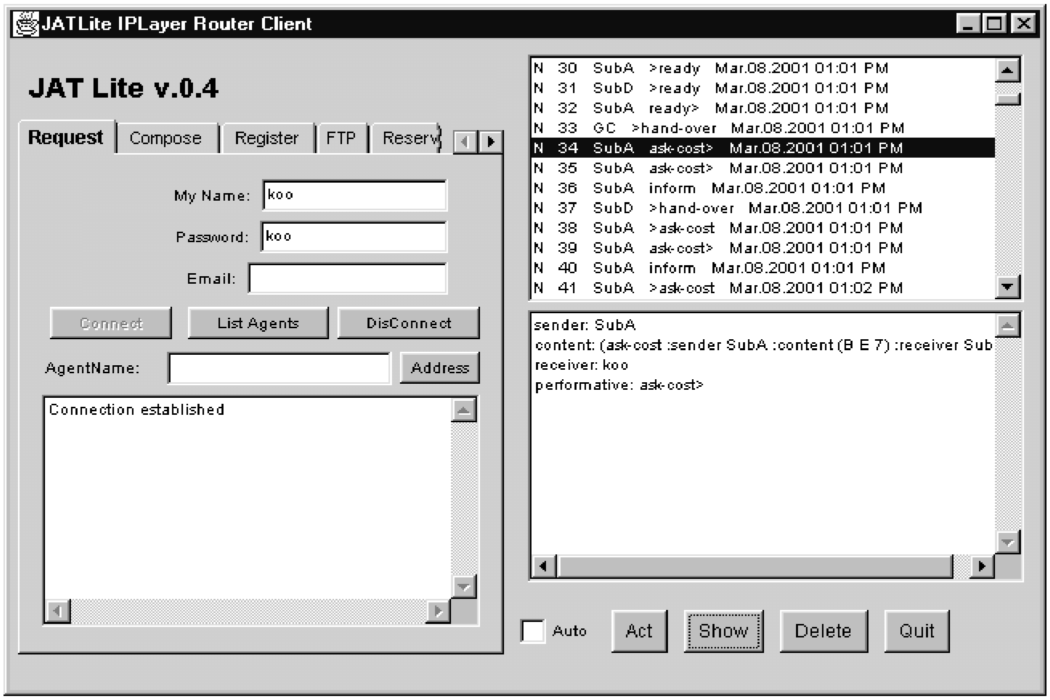
The *NegotiatingAgent* class handles “ready,” “ask-cost,” “reply-cost,” “accept,” “reject,” “confirm,” “renege,” “accept-all,” “reject-all,” “confirm-all,” “renege-all,” “hand-over,” or “done” messages. Based on the received message, it updates the stored information, selects an appropriate option, and/or generates outgoing messages.

It uses the *OpInfo* class to update the *status, check, agent,* and *activity* information stored in the *AgentInfo* and the *ActivityTable* classes. It uses the *SOSelector* class to select an appropriate schedule option based on the received message. It uses the *MsgCreator* class to generate outgoing messages.

5.2.3 GRAPHIC USER INTERFACES

Since I adopted the so-called “ Typed-Message Agent (TMA)” (Petrie, 1996), which stresses message passing based on shared, typed protocols and semantics to which the agent communities have committed, human subcontractors need to send “typed” messages to communicate with their subcontractor agents. DSAS provides each human subcontractor with a Graphic User Interfaces (GUI) to interact with its subcontractor agents. The GUI has the functionality to input “typed” messages for the subcontractor agent to handle.

In DSAS, I have only used the basic form of the GUI that is available in my choice of agent development environment, JATLite, as shown in Figure 5-3, and have done little customization for this particular application. While this has served my needs for research purposes, the GUI would need much more development for DSAS to become usable by real subcontractors.



My Name: koo

Password: koo

Email:

AgentName:

Connection established

```
N  30  SubA  >ready  Mar.08.2001 01:01 PM
N  31  SubD  >ready  Mar.08.2001 01:01 PM
N  32  SubA  ready>  Mar.08.2001 01:01 PM
N  33  GC  >hand-over  Mar.08.2001 01:01 PM
N  34  SubA  ask-cost>  Mar.08.2001 01:01 PM
N  35  SubA  ask-cost>  Mar.08.2001 01:01 PM
N  36  SubA  inform  Mar.08.2001 01:01 PM
N  37  SubD  >hand-over  Mar.08.2001 01:01 PM
N  38  SubA  >ask-cost  Mar.08.2001 01:01 PM
N  39  SubA  ask-cost>  Mar.08.2001 01:01 PM
N  40  SubA  inform  Mar.08.2001 01:01 PM
N  41  SubA  >ask-cost  Mar.08.2001 01:02 PM
```

```
sender: SubA
content: (ask-cost :sender SubA :content (B E 7) :receiver Sub
receiver: koo
performative: ask-cost>
```

Figure 5-3. Graphic User Interface (GUI) screen shot

5.2.4 AGENT MESSAGE ROUTER

In the distributed coordination framework for project schedule changes, the subcontractor agents and GUIs can communicate with other agents and with the GUIs. However, if the intended receiving agent does not exist at the time of communication, the communication will be lost. In fact, agents cannot be assumed to exist all the time in the distributed coordination framework for project schedule changes. Therefore, I needed to develop an

Agent Message Router (AMR) that buffers and forwards messages, much like an email server. The function of the AMR is to update the addresses of registered agents and to route messages between agents.

5.3 SUPPORTING STATE-OF-THE-ART TECHNOLOGIES

This section will review agent development environments and agent communication languages. They provide the supporting state-of-the-art technologies needed for developing my prototype for a distributed subcontractor agent system.

5.3.1 AGENT DEVELOPMENT ENVIRONMENTS

Tables 5-1 and 5-2 show many computer environments in various domains for agent development.

1	2	3	Name and URL	Company	Main Characteristics
	*	*	AgentBuilder (http://agentbuilder.com)	Reticular Systems, Inc.	An integrated software development tool to build intelligent agent-based applications
	*	*	Agentx (http://www.iks.com/agentx.htm)	International Knowledge Systems	Java-based distributed computing libraries that support object request broker, RMI and mobile agent services
*		*	Aglets (http://www.trl.ibm.com/aglets/)	IBM Japan	An environment for programming mobile Internet agents in Java
	*		CABLE (http://public.logica.com/~grace/Architecture/Cable/public/)	Logica Corporation	An environment for developing large and complex distributed applications for i) intelligent decision support and ii) modeling and simulation
		*	JACK (http://www.agent-software.com.au)	Agent Oriented Software Pty. Ltd	An environment for building, running and integrating JAVA-based multi-agent systems using a component-based approach.
		*	MadKit (http://www.madkit.org)	The MadKit Team	A Java multi-agent platform built upon an organizational model
	*	*	Voyager (http://www.objectspace.com)	ObjectSpace	A Java-based Object Request Broker (ORB) designed for mobile agents

Table 5-1. Commercial environments for agent development

1	2	3	Name	Research Group	Main Characteristics and References
*		*	ABS	Univ. of Toronto EIL	Reusable layers of languages and services for building agent systems (Barbuceanu and Fox 1995)
*	*	*	Bee-gent	Toshiba R&D Center	Completely "Agentifies" the communication between software applications (Kawamura et al. 1999)
		*	BOND	Purdue Univ.	A Java-based distributed object system and agent framework (Bölöni et al. 2000)
*		*	DECAF	Univ. of Delaware	A platform to design, develop, and execute intelligent agents to achieve solutions in complex software systems (Graham et al. 2000)
*			FarGo	Technion - Israel Institute of Technology	A Java-based programming environment for the development of mobile-component-based distributed applications (Ben-Shaul et al. 1999)
*		*	FIPA-OS	Nortel	A component-based architecture to enable the development of domain-specific agents which can utilize services of FIPA platform agents (Poslad et al. 2000)
*		*	Hive	MIT Media Lab	A Java software platform for creating distributed applications (Minar et al. 2000)
*		*	JATLite	Stanford Univ. Center for Design Research	A package of Java classes and programs that allow users to create new systems of software agents that communicate over the Internet (Jeon et al. 2000)
*		*	JIAC	Technical University Berlin DAI-Lab	A Java class library for the development of a universal architecture of agent-oriented systems (Albayrak and Wieczorek 1999)
		*	MAST	Technical University of Madrid	A general purpose distributed framework for the cooperation of multiple heterogeneous agents (Iglesias et al. 1995)
*		*	OAA	SRI AI Center	A framework for integrating heterogeneous software agents in a distributed environment (Martin et al. 1999)
*		*	RETSINA	Carnegie Mellon Univ. ISA Group	A system of reusable agent types that can be adapted to address a variety of different domain-specific problems (Sycara et al. 1996)
*	*	*	Zeus	British Telecom Lab ISR Group	A library of software components and tools that facilitate the design, development and deployment of agent systems (Nwana et al. 1999)

Table 5-2. Academic and research project environments for agent development

Note the following World Wide Web sources for agent development environments:

- **UMBC AgentWeb: Applications and Software: Software: Academic: Platforms** (http://agents.umbc.edu/Applications_and_Software/Software/Academic/Platforms/index.shtml, 11/15/2001 accessed)

- **UMBC AgentWeb: Applications and Software: Software: Commercial** (http://agents.umbc.edu/Applications_and_Software/Software/Commercial/index.shtml, 11/15/2001 accessed)

- **AgentBulder: Agent Construction Tools** (http://www.agentbuilder.com/AgentTools/index.html, 11/15/2001 accessed)

Among the many systems, I chose to use JATLite (Java Agent Template Lite) (Jeon et al. 2000), which was developed by the Center for Design Research (CDR) at Stanford University, to create my DSAS. JATLite is a package of programs written in the Java™ language that allow users to quickly create new software agents that communicate robustly over the Internet. JATLite provides a basic infrastructure in which agents register with an Agent Message Router facilitator using a name and password, connect/disconnect from the Internet, send and receive messages, transfer files, and invoke other programs or actions on the various computers where they are running. The advantages of adopting JATLite were:

- **JATLite provides an agent template for developing agents**. It enabled me to focus on implementing the functionality of the subcontractor agent in the Java™ language, without having to consider low-level message-passing details.
- **JATLite provides a simple GUI**. It enabled me to develop and test a prototype of DSAS without developing special GUIs.

- **JATLite provides a robust AMR.** It enabled me to develop and test a prototype of DSAS, without developing a special AMR.
- **JATLite is open-source free software**. It enabled me to modify the source code if necessary and it costs nothing. A disadvantage was the lack of technical support, but I could seek help from the developers directly because JATLite was developed at Stanford. The CDR also maintains JATLite user groups that can be consulted.

5.3.2 AGENT COMMUNICATION LANGUAGES

Currently, two standards exist for the agent communication languages: Knowledge Query and Manipulation Language (KQML) (Finin et al. 1994; Labrou and Finin 1997) and FIPA ACL (FIPA specification 2000). I chose to use KQML because JATLite, which is my choice of environment for agent development, currently uses KQML for its standard agent communication language.

KQML is a language and protocol for exchanging information and knowledge. It is part of a larger effort, the ARPA Knowledge Sharing Effort, which is aimed at developing techniques and methodology for building large-scale knowledge bases that are sharable and reusable. KQML is both a message format[9] and a message-handling protocol that supports run-time knowledge sharing among agents. KQML can allow an application

[9] The KQML string syntax in BNF is as follows (Labrou and Finin 1997):

```
<performative> ::=(<word> {<whitespace> :<word> <whitespace> <expression>}*)
<expression> ::= <word> | <quotation> | <string> | (<word> { <whitespace> <expression>}*)
<word> ::=<character><character>*
<character> ::= <alphabetic> | <numeric> | <special>
<special> ::= < | > | = | + | - | * | / | & | ^ | ~ | _ | @ | $ | % | : | . | ! | ?
<quotation> ::= '<expression> | `<comma-expression>
<comma-expression> ::= <word> | <quotation> | <string> | ,<comma-expression> (<word>
{<whitespace> <comma-expression>}*)
<string> ::="<stringchar>*" | #<digit><digit>*"<ascii>*
<stringchar> ::= \<ascii> | <ascii> -\-<double-quote>
```

program to interact with an intelligent system, or enable two or more intelligent systems to share knowledge in support of cooperative problem solving.

I used KQML to define the message format like:

```
(performative
        :sender         <word>
        :receiver:      <word>
        :language       <word>
        :content:       <expression>)
```

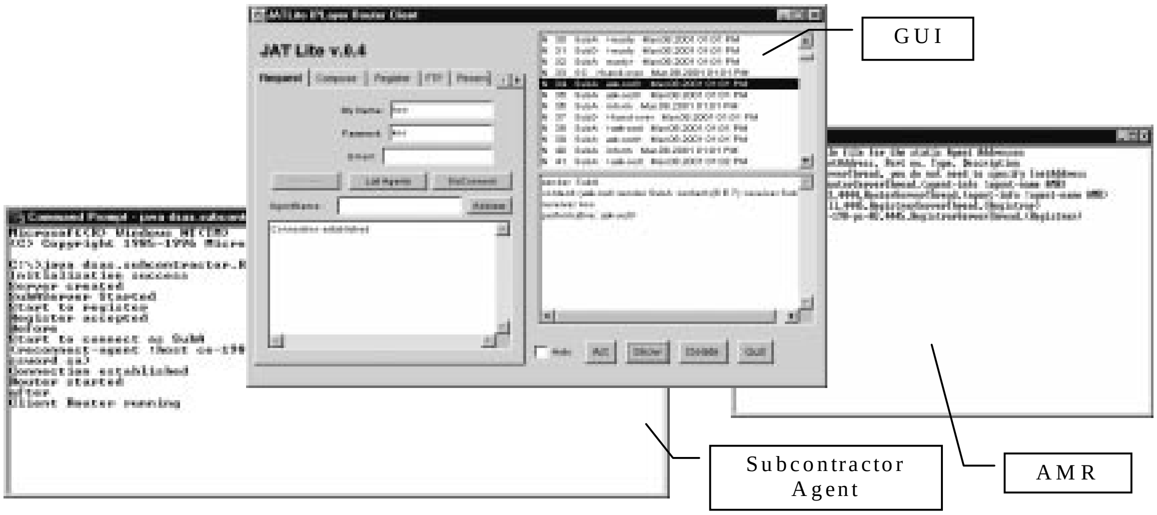
For example, an agent (Sub-A) sends a message to another agent (Sub-B) to find out the cost for starting activity B late because of a delay in activity A. The corresponding KQML message is:

```
(ask-cost
        :sender         Sub-A
        :receiver:      Sub-B
        :language       KQML
        :content:       (A B 5))
```

In summary, KQML allowed me to construct the messages for subcontractor agents in the distributed coordination framework for project schedule changes.

5.4 DSAS IMPLEMENTATION

I implemented the subcontractor agents in the Java™ language, which is object-oriented and portable across platforms, by extending JATLite's RouterLayer.AgentClient.RouterClientAction. Consequently, subcontractor agents can run on any machine that supports JDK™. The subcontractor agent development is also facilitated by JATLite, which provides Graphic User Interfaces (GUIs) — JATLite's ProtocolLayer.IPRCApplet — and the Agent Message Router (AMR) — JATLite's ProtocolLayer.IPRouterAction — for a robust message-passing infrastructure. Figure 5-4 shows a screen shot of DSAS.



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Micro

C:\>java dsas.subcontractor.R
Initialization success
Server created
SubHServer Started
Start to register
Register accepted
Before
Start to connect as SubA
(reconnect-agent :host cs-198
(SubA)
Connection established
Router started
After
Client Router running
```

Figure 5-4. DSAS screen shot

Since the GUI complies with the JATLite AMR, human subcontractors can download the GUIs from Internet web browsers, such as Microsoft™ Internet Explorer, Netscape™ Navigator, or Microsystems™ appletviewer. Thus, human subcontractors can interact with their agents without geographic restrictions. Combined with the portability of the subcontractor agents, human subcontractors can use DSAS to coordinate project schedule changes anywhere in the world.

5.5 SUMMARY OF DSAS

The objective of my work was to demonstrate that the distributed coordination of project schedule changes based on the agent-based compensatory negotiation methodology is formalized enough to develop a multi-agent system called Distributed Subcontractor Agent System (DSAS). DSAS is a multi-agent system that consists of multiple subcontractor agents, multiple graphic user interfaces, and an agent message router.

In DSAS, subcontractor agents negotiate with other subcontractor agents based on schedule-change options for distributed coordination of project schedule changes using KQML messages over the Internet. Through Graphic User Interfaces (GUIs), human subcontractors can interact with their subcontractor agents to provide schedule-change options and get negotiation results. The Agent Message Router (AMR) provides a robust message-passing infrastructure. I implemented the subcontractor agents in the Java™ language. JATLite (Java Agent Template) facilitated the development of subcontractor agents and provided Graphic User Interfaces (GUIs) and the Agent Message Router (AMR).

Using the developed DSAS, the next chapter will describe test methodologies and test results to demonstrate the effectiveness of the distributed coordination of project schedule changes based on the agent-based compensatory negotiation methodology.

CHAPTER 6

EVALUATION

This chapter demonstrates the effectiveness of the agent-based compensatory negotiation (ABCN) methodology for distributed coordination of project schedule changes through evaluation tests. It compares two centralized coordination methodologies used in current practice to the DCPSC-based ABCN methodology in terms of extra costs and project duration. I conducted charrette tests of the distributed subcontractor agent system (DSAS), which is a multi-agent system employing DCPSC-based ABCN methodology, to test the effectiveness compared to manual centralized processes. I also conducted a series of experimental tests with different schedules to measure the system performance of DSAS.

6.1 COMPARISON TESTS

As a basis for evaluating the effectiveness of ABCN methodology, I compared the results of two centralized coordination methodologies used in current practice with the results of ABCN methodology in terms of extra costs and project duration.

6.1.1 SCHEDULE CHANGE OPTIONS

From the available resource histograms in the case example (see Section 1.3), assume that schedule change options are pre-defined in the format of (startDate(day) endDate(day) extraCost($)), as shown in Table 6-1. Note that options marked ‘*’ are initially available options, which are feasible because the start date of an activity is later than the end dates of the preceding activities. The option marked '**'includes liquidated damages ($4,000) for a 2-day project delay, which the GC needs to pay. Keep in mind that the predefined schedule-change options are prepared by each subcontractor and are initially kept private by each subcontractor.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480)	(1 4 0)			
B	(4 7 1920)	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0)				
D	(8 10 0)	(9 10 0)	(10 11 960)		
E	(8 10 0)	(9 10 640)	(10 12 0)		
F	(8 9 0)	(9 10 384)	(10 11 768)		
G	(11 12 0)	(13 14 4512)**			

Table 6-1. Predefined schedule-change options before coordination

6.1.2 AGENT-BASED COMPENSATORY NEGOTIATION

Under the agent-based compensatory negotiation (ABCN), Sub-α evaluates the acceleration cost (AC) ($480) for Activity-A, which is the difference between the option of A1 (1 3 480) and the option of A2 (1 4 0), and the delay cost (DC) ($5,760), which is the cost response from the succeeding activities of Activity-A. Then Sub-α decides to expedite Activity-A to finish on time as the option of A1 (1 3 480) because DC is much more than AC. Note that the cost overrun will not be reimbursed since Sub-α has to decide first according to its position in the network schedule. However, if AC was more than DC, Sub-α could keep its new preferred schedule while reimbursing the costs for succeeding activities.

Then Sub-β and Sub-δ, which have activities succeeding Activity-A, evaluate the options of their activities — Activity-B and Activity-C. Sub-β can use its new preferred schedule as the option of B2 (4 8 0) because the AC ($1,920) for expediting Activity-B is more than the DC ($1,024) for expediting Activity-E and delaying Activity-F. Sub-β will compensate the cost to Sub-α and Sub-δ, $640 for Activity-E and $384 for Activity-F respectively. Note that the initial options are changed due to the compensation. That is, B2' (4 8 1024) from B2 (4 8 0), E2' (9 10 0) from E2 (9 10 680), and F2' (9 10 0) from

F2 (9 10 384). Sub-δ will keep Activity-C the same as the initial schedule as the option of C1 (5 7 0) because its schedule is good.

Next, Sub-α, Sub-β and Sub-δ evaluate the options of their activities — Activity-E, Activity-D, and Activity-F. Due to changed options of Activity-E and Activity-F, all of subcontractors will keep the changed schedule as E2’ (9 10 0), D2’ (9 10 0), and F2’ (9 10 0) because their schedules are good. Finally, Sub-α evaluates the options of Activity-G and decides to keep the schedule as F1 (11 12 0). Appendix-D shows step-by-step ABCN on DSAS.

Figure 6-1 shows revised resource histogram after ABCN. The diagonal hatching indicates the overtime. Figure 6-2 shows the revised schedule after ABCN.

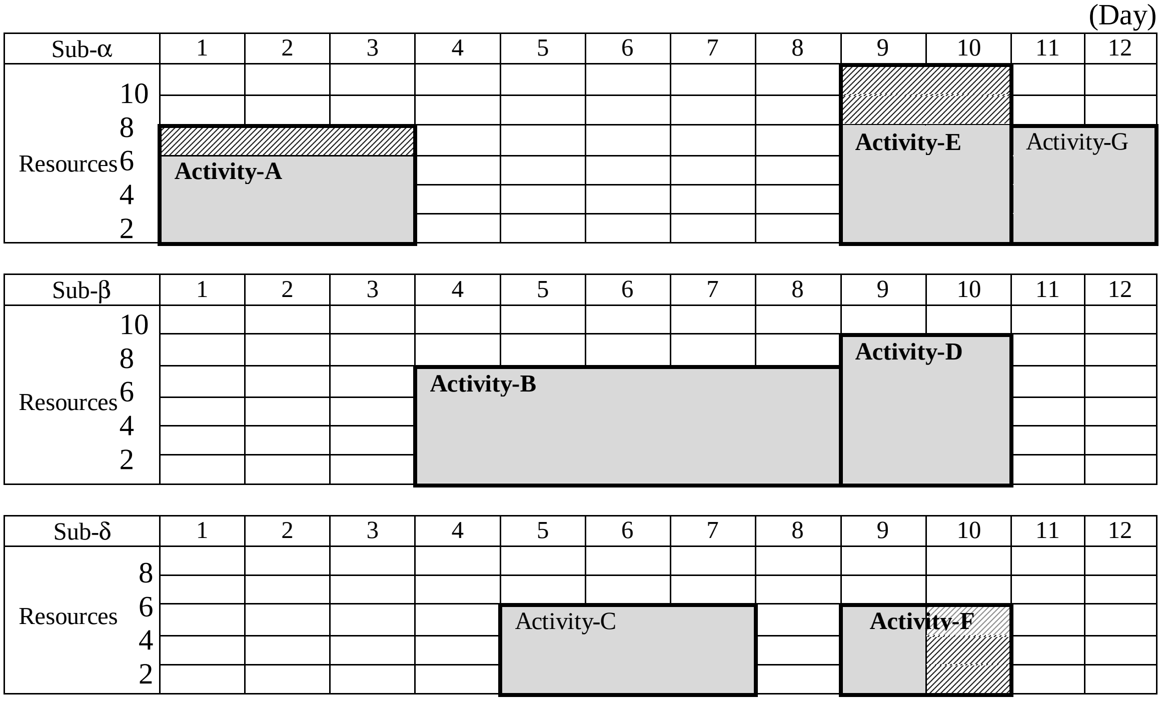
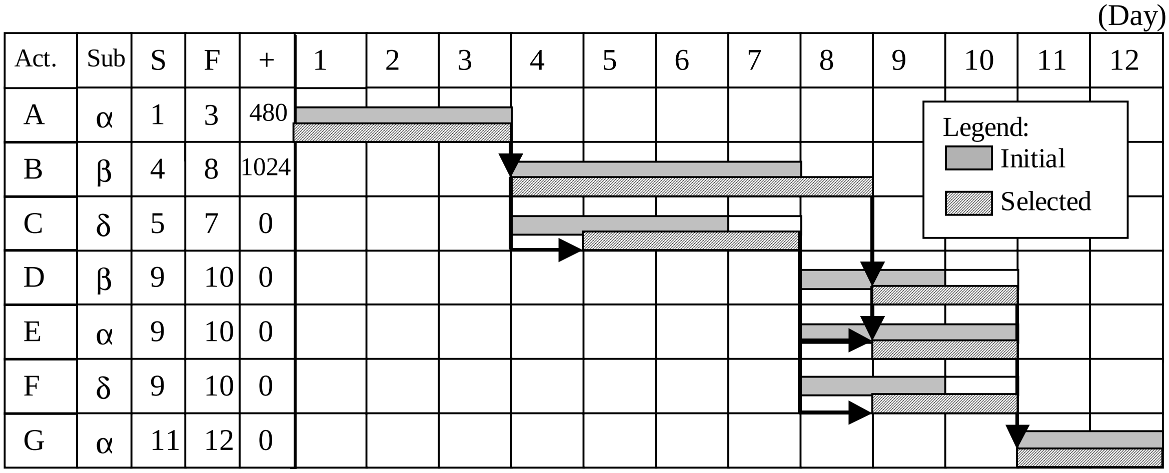


Figure 6-1. Revised resource histogram after ABCN



Act.	Sub	S	F	+
A	α	1	3	480
B	β	4	8	1024
C	δ	5	7	0
D	β	9	10	0
E	α	9	10	0
F	δ	9	10	0
G	α	11	12	0

Figure 6-2. Revised schedule after ABCN

The chosen schedule change options after ABCN are shown marked "✓" in Table 6-2, and Table 6-3 summarizes the results after ABCN.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920)	(4 8 1024) ✓	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0) ✓	(10 11 960)		
E	(8 10 0)	(9 10 0) ✓	(10 12 0)		
F	(8 9 0)	(9 10 0) ✓	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table 6-2. Selected schedule-change options after ABCN

Coordination methodology		Extra cost					Duration
		Sub-α ($)	Sub-β ($)	Sub-δ ($)	GC ($)	Total ($)	(days)
Distributed	ABCN	+480	+1,024	0	0	+1,504	12

Table 6-3. Summary of result after ABCN

In the ABCN above, subcontractors make agreements only if the agreements can lower the sum of subcontractors' costs while compensating others for disadvantageous agreements. In summary, ABCN enhances the project network schedule that has schedule conflicts, while lowering the sum of subcontractors' costs associated with their resource constraints by rescheduling the project subject to the precedence relationship among project activities in cases of changes in subcontractors' resource availabilities. Next, I evaluate the effectiveness of the ABCN by showing whether it can find a solution which is better than or equal to results from current practice.

6.1.3 TIGHT "IRON-FIST" CENTRALIZED COORDINATION

Under TCC, Sub-α has to choose the option of A1 (1 3 480). Other activities remain the same as options of E1 (8 10 0) and G1 (11 12 0). Sub-β also has to choose the option of B1 (4 7 1920). Activity-D will keep the same as option D1 (8 9 0). Sub-δ can keep its preferred schedule as the options of C1 (5 7 0) and F1 (8 9 0).

Chosen schedule change options after TCC are shown marked "✓" in Table 6-4. Table 6-5 summarizes the results after TCC.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920) ✓	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0) ✓	(9 10 0)	(10 11 960)		
E	(8 10 0) ✓	(9 10 640)	(10 12 0)		
F	(8 9 0) ✓	(9 10 384)	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table 6-4. Selected schedule-change options after TCC

Coordination methodology		Extra cost					Duration
		Sub-α ($)	Sub-β ($)	Sub-δ ($)	GC ($)	Total ($)	(days)
Centralized	TCC	+480	+1,920	0	0	+2,400	12

Table 6-5. Summary of results after TCC

In summary, TCC costs more for some subcontractors when they have different resource availability than their initial resource requirements. TCC lowers the resource utilization, even though TCC would guarantee to finish on time. Note that the central coordinator has no consideration of subcontractors' resource utilization in TCC.

6.1.4 LOOSE "LAISSEZ-FAIRE" CENTRALIZED COORDINATION

Under LCC, Sub-α can choose the option of A2 (1 4 0), but Activity-B has to choose the option of B5 (5 9 0) because of the option A2. Sub-δ can keep the option of C1 (5 7 0). Then, Sub-α, Sub-β and Sub-δ must select their options for their other activities — E3 (10 12 0) for Activity-E, D3 (10 11 960) for Activity-D, and F3 (10 11 768) for Activity-F. Next, Sub-α is forced to select the option of G2 (13 14 4512). Note the option includes liquidated damages for a 2-day project delay, which the GC needs to pay.

The chosen schedule change options after LCC are marked "✓" in Table 6-6, and Table 6-7 summarizes the results after LCC.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480)	(1 4 0) ✓			
B	(4 7 1920)	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0) ✓
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0)	(10 11 960) ✓		
E	(8 10 0)	(9 10 640)	(10 12 0) ✓		
F	(8 9 0)	(9 10 384)	(10 11 768) ✓		
G	(11 12 0)	(13 14 4512)** ✓			

Table 6-6. Selected schedule-change options after LCC

Coordination methodology		Extra cost					Duration
		Sub-α ($)	Sub-β ($)	Sub-δ ($)	GC ($)	Total ($)	(days)
Centralized	LCC	+512	+960	+768	+4,000	+6,240	14

Table 6-7. Summary of results after LCC

In summary, LCC costs more for some subcontractors when they have to employ new resources due to delays of preceding activities as well as their resource deviations. The project missed its completion date and the central coordinator has to pay liquidated damages of $4,000 in this case. Options to expedite the activities instead of paying liquidated damages, like time-cost tradeoff analysis (Fondahl 1961, 1991; Antill and Woodhead 1990), are limited because these need information, such as the cost slope for each activity, that usually is not available to the central controller.

6.1.5 COMPARISON OF THREE COORDINATION METHODOLOGIES

Table 6-8 summarizes the comparison among three different coordination methodologies in terms of cost and duration.

		Extra cost					Duration
Coordination methodology		Sub-α ($)	Sub-β ($)	Sub-δ ($)	GC ($)	Total ($)	(days)
Distributed	ABCN	+480	+1,024	0	0	+1,504	12
Centralized	TCC	+480	+1,920	0	0	+2,400	12
	LCC	+512	+960	+768	+4,000	+6,240	14

Table 6-8. Summary of results

ABCN can find a solution that is better than or equal to any of the results from the centralized coordination methodologies. In these examples, it is better. However, it does not eliminate all extra costs for some subcontractors in cases where some subcontractors have to expedite their activities by working overtime to avoid large costs for delaying or expediting succeeding activities. Note that, under ABCN, the project can be flexible on finish time if any subcontractor is willing to pay the project delay penalty in return for a delay. Appendix-E shows the generalization of these evaluation results with mathematical proofs.

6.2 DSAS CHARRETTE TESTS

In order to test the effectiveness of DSAS, I used the charrette test method (Clayton et al. 1998), which the Center for Integrated Facility Engineering (CIFE) at Stanford University has used to test the effectiveness of software systems. I conducted the charrette tests to compare two processes: one was a “manual” centralized coordination process and another was a computer-aided ABCN process on DSAS. The propositions to be tested are whether a computerized DSAS coordination produces the lower cost solution faster than a manual centralized coordination. The task of the participants was to find a better project schedule from schedule options, which were given to participants, in terms of costs and time taken. The reason why I used the charrette test method was that it could test the effectiveness of the prototype system from the human perspective. The human subjects of the charrette were graduate students at Stanford University. Since I used human subjects in the DSAS charrette tests, I applied for a review and approval from the Human Subject Panel at Stanford University (Protocol no: 0001-375).

6.2.1 DSAS CHARRETTE DESIGN

A group of five participants represented a hypothetical project team, which consisted of a general contractor (GC) and four subcontractors (Sub-α, Sub-β, Sub-δ, and Sub-ε). I provided each participant with a 27-activity CPM network schedule, a separate resource histogram, and schedule-change options for his or her activities.

I asked a group to remedy the given schedule while selecting the best combination among given schedule-change options subject to precedence, under one condition: They were not allowed to share their private schedule-change options with others. At first, I asked them to remedy the schedule manually in a 30-minute time frame. This means that they needed to find a better schedule using their collective knowledge and experience. Then, they used the DSAS system to find a computerized ABCN solution in another 20-minute time frame. I gave a 10-minute tutorial of DSAS before the DSAS session. They did not need any prior programming skills in using DSAS. Appendix-F shows the 27-activity and 5-agent schedule, a separate resource histogram, and separate schedule-change options for Sub-α. Other subcontractors and GC have similar information, except their respective resource histograms and schedule-change options.

6.2.2 TWO DSAS CHARRETTE TESTS

I conducted two DSAS charrette tests with two groups of five graduate students at Stanford University. Group-A's self-rated average skills in construction and computers were 2.8 and 3.6 respectively in the scale of 1 (novice) to 5 (expert), but one participant of this group had no prior construction experience, but had some knowledge about the principles of CPM scheduling. Group-B's average skills in construction and computers were 2.4 and 3.0 respectively, but one participant of this group had no prior construction experience and no knowledge about the principles of CPM scheduling.

Before the manual centralized coordination session, I spent about 30 minutes explaining the example schedule, the resource histogram, and especially the schedule-change options. The predicted total incremented costs and total duration for subcontractors were $2,000 and 44 days if all subcontractors would choose Option-1 for their activities, which maintained the initial schedule.

6.2.2.1 Manual Centralized Coordination Sessions

In the manual centralized coordination sessions, these two groups acted differently. Subcontractors in Group-A were more *competitive*, which meant they were seeking compensation for disadvantageous agreements. In contrast, subcontractors in Group-B were more *cooperative*, which means they did not seek compensation for disadvantageous agreements.

Group-A

For the manual trial, Group-A selected a GC whose skills in construction and computers were 3 and 3 respectively in the scale of 1 (novice) to 5 (expert). The GC centrally coordinated other participants who acted like subcontractors (Sub-α, Sub-β, Sub-δ, and Sub-ε).

When a subcontractor reported a change, the GC got the extra-cost information from the affected subcontractors directly and made decisions in a way to lower the sum of subcontractors' extra costs. The GC considered precedence relationships among activities during coordination, like the multi-linked negotiation protocols in Chapter 4. Then subcontractors compensated other *competitive* subcontractors for disadvantageous agreements by changing extra-cost information, like the compensatory negotiation strategy in Chapter 4. Indeed, the GC used a centralized coordination methodology similar to the agent-based compensatory negotiation methodology in Chapter 4. When the

GC had considered the final activity, it ended the trial. The manual centralized coordination session lasted 37 minutes and the solution was $540, with the same 44-day duration.

Group-B

Group-B selected a GC whose skills in construction and computers were 1 and 5 respectively in the scale of 1 (novice) to 5 (expert). Like Group-A, the GC centrally coordinated other subcontractors, but heavily relied on suggestions by more experienced subcontractors. Group-B's GC coordinated subcontractors similar to Group-A's GC, but *cooperative* subcontractors in Group-B did not seek compensation for disadvantageous agreements. Rather, they accepted their extra costs when they found that another's cost was more than theirs. Like Group-A, the GC forced participants to reveal their confidential schedule-change options and other subcontractors present could get that information. The manual centralized coordination session lasted 30 minutes and the solution was $620 with the same 44-day duration.

6.2.2.2 Computerized DSAS Sessions

In the computerized DSAS session, subcontractors coordinated themselves without the aid of a GC. In fact, the GC acted like another subcontractor who had only root and final activities. Before the computerized DSAS session, I gave 10-minute DSAS tutorials using the data of Sub-α, as shown in Appendix-F.

Group-A

This group used the DSAS system, whose subcontractor agents were running in the different computers, to find a better solution. They used the Graphic User Interface, by copying and pasting data from a given text file to provide their agents (Sub-α, Sub-β, Sub-δ, and Sub-ε) with the information needed for agent-based compensatory negotiations, such as the activity name, the initial schedule information, and the schedule-change options. Then they sent "input" messages to their agents. When they finished the inputs, they sent "ready" messages to their agents in order to start negotiations. All participants interacted with their agents at the same time. When a participant sent the last "ready" message to his agent, the agent started negotiations. Participants waited until their agents sent "final" messages to them.

After agents had sent all "final" messages, participants checked the negotiation results, by reading the contents of the "final" messages. Since no participant was responsible for collecting the results, I collected the negotiation results for my research purpose. The computerized DSAS session lasted for 17 minutes, including 11 minutes for data inputs, and the solution was $420 with the same duration.

Group-B

Group-B did not finish the computerized DSAS session due to various reasons: inappropriate preparation, errant data inputs, and time pressures.

6.2.2.3 Results of DSAS charrette Tests

Table 6-9 compared two groups' results. Group-A performed better than Group-B in manual centralized coordination in terms of cost, even though Group-B found their solution faster. Only Group-A produced results in the computerized DSAS session.

	Group-A			Group-B		
Processes	Time (min)	Extra Cost ($)	Duration (days)	Time (min)	Extra Cost ($)	Duration (days)
Manual Centralized	37	540	44	30	620	44
Computerized DSAS	17	420	44	Not available		

Table 6-9. Comparisons of results by group

Group-A

Table 6-10 shows the selected schedule-change options by Sub-α. Sub-α selected schedule-change options marked "✕" in the manual session. Sub-α selected schedule-change options marked "✓" by using the DSAS system. The total costs for Sub-α in manual and DSAS were $200 and $80, respectively. Sub-α chose Option-2 for Activity-

M because he thought Option-3 might affect some succeeding activities, but the succeeding activity, Activity-Q, could start one day later. Therefore, Option-3 was available without any cost. Note that Activity-I's Option-2 marked "*" changed from (11 14 120) to (11 14 0) because Sub-ε compensated Sub-α for delaying Activity-H.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
B	(2 4)	(2 4 80) ✕✓	(4 6 0)		
E	(6 9)	(7 9 0) ✕✓			
I	(10 13)	(10 13 0)	*(11 14 0) ✕✓	(12 15 240)	
M	(14 26)	(14 16 0)	(20 22 120) ✕	(25 27 0) ✓	
Q	(27 30)	(28 30 0) ✕✓			
U	(31 38)	(31 34 0) ✕✓	(32 35 80)		

Table 6-10. Selected schedule-change options for Sub-α (Group-A)

Table 6-11 shows the selected schedule-change options by Sub-β. The total costs for Sub-β by manual and DSAS methods were $0 and $0, respectively. Sub-β chose different options for Activity-V and -Y in manual and DSAS, but the results are the same from Day 35 to Day 43. Note that Activity-V and -Y's Option-2 marked "*" changed from (35 38 100) (40 43 100) to (35 38 0) (40 43 0) because of compensation by Sub-δ for delaying Activity-S.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
F	(10 15)	(13 15 0) ✕✓			
J	(16 23)	(16 17 0) ✕✓			
N	(24 25)	(24 25 0) ✕✓	(25 26 40)		
R	(26 27)	(26 27 0) ✕✓	(27 28 40)		
V	(34 38)	(34 38 0)	*(35 38 0) ✕	(35 39 0) ✓	
Y	(39 43)	(39 43 0) ✕	*(40 43 0) ✓	(40 44 40)	

Table 6-11. Selected schedule-change options for Sub-β (Group-A)

Table 6-12 shows the selected schedule-change options by Sub-δ. The total costs for Sub-δ by manual and DSAS methods were $100 and $100, respectively. Note that Activity-S's Option-2 marked "*" changed from (27 34 0) to (27 34 100) after compensating Sub-β for delaying Activity-V or -Y.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
C	(2 9)	(2 9 0) ✕✓			
G	(10 15)	(10 15 0) ✕✓			
K	(10 13)	(10 13 1200)	(12 15 900)	(16 19 0) ✕✓	
O	(24 26)	(24 26 0) ✕✓	(25 27 300)		
S	(26 33)	(26 33 300)	*(27 34 100) ✕✓		
W	(31 38)	(35 37 0) ✕✓			
Z	(34 43)	(38 40 0) ✕✓			

Table 6-12. Selected schedule-change options for Sub-δ (Group-A)

Table 6-13 shows the selected schedule-change options by Sub-ε. The total costs for Sub-ε by manual and DSAS methods were $120 and $120, respectively. Note that Activity-S's Option-2 marked "*" changed from (6 10 0) to (6 10 120) after compensating Sub-α for delaying Activity-I.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
D	(2 5)	(2 5 0) ✕✓			
H	(5 9)	(5 9 300)	*(6 10 120) ✕✓	(7 11 120)	
L	(16 23)	(16 23 120) ✕✓	(17 24 0)		
P	(24 25)	(24 25 0) ✕✓	(25 26 0)		
T	(27 30)	(27 30 0) ✕✓	(28 31 300)		
X	(28 38)	(31 32 0) ✕✓			

Table 6-13. Selected schedule-change options for Sub-ε (Group-A)

Table 6-14 shows the selected schedule-change options by GC. The total costs for GC by manual and DSAS methods were $0 and $0, respectively.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
A	(1 1)	(1 1 0) ✕✓			
Ω	(44 44)	(44 44 0) ✕✓	(45 45 2000)	(46 46 4000)	(47 47 6000)

Table 6-14. Selected schedule-change options for GC (Group-A)

Group-B

Table 6-15 shows the selected schedule-change options by Sub-α. Sub-α selected schedule-change options marked "✕." The current DSAS could not produce results that are compatible with the manual results by *cooperative* Sub-α because I assumed agents in

DSAS were *competitive*. Therefore, I prepared manually the compatible results marked "✓" for Group-B's Sub-α, by ignoring the compensation in the results by Group-A's Sub-α. The total costs for Sub-α by manual and DSAS methods were $320 and $200, respectively. Note that, like Group-A's Sub-α, Group-B's Sub-α also chose Option-2 manually for Activity-M for the same reason.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
B	(2 4)	(2 4 80) ✕✓	(4 6 0)		
E	(6 9)	(7 9 0) ✕✓			
I	(10 13)	(10 13 0)	(11 14 120)✕✓	(12 15 240)	
M	(14 26)	(14 16 0)	(20 22 120) ✕	(25 27 0) ✓	
Q	(27 30)	(28 30 0) ✕✓			
U	(31 38)	(31 34 0) ✕✓	(32 35 80)		

Table 6-15. Selected schedule-change options for Sub-α (Group-B)

Table 6-16 shows the selected schedule-change options by Sub-β. Like Sub-α, I manually prepared the compatible results marked "✓" for Sub-β, by ignoring the compensation in the results by Group-A's Sub-β. The total costs for Sub-β by manual and DSAS methods were $180 and $100, respectively. Note that Sub-β chose Option-2 for Activity-N and -R to resolve conflicts under time pressure since Sub-ε already chose the delay of Activity-P, which is the preceding activity of Activity-N.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
F	(10 15)	(13 15 0) ✕✓			
J	(16 23)	(16 17 0) ✕✓			
N	(24 25)	(24 25 0) ✓	(25 26 40) ✕		
R	(26 27)	(26 27 0) ✓	(27 28 40) ✕		
V	(34 38)	(34 38 0)	(35 38 100)	(35 39 0)✕✓	
Y	(39 43)	(39 43 0)	(40 43 100) ✕✓	(40 44 40)	

Table 6-16. Selected schedule-change options for Sub-β (Group-B)

Table 6-17 shows the selected schedule-change options by Sub-δ. The total costs for Sub-δ by manual and DSAS methods were $0 and $0, respectively.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
C	(2 9)	(2 9 0) ✕✓			
G	(10 15)	(10 15 0) ✕✓			
K	(10 13)	(10 13 1200)	(12 15 900)	(16 19 0) ✕✓	
O	(24 26)	(24 26 0) ✕✓	(25 27 300)		
S	(26 33)	(26 33 300)	(27 34 0) ✕✓		
W	(31 38)	(35 37 0) ✕✓			
Z	(34 43)	(38 40 0) ✕✓			

Table 6-17. Selected schedule-change options for Sub-δ (Group-B)

Table 6-18 shows the selected schedule-change options by Sub-ε. The total costs for Sub-ε by manual and DSAS methods were $120 and $120, respectively.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
D	(2 5)	(2 5 0) ✕✓			
H	(5 9)	(5 9 300)	(6 10 0) ✕✓	(7 11 120)	
L	(16 23)	(16 23 120)✕✓	(17 24 0)		
P	(24 25)	(24 25 0) ✓	(25 26 0) ✕		
T	(27 30)	(27 30 0) ✕✓	(28 31 300)		
X	(28 38)	(31 32 0) ✕✓			

Table 6-18. Selected schedule-change options for Sub-ε (Group-B)

Table 6-19 shows the selected schedule-change options by GC. The total costs for GC by manual and DSAS methods were $0 and $0, respectively.

Activity	ES-(EF+FF)	Option-1	Option-2	Option-3	Option-4
A	(1 1)	(1 1 0) ✕✓			
Ω	(44 44)	(44 44 0) ✕✓	(45 45 2000)	(46 46 4000)	(47 47 6000)

Table 6-19. Selected schedule-change options for GC (Group-B)

6.2.3 SUMMARY OF DSAS CHARRETTE TESTS

In summary, computerized DSAS coordination produced a lower cost solution faster than any of the manual centralized coordination efforts by two groups. In this section, I compare the results of the DSAS charrette tests to check whether a computerized DSAS coordination produces the lower cost solution faster than a manual centralized coordination.

DSAS produced the solution faster

Based on the time results in Section 6.2.2.3, Figure 6-3 shows the comparison results on the time taken to find its final solution among three cases. The computerized DSAS coordination produced the solution faster than any of the manual centralized coordination by two groups.

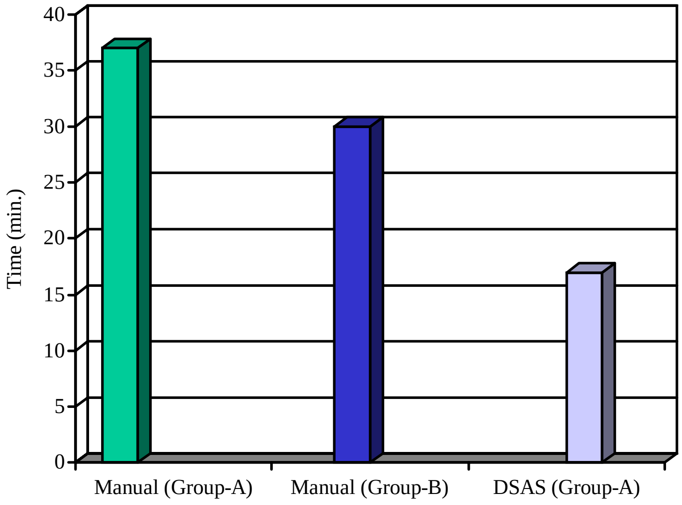


Figure 6-3. Time taken for each session

The reason is that computerized DSAS coordination used software agents that could communicate rapidly, and reasoning mechanisms inside software agents made decisions automatically. If the number of subcontractors and activities in schedules grows, the power of DSAS to produce a solution quickly will be more evident.

DSAS found a lower-cost solution

Based on the cost results in Section 6.2.2.3, Figure 6-4 shows the comparison results based on the total of extra costs among three cases. The computerized DSAS coordination produced a lower-cost solution than any of the manual centralized coordination efforts by the two groups.

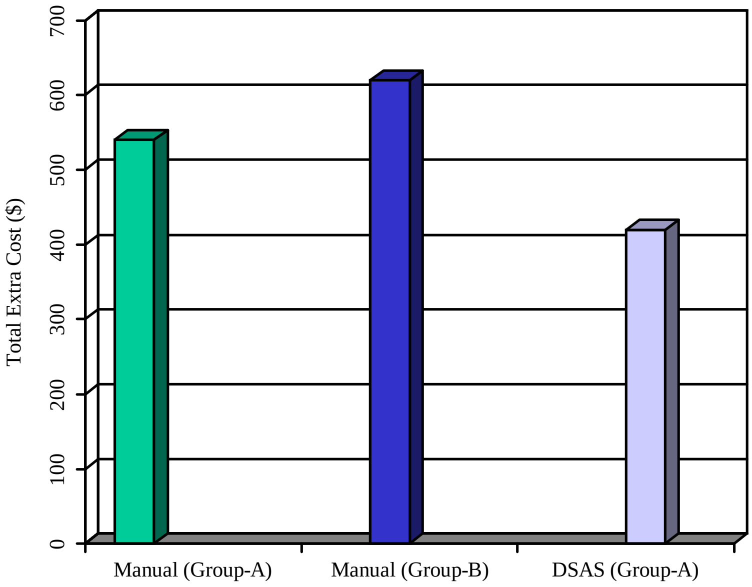


Figure 6-4. Total extra costs from each session

The reason is that computerized DSAS coordination considered more schedule-change options than manual centralized coordination because humans' bounded rationality limited them. Tables 6-10 and 6-15 show that Sub-α excluded the lower-cost option for activity-M, (25 27 0), because the finish date was outside of the allowable duration, (14 26), but the succeeding activity could start late without any extra cost so that the option should be chosen.

DSAS confirmed other advantages of ABCN

Besides showing the above quantitative benefits, a computerized DSAS coordination session confirmed the other advantages of ABCN:

- Computerized DSAS coordination maintained work logic better than manual centralized coordination because of human errors. Even though the final selection of schedule-change options do not show, several intermediate manual solutions by test groups violated the work logic so that the group had to choose unfavorable schedule-change options under time pressure, as shown in Table 6-16.

- Computerized DSAS coordination found a solution that was better than or equal to the initial solution. This test showed the differences between competitive groups and cooperative groups. *Competitive* subcontractors in Group-A selected options that were better than or equal to their initial options. In contrast, *cooperative* subcontractors in Group-B did not ask for compensation and, as a result**,** sometimes chose worse options than their initial options to select a better group solution, as shown in Figure 6-15. ABCN supports *competitive* subcontractors.

- Computerized DSAS coordination allowed subcontractors to keep the confidential information private. In manual centralized coordination, the GC forced subcontractors to reveal the confidential schedule-change options and other present subcontractors could get that information. In computerized DSAS coordination, no GC or subcontractor could get others' confidential information because the cost information was "scrambled" by agents. Every agent could reply with one of two different costs so that no one could guess a particular subcontractor's costs.

This DSAS charrette tests also revealed two limitations of applying the current DSAS system to real construction projects:

- One failure in a distributed agent in the DSAS system can cause the whole system to crash because of the tightly coupled nature of construction project schedules. I have experienced failures only when a user has provided wrong input data to his agent. When the agent crashes, it loses its data. The other agents are then not able to continue the negotiation processes. Since the AMR keeps messages but not data, the AMR does not help the agent to recover the lost data. I should upgrade the DSAS system so that agents can recover their data if one of them fails during the negotiation process.

- Human subcontractors may not trust their agents nor accept the results from DSAS until they fully understand the inside algorithms of their agents. Subcontractors would need intensive training about the ABCN.

6.3 SYSTEM PERFORMANCE MEASUREMENTS OF DSAS

In this section, I describe the test results of DSAS performance on the following five sample schedules from various sources.

- 3-activity & 3-agent case (constructed)

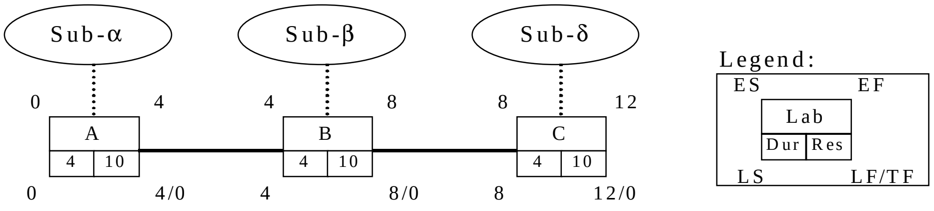


Figure 6-5. 3-activity and 3-agent sample schedule

- 7-activity & 3-agent case (constructed)

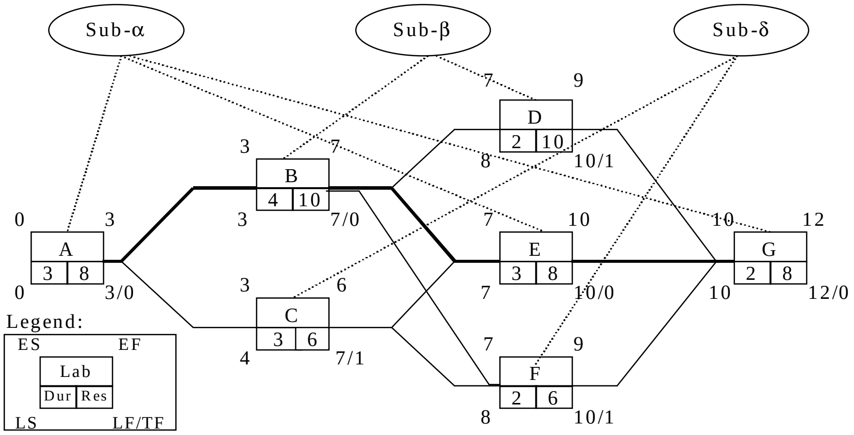


Figure 6-6. 7-activity and 3-agent sample schedule

- 15-activity & 5-agent case (adapted from Son et al. (1999))

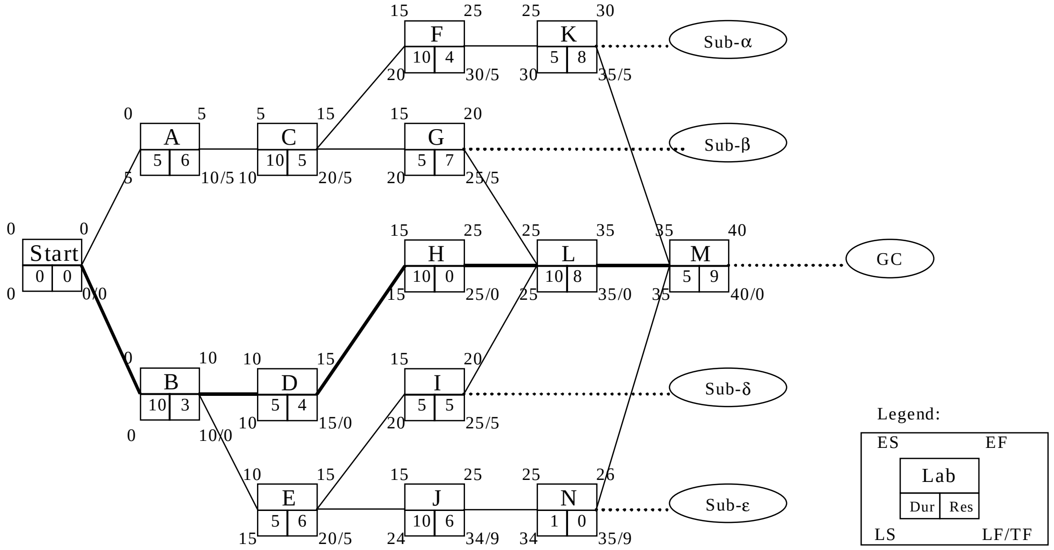


Figure 6-7. 15-activity and 5-agent sample schedule

- 22-activity & 5-agent case (adapted from Hegazy et al. (2000))

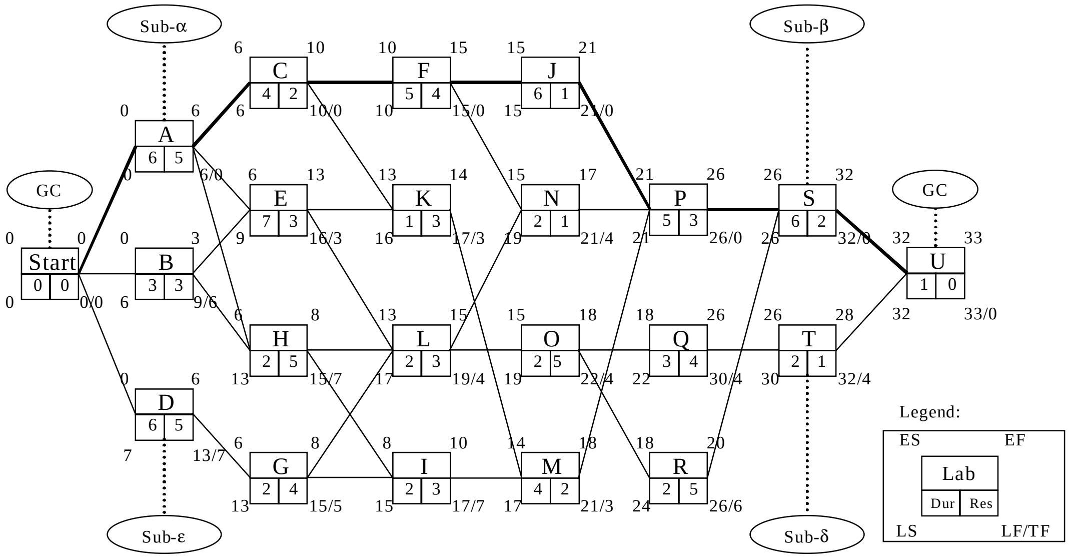


Figure 6-8. 22-activity and 5-agent sample schedule

- 27-activity & 5-agent case (adapted from Davis (1968))

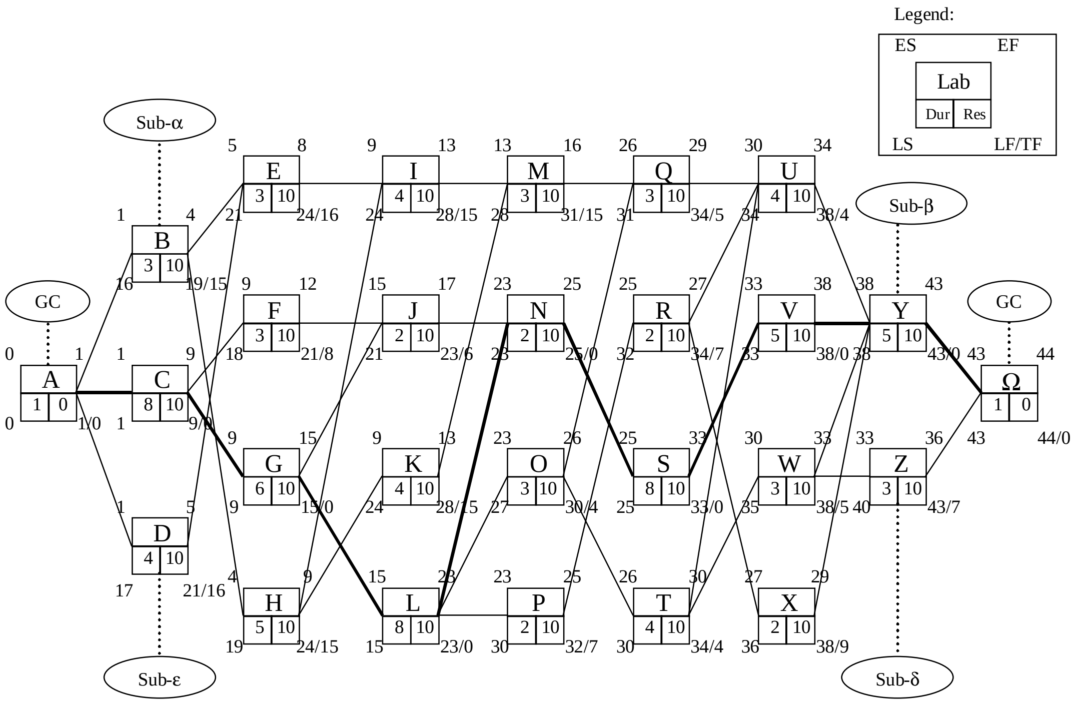


Figure 6-9. 27-activity and 5-agent sample schedule

I constructed the following test scenarios for each schedule and tested each case. A change means that an activity does not have enough resources for following the initial schedule:

- 3-activity & 3-agent case: 0 to 3 changes
- 7-activity & 3-agent case: 0 to 6 changes
- 15-activity & 5-agent case: 0 to 7 and 11 changes
- 22-activity & 5-agent case: 0 to 7 and 19 changes
- 27-activity & 5-agent case: 0 to 7 and 26 changes

I conducted tests on a Toshiba™ laptop computer equipped with a 400 MHz Mobile Intel™ Celeron™ processor. Tables 6-20 and 6-21 show the test results in terms of the number of messages and the time taken in minutes, and Figures 6-10 and 6-11 show the test results in graphical form.

(Unit: no. of Messages)

	No. of changes										
No. of activities	0	1	2	3	4	5	6	7	14	20	26
3	11	27	33								
7	43	122	162	174	181	187	192				
15	83	152	233	284	298	338	352	350	405		
22	142	403	413	509	529	614	665	685		906	
27	167	653	774	1,048	1,055	1,119	1,180	1,344			1,904

Table 6-20. Test results in terms of number of messages

(Unit: minutes)

	No. of changes										
No. of activities	0	1	2	3	4	5	6	7	14	20	26
3	1	1	1								
7	1	2	2	2	2	3	2				
15	2	2	2	3	2	3	3	3	3		
22	2	3	3	4	3	5	5	5		8	
27	1	4	6	6	6	7	8	8			11

Table 6-21. Test results in terms of time taken

In summary, the test results show that the system performance of DSAS does not grow exponentially with the number of activities or with the number of changes. As shown in Appendix-G, I estimate that the worst-case computational complexity of DSAS is O(n^3), where O is the approximate running time of DSAS, measured as a function of the number of activities, *n*, in a schedule. However, since under 3 changes at a time is common, the common computational complexity is O(n^2), as shown in Figures 6-10 and 6-11. The linear plots show the number of messages or time taken in the cases of no change in each schedule. The quadratic plots show the number of messages or time taken in the cases of three changes in each schedule. The cubic plots show the number of messages or time taken in the cases of maximum changes in each schedule.

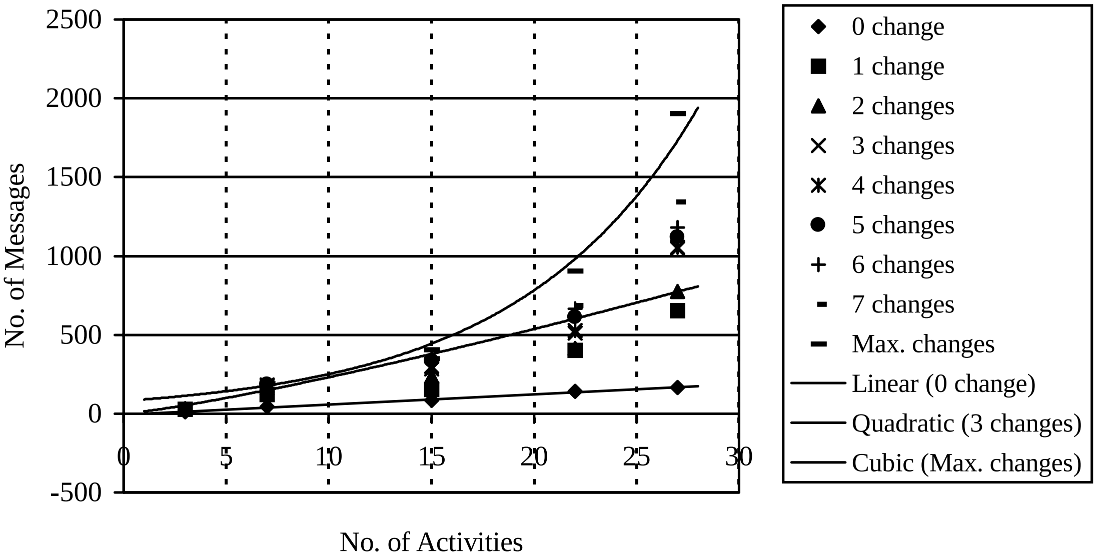


Figure 6-10. Test results in terms of number of messages

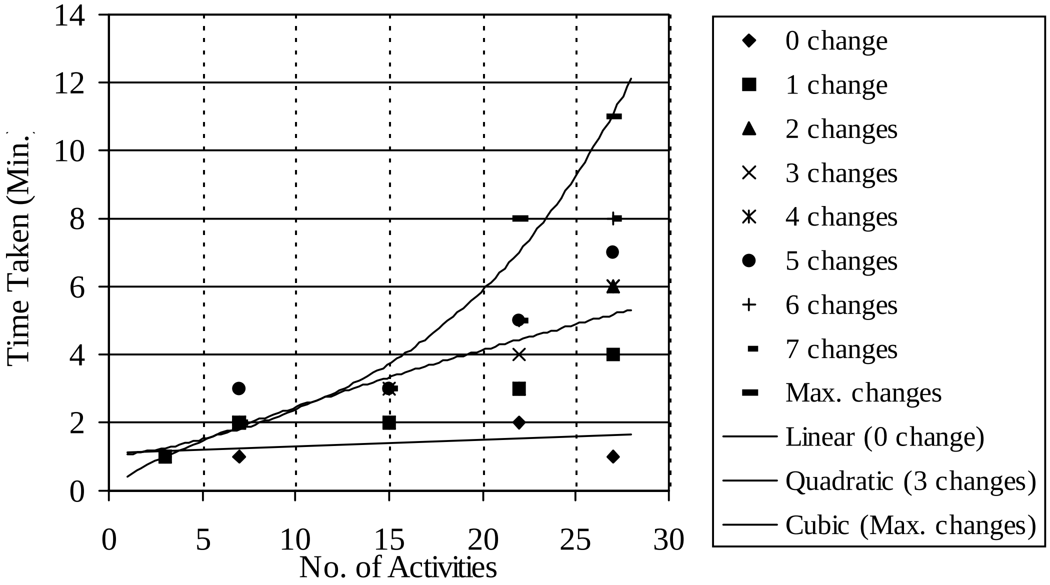


Figure 6-11. Test results in terms of time taken

A few limitations of DSAS also became more evident in this phase of testing. For example:

- **DSAS is only applicable to "balanced" schedules**. A "balanced" schedule is a schedule in which paths between two nodes have the same number of nodes if there are multiple paths. Otherwise, DSAS stops due to "deadlocks." I can modify "unbalanced" schedules by adding 0-duration dummy nodes.

- **Every schedule should have one start node and one finish node**. Otherwise, DSAS stops due to "deadlocks." I can easily modify schedules without these by adding 0-duration start and end nodes.

Figure 6-12 shows an example of "balanced" schedule. This schedule has two paths between Node1 and Node 6: 1-2-4-6 and 1-2-5-6. These two paths have the same number of nodes (4).

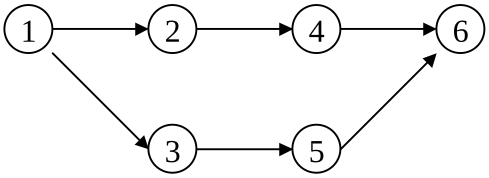


Figure 6-12. Balanced schedule

Figure 6-13 shows an example of "unbalanced" schedule. Node 4 is missing from Figure 6-12. Therefore, a path (1-3-5-6) has one more node than the other path (1-2-6).

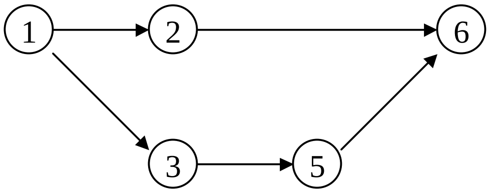


Figure 6-13. Unbalanced schedule

In this unbalanced schedule, DSAS stops due to “deadlocks.” Deadlocks happen in the following situation: Node 1 is infinitely waiting for a “reply-cost” message from Node 3 and Node 6 is infinitively waiting for an “accept/reject” message from Node 5, as shown in Figure 6-14.

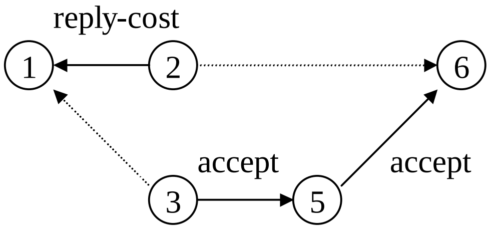


Figure 6-14. Deadlock in unbalanced schedule

I can modify “unbalanced” schedules by adding 0-duration dummy nodes. In this case I add Node 4 to convert an “unbalanced” schedule to a “balanced” schedule, as shown in Figure 6-15.

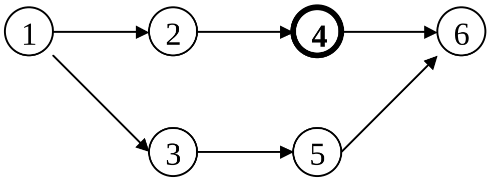


Figure 6-15. Making balanced schedule

Inserting dummy activities can minimize these limitations, but identifying these situations before using DSAS is additional computational overhead. I need to check every node and relationship to identify unbalanced paths in the schedule. Like forward propagation, the first node is 0 and adds one (+1) to its relationships to the succeeding activities. When two or more relationships come to one node, the node checks if the numbers are the same. If the numbers are the same, the schedule is balanced, as shown in Figure 6-16.

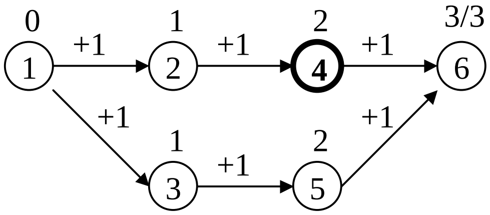


Figure 6-16. Checking balanced schedule

Therefore, the estimated computational complexity for identifying these situations is $O(n+m)$, where n is the number of activities and m is the number of relationships in a schedule.

6.4 SUMMARY OF EVALUATION

This chapter demonstrated the effectiveness of the ABCN methodology through evaluation tests. It compared two centralized coordination methodologies used in current practice to the ABCN methodology. I demonstrated through mathematical proofs (see Appendix-E) that the ABCN methodology always finds a solution that is better than or equal to those of two centralized coordination methodologies. Charrette tests demonstrated that DSAS produces the lower-cost results faster than manual centralized processes. Results of experimental tests with different schedules showed that DSAS finds a solution in a reasonable time.

CHAPTER 7

CONCLUSIONS

In this dissertation, I have introduced a framework for distributed coordination of project schedule changes (DCPSC). The framework emphasizes the distributed nature of coordinating project schedule changes, particularly subcontractors' resource-driven schedule coordination. In the framework, subcontractors have their own activities and schedules for the activities. Subcontractors optimize their schedules based on their available resources and, as a result, schedule conflicts arise.

The dissertation introduces a methodology for negotiation called the agent-based compensatory negotiation (ABCN) methodology. The methodology assumes that subcontractors are competitive and they need monetary compensation for disadvantageous agreements. The methodology preserves the sequence logic in the original project plan, and guarantees that subcontractors always reach a consensus, since the methodology uses the project plan to guide the negotiations.

The research presented in this dissertation shows a distributed approach to coordination of project schedule changes and demonstrates agent-based compensatory negotiation as a vehicle for enabling this approach. The approach emphasizes the distributed nature of coordination of subcontractors' resource-driven project schedule changes and the autonomy of subcontractors while it provides effective methods for coordinated exchange of information and negotiation protocols and a negotiation methodology for resolving schedule conflicts. This in itself is a significant departure from prior and recent research, particularly in the area of construction project planning and scheduling, which has traditionally attempted to centralize the coordination process to enhance a project schedule.

I tested the framework by developing a prototype of the Distributed Subcontractor Agent System (DSAS) that implemented the framework in a multi-agent environment. I compared the project schedules simulated on the prototype of the DSAS with those of two centralized coordination methodologies used in current practice, and concluded that that DCPSC produced a solution that is better than or equal to the initial solution, without revealing the subcontractor's private information. DCPSC performed better than or equal to either of the two centralized coordination methodologies.

I conducted a series of charrette tests that show DCPSC produced a solution that is better than or equal to the initial solution, without the need for subcontractors to reveal private information, and DSAS is faster, more accurate, and more usable than conventional manual processes. These comparison studies, experimental tests, and charrette tests provide evidence of the power, generality, and practical value of my work. Consequently, the results of the research contribute to the current knowledge of construction project planning and scheduling and distributed coordination of complex systems. I also measured the system performance of DSAS by comparing results of several different schedules, and concluded that DSAS finds a solution in a reasonable time.

I conclude that the DCPSC framework enhances the project network schedule (in terms of lowering the sum of subcontractors' costs associated with their resource constraints) by rescheduling the project subject to the precedence relationship among project activities in cases of changes in subcontractors' resource availabilities. I also conclude that the proposed agent-based compensatory negotiation methodology facilitates the distributed coordination of project schedule changes by enabling subcontractors to compensate the affected subcontractors for disadvantageous agreements (see Section 1.5); by allowing subcontractors to identify and resolve schedule conflicts in a tightly coupled network of related activities (see Section 1.5); and by enabling subcontractors to maintain work logic and ensure convergence of distributed coordination (see Section 1.5). Section 7.1 will review the contributions.

This research presents a new agent-based distributed approach for resolving schedule conflicts in project schedules. The distributed subcontractor agent system implemented in this research, together with schedule-change options, suggests that this new approach enables proactive coordination of various conflicts by subcontractors. Thus, this research would help construction project participants to increase efficiency in their resource use in a distributed manner, which will lead to successful completion of whole projects. Section 7.2 of this chapter will discuss the practical demonstrations of this research.

To make solid contributions and validate those contributions within a reasonable period of time, I limited the focus of my study in several ways. For example, I limited the contributions to the coordination of project schedule changes after the general contractor makes a master schedule and assigns parts of the master schedule to subcontractors. Section 7.3 of this chapter discusses the current limitations of the research accomplishments. Section 7.4 of this chapter suggests several directions for future research.

This research can improve the competitive performance of construction projects to improve schedules, enhance resource utilization, and support cooperative relationships among subcontractors. General contractors as well as subcontractors get benefits from employing the research results. Industry can apply this research to many important areas of project management. Section 7.5 of this chapter explains the value of the research to industry.

7.1 SUMMARY OF CONTRIBUTIONS

In general, this research contributes to the field of construction project management and multi-agent systems through the introduction of a distributed coordination framework for project schedule changes by subcontractors in a construction project, the investigation of an agent-based compensatory negotiation methodology for distributed coordination of project schedule changes, and the development of a prototype of a multi-agent system for distributed coordination of project schedule changes. The specific contributions answers the research questions (see Section 2.4) as follows:

- **Formal problem definition for the distributed coordination of project schedule changes:** I defined the objective function of the distributed coordination of project schedule changes. This shows a new way to enhance the global outcome while pursuing individual incentives.

- **Formal definitions for the agent-based compensatory negotiation methodology:** I defined utility of timing as real money. I also defined schedule-change options to represent the impacts on project schedule of resource constraints.

- **Negotiation protocols and algorithms of various message-handling mechanisms for subcontractor agents.** I defined negotiation protocols and algorithms for subcontractor agents so that subcontractor agents can negotiate with other agents.

- **Message-handling mechanisms that use the project plan for coordination of message passing among agents:** I exploited the sequence logic in the project plan for agents to coordinate message passing and to ensure successful completion of distributed computation.

- **Linking CPM to agent-based distributed negotiation:** I have shown how the subcontractors' individual utility function for resource allocation over time can be rigorously combined with the critical path method (CPM) to propagate changes in individual activities to the activities performed by other subcontractors. This allows researchers to go beyond a simplistic representation of the "utility of timing" to a rigorous and operational representation of time. This is a key contribution from linking CPM to agent-based distributed negotiation.

- **A multi-agent system for distributed coordination of project schedule changes that demonstrates various aspects of the framework:** I developed a prototype of the Distributed Subcontractor Agent System (DSAS). The comparison test results on a prototype implementation of DSAS showed that DCPSC produced a solution that is better than or equal to the initial solution, without revealing any subcontractor's private information. DCPSC performed better than or equal to either of two centralized coordination methodologies. A series of charrette tests also demonstrated that DSAS produces the lower-cost results faster than manual centralized processes. Experimental test results showed that DSAS scales in a way that makes its use feasible in real projects.

Thus, this research provides a distributed coordination methodology that can improve interaction and collaboration among agents and people.

7.2 PRACTICAL DEMONSTRATIONS

This section discusses benefits of the distributed coordination of project schedule changes (DCPSC) compared to the centralized coordination methodologies, which most practitioners employ in current project management practice. DCPCS has benefits as noted below:

- **DCPSC handles change problems proactively.** Most centralized coordination methodologies try to solve already-incurred delay problems and, therefore, lose many opportunities to find a solution proactively. DCPCS identifies change problems and opportunities beforehand and finds a solution proactively.

- **DCPSC finds a more equitable solution among all participants in a project**. Most centralized coordination methodologies make the general contractors and subcontractors together pay for a delay, but the sources of trouble are difficult to define, which leads to an unfair cost distribution. DCPCS can identify the sources of possible delays beforehand and charge the sources when they decide to delay their activities. DCPCS does not ask the central coordinator or the owners for extra costs for subcontractors' faults, and costs will be distributed in an equitable manner so that disputes among participants can be resolved easily before becoming worse.

- **DCPSC promotes distributed decision-making to the level where accurate information exists**. The general contractors employing the centralized coordination methodologies usually do not have the needed information for coordinating subcontractors because most of the needed information, such as resource and cost information, is kept within subcontractors. The general contractors could get the needed information through bids, but this would lead to opportunistic behaviors by subcontractors. DCPCS provides a methodology for subcontractors to make their decisions in collaboration with other subcontractors who hold accurate local information. Accordingly, most computation is carried out at distributed sites with relatively light message traffic among them.

- **DCPSC frees central coordinators from having to control changes directly**. This means that the central coordinator takes less time and uses fewer resources to control the project. More importantly, it means a distributed system scales where a centralized system will not. There is also a revolution inherent in my work. The central coordinator no longer has to control directly but rather can do so indirectly by incentives (bonuses and penalties) in the contracts with the subcontractors. My system allows subcontractors to work with each other in their own best interests, since it allows bonuses and penalties to be included in the calculation of "extra costs."

The significance of these practical demonstrations is that DCPSC employs the agent-based approach, which is a powerful tool to explore and exploit the opportunities offered by the distributed approach. Software agents have capabilities of fast communication with each other over the Internet, which supports project participants without geographic restrictions. Thus, this research can help construction project participants to increase the efficiency of their resource utilization in a distributed manner, and thus enhance successful completions of whole projects.

7.3 PRESENT LIMITATIONS OF RESEARCH

DCPSC has some present limitations, as noted below, some of which are shared with centralized coordination methodologies:

- **DCPSC does not guarantee an optimal solution for the changes**. Considering all alternatives in DCPSC is infeasible in a large network, because the number of alternatives grows exponentially with the number of succeeding activities, even with only a few options. Therefore, my methodology requires subcontractors to make quick decisions, i.e., choosing the best option of all, without considering all possibilities, thus the decision might eliminate a globally optimal solution. However, given that the needed information is available, centralized coordination methodologies can use optimization tools like integer programming to find an optimal solution in some cases, but these optimization tools are not applicable to the distributed coordination situation, where the needed information for optimization is distributed and private.

- **DCPSC does not allow changes of work logic in a project schedule**. Because of the limited capability of its subcontractor agents, the current DCPSC can only evaluate impacts of proposed delays. The impacts are found by delaying affected activities. In the project context, some activities can shorten their durations for their purposes and the saved duration can be exploited for enhancing resource utilization of other activities. DCPSC does not allow subcontractors to change work logic in a project schedule, but only finds a better schedule with the fixed work logic.

- **DCPSC does not provide direct relationships between resource constraints and schedule-change options**. There are too many parameters to determine the relationships between resource constraints and schedule-change options. Therefore, DCPSC uses a set of schedule-change options given by subcontractors.

7.4 FUTURE RESEARCH

Researchers need to do the following future research to overcome limitations stated above:

- **Apply heuristic distributed search techniques for producing an optimal or near-optimal solution.** A better approach to produce an optimal or near-optimal solution would take into account the reduction of the distributed search space effectively.

- **Develop a more general distributed coordination methodology for allowing changes in work logic in project schedules.** A more general distributed coordination methodology would take into account a different representation of activities, such as input and output requirements for each activity. The input/output requirements that allow it to be dynamic is an advance over the traditional project CPM model that only includes fixed precedence relationships between activities.

- **Accommodate "counter-offers" for richer negotiation protocols.** Richer negotiation protocols would take into account mechanisms to detect the termination of negotiations because accommodating protocols like "counter-offers" would lead to infinite loops.

- **Investigate direct relationships between resource constraints and schedule-change options.** Direct relationships between resource constraints and schedule-change options would take into account the investigation and parameterization of many factors that would relate resource constraints to schedule-change options, such as units, fixed/variable, timing, and upper limits of resources.

- **Extend the framework to allow distributed coordination/negotiation for variables other than time.** Researchers can extend this research to help subcontractors resolve conflicts among competitive actors in other areas, such as mechanical, electrical, and plumbing (MEP) coordination and workspace management**,** e.g., HVAC, mechanical, electrical contractors negotiate for access to space for straight pipe or duct runs in ceiling planning, etc. These involve spatial as well as temporal constraints between the activities performed by different subcontractors, and would require conceptual extensions to time constraints in this research.

7.5 VALUE TO INDUSTRY

This research could impact the industry by improving the competitive performance of construction projects to improve schedules, enhance resource utilization, and support cooperative relationships among subcontractors. My research provides a foundation to develop a distributed scheduling and control system that helps subcontractors represent resource requirements of activities and consider the timing of activities; helps them identify and analyze their resource constraints in a given schedule; helps them predict the behavior of activities; helps them incorporate the schedule impacts of those behaviors; and helps them coordinate their different scheduling perspectives by working together toward a better solution.

This approach has the potential to free the central coordinator from having to control schedule changes directly while getting better schedules. The project schedule gets better because all the subcontractors have committed to their parts of the project schedule. Thus, this research will help project participants increase efficiency of their resource uses in a distributed manner, which should lead to more successful completions of whole projects.

Industry can apply this research to improve the efficiency of coordinating complex distributed systems, such as electronic supply-chain and e-markets, where subcontractors interact and transact with other subcontractors within supply-chain networks with the assistance of software agents. Subject to external changes, subcontractors could use e-markets to trade not only their timing with other subcontractors through multi-agent negotiation, but also their activities with the similarly qualified specialty contractors, which may be in better positions to execute the activities in a given time. Agent-based e-markets will allow specialty contractors to trade their activities with other specialty contractors to improve their resource utilization.

There has been a long effort to improve resource utilization in the construction industry, whose project nature is a centralized constraint-satisfaction problem. Since the subcontractors in current practice manage the resources, subcontractors should play important roles in improving their resource utilization in a distributed manner. Agent-based e-markets leverage agent technology to facilitate activity reallocation among specialty contractors, resulting in better resource utilization. Through transactions in the e-markets, agents will trade activities on behalf of human specialty contractors.

An agent-based distributed approach has the potential to impact the performance of the construction industry significantly, through Internet-based project management, distributed control of large-scale engineering systems, and management of complex distributed systems. As project participants are provided with software agents that can communicate over the Internet, Internet-based project management, including routing of construction material and information flows, and dispatching of pickup and delivery systems, will become possible. Since the software agent can represent any participant in construction projects, researchers can develop the distributed coordination framework further for the real-time monitoring and control of large-scale civil and environmental engineering systems, e.g., bridges and water systems. Researchers could even further extend the distributed coordination methodology to the management of complex distributed systems, such as air traffic control, highway traffic operations, and disaster response.

Although the focus of this research has been placed on the construction industry, this approach is valid for any project where a system integration (“prime”) contractor develops a project schedule, but subcontracts much of direct work to other firms, as in aerospace or custom network solution projects.

APPENDIX-A. BIBLIOGRAPHY

Adler, M. R., Davis, A. B., Weihmayer, R., and Worrest, R. W. (1989). "Conflict-resolution strategies for nonhierarchical distributed agents." *Distributed Artificial Intelligence,* vol. 2., Gasser, L. and Huhns, M. eds., Morgan Kaufman, San Francisco, 139-161.

Albayrak, S., and Wieczorek, D. (1999). "JIAC-a toolkit for telecommunication applications." *Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication Applications (IATA'99),* Stockholm, Sweden, Springer-Verlag, Berlin, Germany, 1-18

Antill, J. M., and Woodhead, R. W. (1990). *Critical Path Methods In Construction Practice*, 4th ed., Wiley, New York.

Barbuceaue, M., and Fox, M. (1996). "The Agent Building Shell: A tool for building enterprise multi-agent systems." *Canadian Artificial Intelligence*, Canadian Society for Computational Studies of Intelligence, Toronto, Canada, 40, 9-11.

Ben-Shaul, I., Gazit, H., Gidron, Y., Holder, O., and Lavva, B. (1999). "FarGo: A system for mobile component-based application development." *Proceedings of The Twenty-First International Conference on Software Engineering (ICSE '99)*, Los Angeles, CA, ACM, New York, 658-659.

Bölöni, L., and Marinescu, D. C. (2000). "An object-oriented framework for building collaborative network agents." *Intelligent Systems and Interfaces*, A. Kandel, K. Hoffmann, D. Mlynek, and N.H. Teodorescu, eds., Kluwer Publishing, Dordrecht, the Netherlands, 31-64.

Chan, W., Chua D., and Kannam, G. (1996). "Construction resource scheduling with genetic algorithms." *Journal of Construction Engineering and Management,* ASCE, 122(2), 125-132.

Chen, Y., Peng, Y., Finin, T., Labrou, Y., Cost, S., Chu B., Yao, J., Sun, R., and Wilhelm, B. (1999). “A negotiation-based multi-agent system for supply chain management.” *Proceedings of Agents 99 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain*, Seattle, Washington, ACM Press, New York, 15-20.

Choo, H. J., and Tommelein, I. D. (2000). “Interactive coordination of distributed work plan.” *Proceedings of the Sixth Construction Congress*, Orlando, FL, ASCE, 11-20.

Choo, H. J., Tommelein, I. D., Ballard, G., and Zabelle, T. D. (1999). “WorkPlan: Constraint-based database for work package scheduling.” *Journal of Construction Engineering and Management*, ASCE, 125(3), 151-160.

Clayton, M. J., Kunz, J. C., and Fischer, M. A. (1998). “The Charrette test method.” *Technical Paper*, No. 120, CIFE, Stanford Univ., Stanford, Calif.

Cliff, D., and Bruten, J. (1998). “Simple bargaining agents for decentralized market-based control.” *Technical Report,* HPL-98-17, Hewlett Packard Laboratories, Bristol.

Clough, R. H., and Sears, G. A. (1991). *Construction Project Management*, 3rd ed., John Wiley & Sons, Inc., New York.

Davis, E. W. (1968). “An exact algorithm for the multiple constrained resource project scheduling program.” PhD thesis, Department of Administrative Science, Yale University, New Haven, Conn.

Davis, R., and Smith, R. G. (1988). “Negotiation as a metaphor for distributed problem solving.” *Readings in Distributed Artificial Intelligence*, A. H. Bond and L. Gasser, eds., Morgan Kaufmann Publishers, San Francisco, 333-356.

Decker, K. S., and Lesser, V (1992). “Generalizing the partial global planning algorithm.” *International Journal on Intelligent Cooperative Information Systems*, World Scientific Publishing Company, Singapore, 1(2), 319-346.

Decker, K. S., and Lesser, V. (1995). “Designing a family of coordination algorithms.” *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press, Menlo Park, 73-80.

Decker, K. S. (1996). “TAEMS: A framework for Analysis and design of coordination mechanisms.” *Foundations of Distributed Artificial Intelligence*, G. O’Hare and N. Jennings, eds., John Wiley & Sons, Inc., New York, 429-447.

Dijkstra, W., and Sholten, C. S. (1980). “Termination detection for diffusing computation.” *Information Processing Letters*, North-Holland Pub. Co., Amsterdam, 11(1), 1-4.

Durfee, E. H., Lesser, V. R., and Corkill, D. D. (1989). “Trends in cooperative distributed problem solving.” *IEEE Transactions on Knowledge and Data Engineering*, IEEE, 1(1), 63-83.

Durfee, E. H., and Lesser, V. R. (1991). “Partial global planning: A coordination framework for distributed hypothesis formation.” *IEEE Transactions on Systems, Man, and Cybernetics*, IEEE, 21(5), 1176-1183.

Durfee, E. H., and Montgomery, T. A. (1991). “Coordination as distributed search in a hierarchical behavior space.” *IEEE Transactions on Systems, Man, and Cybernetics,* IEEE, 21(6), 1363-1378.

Easa, S. M. (1989). “Resource leveling in construction by optimization.” *Journal of Construction Engineering and Management*, ASCE, 115(2), 302-316.

El-Rayes, K., and Moselhi, O. (1996). “Resource-driven scheduling of repetitive activities on construction projects.” *Construction Management and Economics,* E & FN Spon, London, 16, 433-446.

El-Rayes, K., and Moselhi, O. (2001). “Optimizing resource utilization for repetitive construction projects.” *Journal of Construction Engineering and Management*, ASCE, 127(1), 18-27.

Ephrati, E., and Rosenschein, J. S. (1996). “Deriving consensus in multi-agent systems.” *Artificial Intelligence*, Elsevier Science BV, Amsterdam, 87(1-2), 21-74.

Feldman, A. M. (1980). *Welfare Economics and Social Choice Theory,* Martinus Nijhoff Publishing, Boston.

Finin, T., Fritzson, R., McKay, D., and McEntire, R., (1994). “KQML as an agent communication language.” *Proceedings of the Third International Conference On Information and Knowledge Management*, ACM Press, New York, 456-463.

FIPA Specification. (2000). “Agent Communication Language.” The Foundation for Intelligent Physical Agents.

Fondahl, J. M. (1961). “A non-computer approach to the critical path method for the construction industry.” *Technical Report, No. 9,* Construction Institute, Dept. of Civ. Engrg., Stanford University, Stanford, Calif.

Fondahl, J. W. (1991). “The development of the construction engineer: Past progress and future problems.” *Journal of Construction Engineering and Management,* ASCE, 117(3), 380-392.

Glossary. (1999). *Multiagent Systems*: *A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, ed., MIT Press, Cambridge.

Gomes, C.P., Tate A., and Thomas, L. (1994). “Distributed scheduling framework.” *Proceedings of the International Conference on Tools with Artificial Intelligence*, IEEE Press, Piscataway, 49-55.

Graham, J., Windley, V., McHugh, D., McGeary, F., Cleaver, D., and Decker, K. (2000). “A programming and execution environment for distributed multi agent systems.” *Proceedings of Workshop on Infrastructure for Scalable Multi-agent Systems at the Fourth International Conference on Autonomous Agents*, Barcelona, ACM Press, New York.

Harris, R. B. (1990). “Packing method for resource leveling (pack).” *Journal of Construction Engineering and Management*, ASCE, 116(2), 331-350.

Hegazy, T. (1999). “Optimization of resource allocation and leveling using genetic algorithms.” *Journal of Construction Engineering and Management*, ASCE, 125(3), 167-175.

Hegazy, T., Shabeeb, A. K., Elbeltagi, E., and Cheema, T. (2000). “Algorithm for scheduling with multi-skilled constrained resources.” *Journal of Construction Engineering and Management*, ASCE, 126(6), 414-421.

Hiyassat, M. A. (2000). “Modification of minimum moment approach in resource leveling.” *Journal of Construction Engineering and Management*, ASCE, 126(4), 278-284.

Iglesias, C. A., González, J. C., and Velasco, J. R. (1995). “MIX: A general purpose multiagent architecture.” *Proceedings of the IJCAI'95 Workshop on Agent Theories, Architectures and Languages,* Montréal, Canada, ACM Press, 216-224.

Jeon, H. (2000). “Dynamic constraint management in collaborative design.” PhD thesis, Dept. of Mechanical Engineering, Stanford Univ., Stanford, Calif.

Jeon, H., Petrie, C., and Cutkosky, M. R. (2000). “JATLite: A Java agent infrastructure with message routing.” *IEEE Internet Computing*, IEEE, 4(2), 87-96.

Jin, Y., and Levitt, R. E. (1993). “i-AGENTS: Modeling organizational problem solving in multi-agent teams.” *Intelligent Systems in Accounting, Finance and Management*, John Wiley & Sons, Ltd., New York, 2, 247-270.

Kawamura, T., Tahara, Y., Hasegawa, T., Ohsuga, A., and Honiden, S. (2000). “Bee-gent: Bonding and encapsulation enhancement agent framework for development of distributed systems.” *Systems and Computers in Japan*, Scripta Technica, 31(13), 42-56.

Khedro, T., Genesereth, M.R., and Teicholz, P.M. (1993). "Agent-based framework for integrated facility engineering." *Engineering with Computers*, Springer Verlag, New York, 9(2), 94-107.

Khedro, T. (1996). "A distributed problem-solving approach to collaborative facility engineering." *Advances in Engineering Software*, Elsevier Applied Science, Barking, Essex, England, 25, 243-252.

Koo, C. C. (1987). "A distributed model for performance systems." PhD thesis, Dept. of Civil Engineering, Stanford Univ., Stanford, Calif.

Labrou, Y., and Finin, T. (1997). "A proposal for a new KQML specification." *TR CS-97-03*, Computer Science and Electrical Engineering Department, Univ. of Maryland, Baltimore.

Leu, S., and Yang, C. (1999). "GA-Based multicriteria optimal model for construction scheduling." *Journal of Construction Engineering and Management*, ASCE, 125(6), 420-427.

Malone, T. W., Fikes, R. E., and Howard, M. T. (1988). "Enterprise: A market like task scheduler for distributed computing environment." *The Ecology of Computation*, B. A. Huberman, ed., Elsevier Science BV, Amsterdam, 177-205.

Martin, D. L., Cheyer, A. J., and Moran, D. B. (1999). "Open Agent Architecture: A framework for building distributed software systems." *Applied Artificial Intelligence,* Taylor & Francis Ltd, London, 13(1-2), 91-128.

Martinez, J., and Ioannou, P. G. (1993). "Resource leveling based on the modified minimum moment heuristics." *Computing in Civil and Building Engineering*, ASCE, New York, 287-294.

Mattila, K. G., and Abraham, D. M.. (1998). "Resource leveling of linear scheduling using integer linear programming." *Journal of Construction Engineering and Management*, ASCE, 124(3), 232-244.

Minar, N., Gray, M., Poop, O., Krikorian, R., and Maes, P. (2000). “Hive: Distributed agents for networking things.” *IEEE Concurrency,* IEEE, 8(2):24-33.

Moselhi, O., and El-Rayes, K. (1993). “Scheduling of repetitive projects with cost optimization.” *Journal of Construction Engineering and Management,* ASCE, 119(4), 681-697.

Moselhi, O., and Lorterapong, P. (1993). “Near optimal solutions for resource-constrained scheduling problems.” *Construction Management and Economics*, E & FN Spon, London, 11(4), 293-303.

Mudgal, C., and Vassieva, J. (1999). “Negotiation among agents in a multi-agent environment supporting peer-help: A decision-theoretic approach.” *Proceedings of Agents 99 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain,* Seattle, Washington, ACM Press, New York, 35-43.

Neiman, D., Hildum, D., Lesser, V., and Sandholm, T. (1994). “Exploiting meta-level information in a distributed scheduling system.” *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI, 394-400.

Neiman, D., and Lesser, V. (1996). “A cooperative repair method for a distributed scheduling system.” *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, Edinburgh, Scotland, AAAI, Menlo Park, 166-173.

Nwana, H. S., Ndumu, D. T., Lee, L. C., and Collis, J. C. (1999). “ZEUS: A toolkit for building distributed multiagent systems.” *Applied Artificial Intelligence*, Taylor & Francis, London, 13(1-2):129-85.

Oberlender, G. D. (1993). *Project Management for Engineering and Construction,* McGraw-Hill, New York.

O’Brien, W., Fischer, M. A., and Jucker, J. V. (1995). “An economic view of project coordination.” *Construction Management and Economics*, E & FN Spon, London, 13(5), 393-400.

O'Brien, W. J. (1998). "Capacity costing approaches for construction supply-chain management." PhD thesis, Dept. of Civil Engineering, Stanford Univ., Stanford, Calif.

O'Brien, W., and Fischer, M. A. (2000). "Importance of capacity constraints to construction cost and schedule." *Journal of Construction Engineering and Management*, ASCE, 126(5), 366-373.

Petrie, C., Goldmann, S., and Raquet, A. (1998). "Agent-based project management." *Technical Report, #19981118,* Center for Design Research, Stanford Univ., Stanford, Calif.

Petrie, C. (1996). "Agent-based engineering, the Web, and intelligence." *IEEE Expert*, IEEE, 11(6), 24-29.

Poslad, S., Buckle, P., and Hadingham R. (2000). "The FIPA-OS agent platform: Open source for open standards." *Proceedings of the Fifth International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, Manchester, UK, Practical Application Company, Blackpool, UK, 355-368.

Rosenschein, J. S., and Zlotkin. G. (1994). *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers,* MIT Press, Cambridge.

Rosenschein, J. S., and Zlotkin. G. (1998). "Designing conventions for automated negotiation." *Readings in Agents*, M. N. Huhn and M. P. Singh, eds., Morgan Kaufmann Publishers, San Francisco.

Russell, A., and Dubey, A. (1995). "Resource leveling and linear scheduling." *Computing in Civil Engineering*, ASCE, New York, 1134-1141.

Sandholm, T. (1993). "An implementation of the contract net protocol based on marginal cost calculations." *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI, Menlo Park, 256-262.

Sandholm, T., and Lesser, V. (1995). "Issues in automated negotiation and electronic commerce: Extending the contract net framework." *Proceedings of the First International Conference on Multiagent Systems (ICMAS-95)*, San Francisco, Calif., MIT Press, Cambridge, 328-335.

Seibert, J. E., and Evans, G. W. (1991). "Time-constrained resource leveling." *Journal of Construction Engineering and Management*, ASCE, 117(3), 503-520.

Sen, S., and Durfee, E. H. (1996). "A contracting model for flexible distributed scheduling." *Annals of Operations Research*, J.C. Baltzer, Basel, Switzerland, 65, 195-222.

Sen, S., Haynes, T., and Arora, N. (1997). "Satisfying user preferences while negotiating meetings." *International Journal of Human-Computer Studies*, Academic Press, London, 47, 407-427.

Sen, S., and Durfee E. H. (1998). "A formal study of distributed meeting scheduling." *Group Decision and Negotiation*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 7, 265-289.

Shah, K. A., and Baugh, J. W. (1993). "Optimal resource leveling using integer-linear programming." *Computing in Civil and Building Engineering*, ASCE, New York, 501-508.

Shoham, Y., and Tanaka, K. (1997). "A dynamic theory of incentives in multi-agent systems." *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence, Inc., Menlo Park, 626-631.

Smith, R. G. (1980). "The contract net protocol: high level communication and control in a distributed problem solver." *IEEE Transactions on Computers*, IEEE, C-29, 1104-1113.

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

Son, J., and Skibniewski, M. J. (1999). “Multiheuristic approach for resource leveling problem in construction engineering: Hybrid approach.” *Journal of Construction Engineering and Management*, ASCE, 125(1), 23-31.

Sycara, K. (1989). “Multi-agent compromise via negotiation.” *Distributed Artificial Intelligence,* vol. 2., Gasser, L. and Huhns, M. eds., Morgan Kaufman, San Francisco, 119-137.

Sycara, K., Roth, S., Sadeh, N., and Fox, M. (1991). “Distributed constrained heuristic search.” *IEEE Transactions on Systems, Man, and Cybernetics*, IEEE press, 21(6), 1446-1461.

Sycara, K., and Pannu, A. (1998). “The RETSINA multiagent system: Towards integrating planning, execution and information gathering.” *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, ACM Press, New York, 350-351.

Tommelein, I. D., and Ballard, G. (1997). “Coordinating specialists.” *Technical Report,* No. 97-8, Dept. of Civil and Environmental Engineering, Univ. of Calif., Berkeley, Calif.

Varian, H. R. (1978). *Microeconomic Analysis*. W.W. Norton & Company, New York.

Walsh, W. E., and Wellman, M. P. (1998). “A market protocol for decentralized task allocation.” *Proceeding of the Third International Conference on Multiagent Systems,* IEEE Computer Society, Los Alamitos, California, 325-332.

Wellman, M. P. (1993). “A market-oriented programming environment and its application to distributed multicommodity flow problems.” *Journal of Artificial Intelligence Research*, Morgan Kaufmann Publishers, San Francisco, 1, 1-23.

Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. (1992). “Distributed constraint satisfaction for formalizing distributed problem solving.” *Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, IEEE Press, Piscataway, NJ, 614-621.

Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. (1998). “Distributed constraint satisfaction problem: Formalization and algorithms.” *IEEE Transactions on Knowledge and Data Engineering*, IEEE, 10(5), 673-685.

APPENDIX-B. MULTI-LINKED NEGOTIATION PROTOCOLS

In this section, I express Multi-linked negotiation (MLN) protocols based on Knowledge Query and Manipulation Language (KQML) (Finn et al 1993), along with extended performatives and a restrictive format for the message content consisting of field name and value pairs specific to the predefined performatives. I used BNF but avoided special symbols for repetition for readability. Strings are not case-sensitive.

I classified the multi-linked negotiation protocols into three classes: human interaction, negotiation, and negotiation control. The human interaction performatives allow a human subcontractor to provide input data to its agent and an agent to inform its subcontractor of the current status of negotiation. The negotiation performatives facilitate the actual compensatory negotiation processes. The negotiation control performatives manage the states of negotiation processes.

B.1 SYNTAX

B.1.1 Outer Syntax

```
KQML-MLN-EPLMessage := "(" <performative> <mandatory> <otherfields> <content> ")"
performative := <string>
mandatory := ":" "Sender" <name> ":" "Receiver" <name> ":" "Language" "KQML"
otherfields := ":" <name> <value> | ":" <name> <value> <otherfields> | NIL
content := ":" "Content" "(" <content_fields> ")"
content_fields := <name> "|" <value> "&" <content_fields> | <name> "|" <value> "&" | NIL
value := <string> | "(" <string> <value> ")"
name := <string>
string := ASCII string
```

B.1.2 Semantics

This is the general form of a KQML message understood by MLN-compliant agents and defines the MLN general message syntax.

B.2 HUMAN INTERACTION PROTOCOLS

These performatives are formal interfaces needed for a human subcontractor to provide input data to its agent and for an agent to inform the subcontractor of the results of negotiation.

B.2.1 Input

This is a MLN protocol message with <performative> “input”. The <receiver> is agent and the <sender> is a human GUI.

- Syntax

```
:content := “(“AGENT”(“<projStartDate>” “<projEndDate>” “<projPenalty>” “
                <subName>”))”

or
:content := “(“ACTIVITY”(“<activityName>” “ <startDate>” “
                <endDate>”){“”(“<preSubName>” “
                <preActivityName>”)””}{“”(“<sucSubName>” “
                <sucActivityName>”)””}{“”(“<optionStartDate>”
                “<optionEndDate>” “ <optionExtraCost>”)””})”
```

- Semantics

This performative allows a human subcontractor to provide its agent with agent information and activity information, including precedence and schedule-change options.

- Field description

projStartDate: Project start date
projEndDate: Project end date
projPenalty: project delay penalty
subName: Name of human GUI
activityName: Activity name

startDate: Activity start date

endDate: Activity end date

preSubName: Name of subcontractor agent that has a preceding activity

preActivityName: Name of preceding activity

sucSubName: Name of subcontractor agent that has a succeeding activity

sucActivityName: Name of preceding activity

optionStartDate: Alternative activity start date of a schedule-change option

optionEndDate: Alternative activity end date of a schedule-change option

optionExtraCost: Extra cost for a schedule-change option

- Example

```
(input
:sender     kim
:receiver   SubA
:content    (AGENT(1 12 2000.0 kim))
```

This example shows that a human subcontractor (kim) provides it agent (SubA) with agent information that consists of project start date (Day 1), project end date (Day 12), project delay penalty ($2,000), and name of human subcontractor (kim).

```
(input
:sender     kim
:receiver   SubA
:content    (ACTIVITY(E 8 10){(SubB B)(SubC C)}{(SubA G)}{(8 10 0)(9 10
            640)(10 12 0)}))
```

This example shows that a human subcontractor (kim) provides it agent (SubA) with activity information that consists of a tuple of activity name (E), activity start date (Day 8), activity end date (Day 10); tuples of names of subcontractor agents (SubB and SubC) and name of preceding activities (C and G); a tuple of name of subcontractor agent (SubA), name of a succeeding activity (G); tuples of alternative activity start dates (Day 8, Day 9, and Day 10), alternative activity end dates (Day 10, Day 10, and Day 12), and extra costs for a schedule-change options ($0, $640, and $0).

B.2.2 Ready

This is a MLN protocol message with <performative> “ready”. The <receiver> is agent and the <sender> is a human GUI.

- Syntax

```
:content := “(“<activityName>”)”
```

- Semantics

This performative allows a human subcontractor to inform its agent that input is finished on the specific activity.

- Field description

activityName: Activity name

- Example

```
(ready
:sender     kim
:receiver   SubA
:content    (E))
```

This example shows that the human subcontractor (kim) inform it agent (SubA) that input of activity information (E) is finished.

B.2.3 Final

This is a MLN protocol message with <performative> “final”. The <receiver> is a human GUI and the <sender> is an agent.

- Syntax

```
:content := “(“<activityName>” “NewStartDate: “<newStartDate>” “NewEndDate: “
               <newEndDate>” “ActivityBenefit: “<activityBenefit>” “Receivable: “
               “((“<preSubName>” “<preActivityName>” “<receivable>”))”” “
               “Payable: ““((“<sucSubName>” “<sucActivityName>”“<payable>”))”” “
```

“Schedule Options: “”((“<optionStartDate>” “<optionEndDate>” “<optionExtraCost>”)))”

- Semantics

This performative allows an agent to inform the human subcontractor of the final result of negotiation on the specific activity.

- Field description

activityName: Activity name

newStartDate: New activity start date

newEndDate: New activity end date

activityBenefit: Benefit for an activity from the negotiation

preSubName: Name of subcontractor agent that has a preceding activity

preActivityName: Name of preceding activity

receivable: Amount to be received from the subcontractor agent that has a preceding activity

sucSubName: Name of subcontractor agent that has a succeeding activity

sucActivityName: Name of succeeding activity

payable: Amount to be paid to the subcontractor agent that has a succeeding activity

optionStartDate: Changed activity start date of a selected schedule-change option

optionEndDate: Changed activity end date of a selected schedule-change option

optionExtraCost: Extra cost for a selected schedule-change option

- Example

```
(final
:sender     SubA
:receiver   kim
:content    (E NewStartDate: 9 NewEndDate: 10 ActivityBenefit: 0 Receivable:
            ((SubB B 640)) Payable: ( ) Schedule Options: ((8 10 0)(9 10 0)(10 12 0)))
```

This example shows that the agent (SubA) inform its human subcontractor (kim) of the final result of negotiation that the agent agreed to change its schedule-change option for activity (E), which consists of new start date (Day 9); new end date (Day

10); activity benefit ($0); a tuple of preceding agent name (SubB), preceding activity name (B), and receivable ($640); an empty tuple of payable; and tuples of changed activity start dates (Day 8, Day 9, and Day 10), changed activity end dates (Day 10, Day 10, and Day 12), and extra costs for a schedule-change options ($0, $0, and $0). Note that the second extra cost is changed from $640 to $0 because of receivable.

B.3 NEGOTIATION PROTOCOLS

The following performatives are used for facilitating compensatory negotiation processes.

B.3.1 Ask-cost

This is a MLN protocol message with <performative> "ask-cost". The <receiver> is an agent and the <sender> is an agent.

- Syntax

```
:content := "("<sendingSub_activityName>" "<receivingSub_activityName>"
            "<proposedStartDate>")"
```

- Semantics

This performative allows an agent to ask its 'succeeding' agent, which has the succeeding activity, to find out any cost which is incurred by the delay of the proposed start date. This performative makes the receiving activity "flag."

- Field description

sendingSub_activityName: Activity name of sending agent

receivingSub_activityName: Activity name of receiving agent

proposedStartDate: Proposed start date for the activity of receiving agent

- Example

```
(ask-cost
:sender     SubB
:receiver   SubA
:content    (B E 9))
```

This example shows that the agent (SubB) ask its ‘succeeding’ agent (SubA) to find out any cost, which is incurred by the delay of its activity (B), to the succeeding activity (E) with the proposed start date (Day 9).

B.3.2 Reply-cost

This is a MLN protocol message with <performative> “reply-cost”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

```
:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”
            “<cost>”)”
```

- Semantics

This performative allows an agent to reply to its ‘preceding’ agent, which has the preceding activity, with the cost which is incurred by the delay.

- Field description

sendingSub_activityName: Activity name of sending agent

receivingSub_activityName: Activity name of receiving agent

cost: Its own activity’ cost or a sum of succeeding activities’ cost

- Example

```
(reply-cost
:sender     SubA
:receiver   SubB
:content    (E B 640))
```

This example shows that the agent (SubA) replies to its ‘preceding’ agent (SubB) with the cost ($640), which is incurred by the delay of the preceding activity (B), to its activity (E).

B.3.3 Accept

This is a MLN protocol message with <performative> “accept”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent, which has “flag” activity, to accept the cost response from its ‘succeeding’ agent that has the succeeding activity. This means that the agent chooses to delay its activity rather than accelerate it.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(accept
:sender      SubB
:receiver    SubA
:content     (B E))
```

This example shows that the agent (SubB), which has “flag” activity (B), accepts the cost response from its ‘succeeding’ agent (SubA) for the succeeding activity (E). This means that the agent (SubB) chooses to delay its activity (B) rather than accelerate it.

B.3.4 Reject

This is a MLN protocol message with <performative> “reject”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent, which has “flag” activity, to reject the cost response from its ‘succeeding’ agent that has the succeeding activity. This means that the agent chooses to accelerate its activity rather than delay it.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(reject
:sender      SubB
:receiver    SubA
:content     (B E))
```

This example shows that the agent (SubB), which has “flag” activity (B), rejects the cost response from its ‘succeeding’ agent (SubA) for the succeeding activity (E). This means that the agent (SubB) chooses to accelerate its activity (B) rather than delay it.

B.3.5 Confirm

This is a MLN protocol message with <performative> “confirm”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent to confirm the “accept” for its activity from its ‘preceding’ agent that has the preceding activity. This means that its cost reply is accepted, but the final contract is pending.

- Field description

sendingSub_activityName: Activity name of sending agent receivingSub_activityName: Activity name of receiving agent

- Example

```
(confirm
:sender     SubA
:receiver   SubB
:content    (E B))
```

This example shows that the agent (SubA) confirms the “accept” for its activity (E) from its ‘preceding’ agent (SubB) that has the preceding activity (B). This means that its cost reply is accepted, but the final contract is pending.

B.3.6 Renege

This is a MLN protocol message with <performative> “renege”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent to renege the “accept” or “reject” for its activity from its ‘preceding’ agent that has the preceding activity. This means that its cost reply is rejected and the agent has to keep the original schedule.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(confirm
:sender     SubA
:receiver   SubB
:content    (E B))
```

This example shows that the agent (SubA) reneges the “reject” for its activity (E) from the ‘preceding’ agent (SubB) that has the preceding activity (B). This means that its cost reply is rejected and the agent has to keep the original schedule.

B.3.7 Accept-all

This is a MLN protocol message with <performative> “accept-all”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent, which has “active” activity, to accept the cost response from its ‘succeeding’ agent that has the succeeding activity. This means that the agent chooses to delay its activity rather than accelerate it.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(accept-all
:sender      SubB
:receiver    SubA
:content     (B E))
```

This example shows that the agent (SubB), which has “active” activity (B), accepts the cost response from its ‘succeeding’ agent (SubA) that has the succeeding activity (E). This means that the initiating agent (SubB) chooses to delay its activity (B) rather than accelerate it.

B.3.8 Reject-all

This is a MLN protocol message with <performative> “reject-all”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent, which has “active” activity, to reject the cost response from its ‘succeeding’ agent that has the succeeding activity. This means that the agent chooses to accelerate its activity rather than delay it.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(reject-all
:sender      SubB
:receiver    SubA
:content     (B E))
```

This example shows that the agent (SubB), which has “active” activity (B), rejects the cost response from its ‘succeeding’ agent (SubA) that has the succeeding activity (E). This means that the initiating agent (SubB) chooses to accelerate its activity (B) rather than delay it.

B.3.9 Confirm-all

This is a MLN protocol message with <performative> “confirm-all”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent to confirm the “accept-all” for its activity from its ‘preceding’ agent that has the preceding activity. This means that its cost reply is accepted and the contract is binding.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(confirm-all
:sender      SubA
:receiver    SubB
:content     (E B))
```

This example shows that the agent (SubA) confirms the “accept-all” for its activity (E) from its ‘preceding’ agent (SubB) that has the preceding activity (B). This means that its cost reply is accepted and the contract is binding.

B.3.10 Renege-all

This is a MLN protocol message with <performative> “renege-all”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent to renege the “accept-all” or “reject-all” for its activity from its ‘preceding’ agent that has the preceding activity. This means that its cost reply is rejected and the agent has to keep the original schedule of its activity.

- Field description

sendingSub_activityName: Activity name of sending agent receivingSub_activityName: Activity name of receiving agent

- Example

```
(renege-all
:sender      SubA
:receiver    SubB
:content     (E B))
```

This example shows that the agent (SubA) reneges the “accept-all” or “reject-all” for its activity (E) from its ‘preceding’ agent (SubB) that has the preceding activity (B). This means that its cost reply is rejected and the agent (SubA) has to keep the original schedule of its activity (E).

B.4 NEGOTIATION CONTROL PROTOCOLS

These performatives are used for managing the states of negotiation processes.

B.4.1 Ready

This is a MLN protocol message with <performative> “ready”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent to inform its ‘preceding’ agent, which has the preceding activity, that its activity is ready for negotiation.

- Field description

sendingSub_activityName: Activity name of sending agent

receivingSub_activityName: Activity name of receiving agent

- Example

```
(ready
:sender      SubA
:receiver    SubB
:content     (E B))
```

This example shows that the agent (SubA) informs its ‘preceding’ agent (SubB), which has the preceding activity (B), that its activity (E) is ready for negotiation.

B.4.2 Hand-over

This is a MLN protocol message with <performative> “hand-over”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

```
:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”
             “<proposedStartDate>”)”
```

- Semantics

This protocol allows an agent, which has the active activity, to inform its ‘succeeding’ agent, which has the succeeding activity, of starting a negotiation.

- Field description

sendingSub_activityName: Activity name of sending agent

receivingSub_activityName: Activity name of receiving agent

proposedStartDate: Proposed start date for the activity of receiving agent

- Example

```
(hand-over
:sender      SubB
:receiver    SubA
:content     (B E 9))
```

This example shows that the agent (SubB), which has the active activity (B), to inform the ‘succeeding’ agent (SubA), which has the succeeding activity (E), of starting a negotiation at Day 9.

B.4.3 Done

This is a MLN protocol message with <performative> “done”. The <receiver> is an agent and the <sender> is an agent.

- Syntax

:content := “(“<sendingSub_activityName>” “<receivingSub_activityName>”)”

- Semantics

This protocol allows an agent to inform its ‘preceding’ agent, which has the preceding activity, that its activity is finished the negotiation.

- Field description

sendingSub_activityName: Activity name of sending agent
receivingSub_activityName: Activity name of receiving agent

- Example

```
(done
:sender     SubA
:receiver   SubB
:content    (E B))
```

This example shows that the agent (SubA) informs its ‘preceding’ agent (SubB), which has the preceding activity (B), that its activity (E) is finished the negotiation

APPENDIX-C. MESSAGE-HANDLING MECHANISMS

The subcontractor agent reacts according to what message it receives. Therefore, the subcontractor agent should have the functionality of handling messages for each type of multi-linked message protocols. When the agent receives a message, it should also make a decision accordingly. Note that agents do not have to react to “final” or “sorry” messages because those messages are sent only to human subcontractors.

C.1 Handling “input” Messages

As shown in Figure C-1, when an agent receives an “input” message, it should check whether it is allowed to handle the message. The agent is not allowed to handle the message when the agent is currently involved in the negotiation, which means that the agent is ‘lock’. In that case, it sends a “sorry” message back to its human subcontractor. When the agent is not ‘lock’, it parses the content of the message and handles it according to the content type: AGENT or ACTIVITY. The agent stores the parsed information in the table for negotiation using activity ID as a key. The agent updates information when the same activity ID exists already. When the agent fails to parse the content of message due to typos, it sends a “sorry” message back to its human subcontractor.

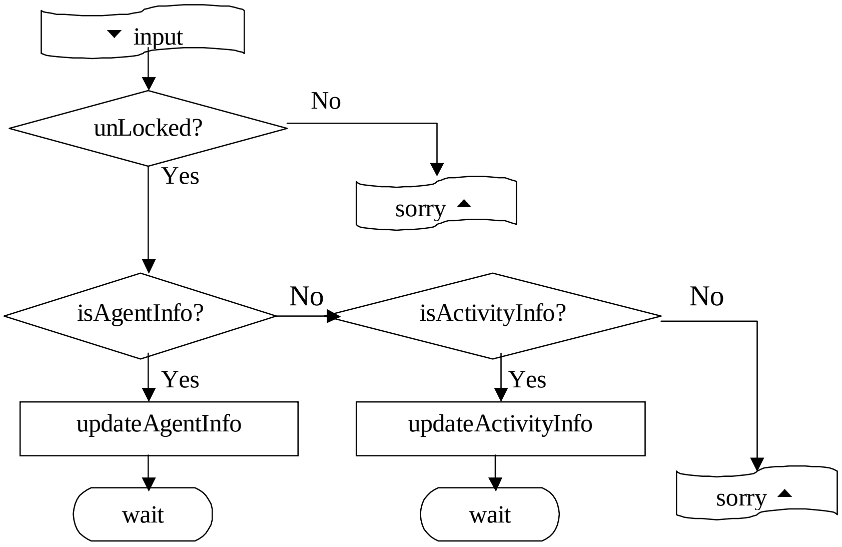


Figure C-1. Handling “input” messages

C.2 Handling "ready" Messages

As shown in Figure C-2, when an agent receives a "ready" message, whether it comes from its human subcontractor or an agent, it checks whether all the possible "ready" messages for the activity are received, which is necessary in order to ensure the synchronous negotiation. When the agent has received all the possible "ready" messages, it updates its status as 'lock' that prevents the agent from receiving any "input" message. Then the agent handles the message according to the position of its activity.

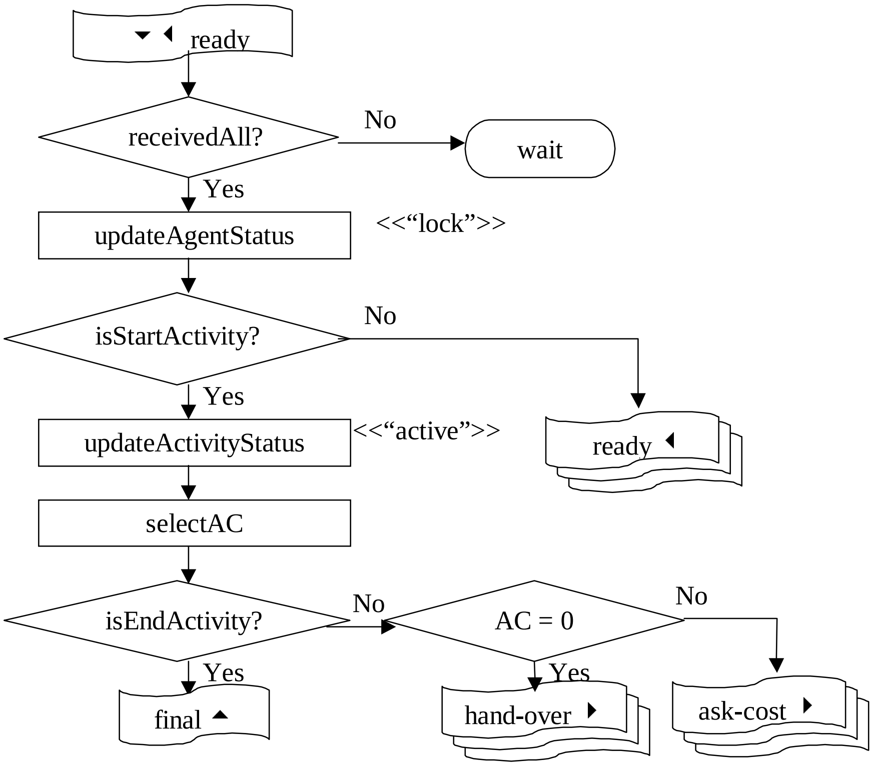


Figure C-2. Handling "ready" messages

If the activity is the 'start' activity, the agent becomes the 'active' agent for the activity. Unless the activity is also the 'end' activity, in which the agent sends a "final" message to its human subcontractor to inform it of the result, the agent finds the best option to pursue and sends "ask-cost" messages to the agents that have succeeding activities if AC is positive. This means the agent has an opportunity to save money, or sends "hand-over" messages to the agents that have succeeding activities, which means the agent decides to keep the original schedule. If the activity is the 'middle' activity, the agent sends "ready" messages to the agents that have preceding activities.

C.3 Handling "ask-cost" Messages

As shown in Figure C-3, when an agent receives an "ask-cost" message for its activity from an agent that has a preceding activity, it checks whether all the possible "ask-cost" messages for the activity are received. A "hand-over" message will be handled as a possible "ask-cost" message because there will be no "ask-cost" from the activity. When the agent has received all the possible "ask-cost" messages, the agent handles the message according to the position of its activity. One important feature of handling "ask-cost" messages is finding a critical activity or critical activities, which will dictate the start date of the receiving activity.

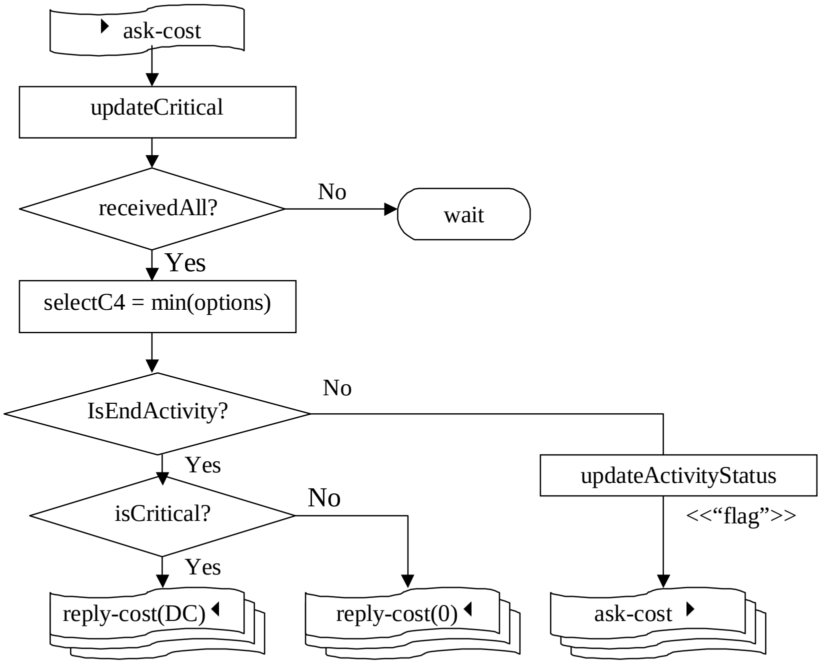


Figure C-3. Handling "ask-cost" messages

If the activity is the 'end' activity, the agent selects the lowest cost option (minimum of (C3, C4)) and sends "reply-cost" messages with the costs incurred by the "ask-cost" message. The aforementioned criticality is used here to reply with the costs to multiple activities: zero-cost for non-critical activities and full-cost for a critical activity. If there are multiple critical activities, the costs will be shared equally among them. For other 'middle' activities, the agent updates its activity's status as 'flag' for handling "reply-cost" messages, which means the 'flag' activity will determine the best option for replying cost (minimum of (C3, C4)) when it receives all the "reply-cost" messages. It also selects the lowest cost option (C4) and sends "ask-cost" messages to the agents that have succeeding activities according to its C4 option, which means that the agents want to know the consequence of delaying its activity schedule.

C.4 Handling "reply-cost" Messages

As shown in Figure C-4, when an agent receives a "reply-cost" message for its activity from an agent that has a succeeding activity, it checks whether all the "reply-cost" messages for the activity are received. When the agent has received all the "reply-cost" messages, it handles the message according to the position of its activity.

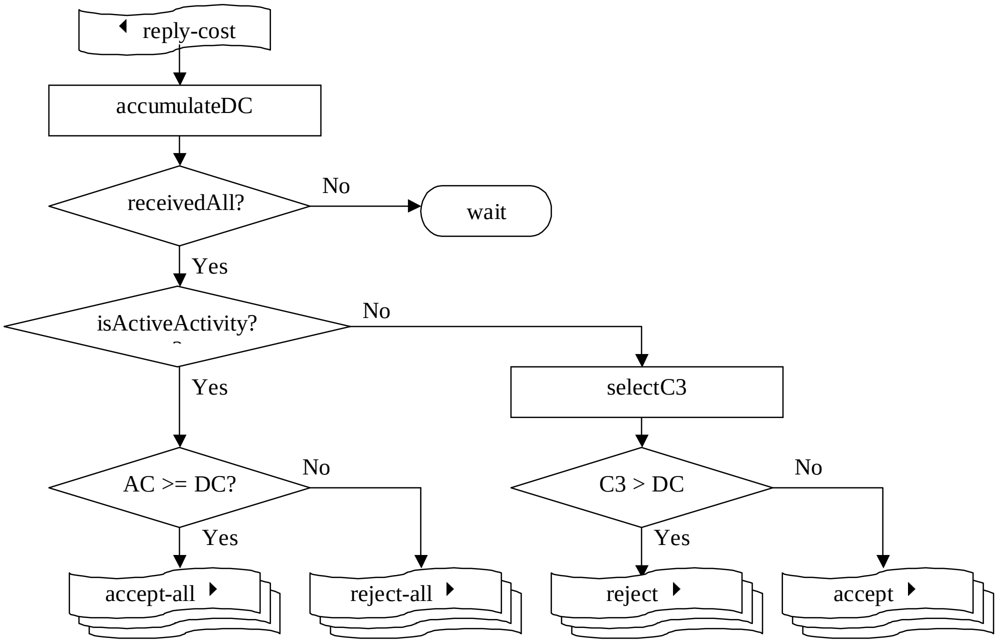


Figure C-4. Handling "reply-cost" messages

If the activity is the ‘flag’ activity, it selects the C3 and determines the best option for replying with the cost (minimum of (C3, C4)) and sends “accept” messages to the agents that have succeeding activities if C3 is bigger than the accumulated C4, which means delaying costs less than accelerating costs, or sends “reject” messages, which means the agent decides to accelerate its activity schedule. If the activity is the ‘active’ activity, the agent compares AC and DC and sends “accept-all” messages, which means the agent decides to change the original schedule and transfers (the DC portion of) utility to the succeeding activities, or “reject-all” messages, which means the agent decides to keep the original schedule, to the agents that have succeeding activities according to the results of comparison.

C.5 Handling “accept” Messages

As shown in Figure C-5, when an agent receives an “accept” message for its activity from an agent that has a preceding activity, it checks whether all the possible “accept” or “reject” messages for the activity are received. When the agent has received all the possible “accept” or “reject” messages, it handles the message according to the position of its activity.

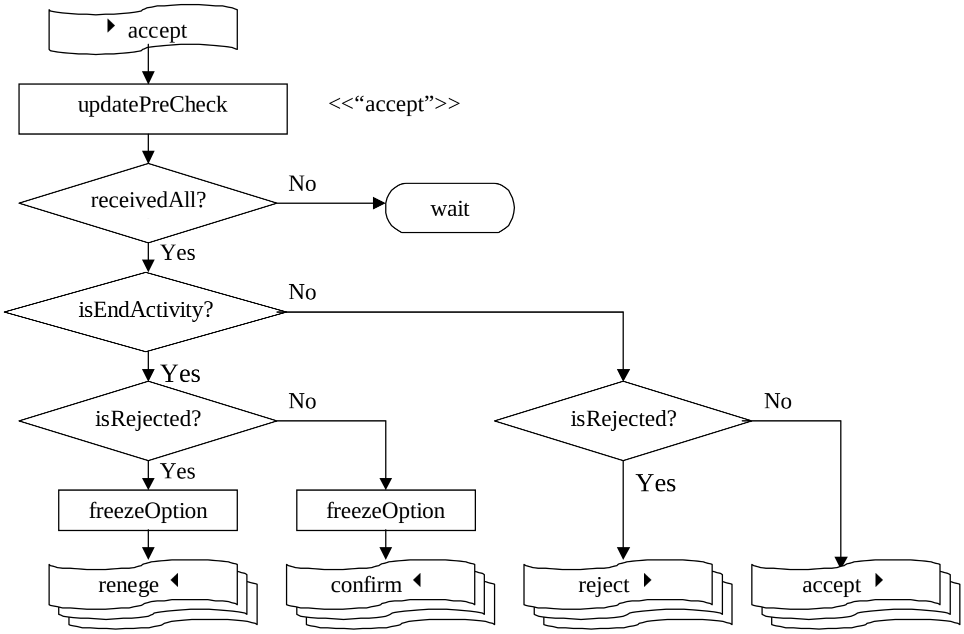


Figure C-5. Handling “accept” messages

If the activity is the ‘end’ activity, the agent checks whether a “reject” message is received and sends one of two messages to the agents that have sent “accept” or “reject” messages: a “confirm” message, which means the agent freezes the selected option for replying costs; or a “renege” message, which means the agent has to go back to the original option. For other ‘middle’ activities, the agent checks whether a “reject” message is received and sends “accept” or “reject” messages to the agents that have succeeding activities. Note that any “reject” message will cause the agent to nullify the “accept” messages, which results in consensus decisions.

C.6 Handling “reject” Messages

As shown in Figure C-6, when an agent receives a “reject” message for its activity from an agent that has a preceding activity, it checks whether all the possible “accept” or “reject” messages for the activity are received. When the agent has received all the possible “accept” or “reject” messages, it handles the message according to the position of its activity.

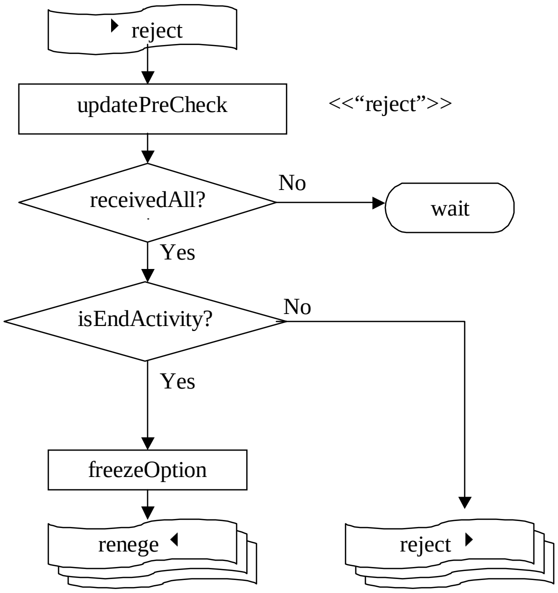


Figure C-6. Handling “reject” message

If the activity is the ‘end’ activity, without checking whether any “reject” message is received, the agent sends “renege” messages to the agents that have sent “accept” or "reject" messages, which means the agent has to go back to the original option. For other 'middle' activities, without checking whether any "reject" message is received, the agent sends "reject" messages to the agents that have succeeding activities. Although any "reject" message will cause the agent to nullify the "accept" messages, "reject" messages should be propagated to the end activity in order to avoid deadlocks, instead of sending "renege" messages immediately.

C.7 Handling "confirm" Messages

As shown in Figure C-7, when an agent receives a "confirm" message for its activity from an agent that has a succeeding activity, it checks whether all the "confirm" messages for the activity are received. When the agent has received all the "confirm" messages, it handles the message according to the position of its activity. Although any "confirm" message implies that all the "confirm" messages will be received, the agent should wait until all the "confirm" messages are received in order to avoid deadlocks.

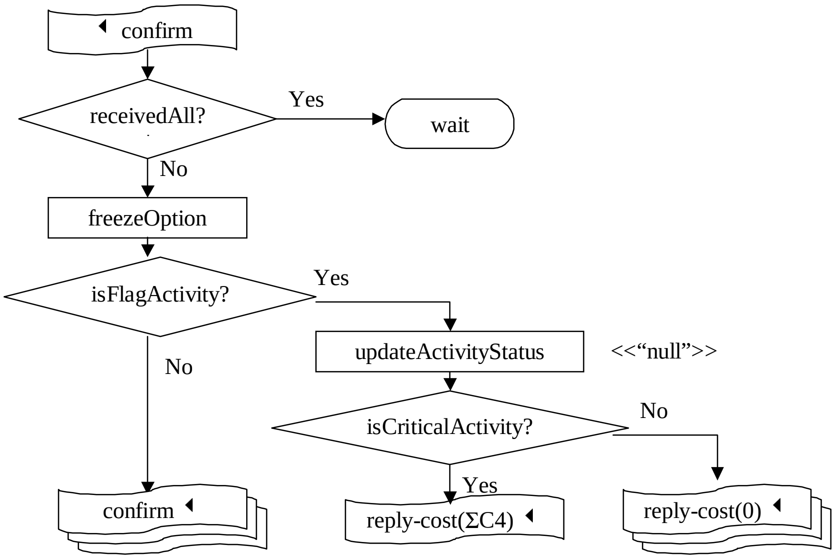


Figure C-7. Handling "confirm" messages

If the activity is the 'flag' activity, it selects the C4 and sends "reply-cost" messages to the agents that have succeeding activities with the same principle of criticality. Then the agent updates its activity's status to 'null', which means the agent is no longer the 'flag' activity. For other ‘middle’ activities, the agent sends “confirm” messages to the agents that have preceding activities.

C.8 Handling “renege” Messages

As shown in Figure C-8, when an agent receives a “renege” message for its activity from an agent that has a succeeding activity, it checks whether all the “renege” messages for the activity are received. When the agent has received all the “renege” messages, it handles the message according to the position of its activity. Although any “renege” message implies that all the “renege” messages will be received, the agent should wait until all the “renege” messages are received in order to avoid deadlocks.

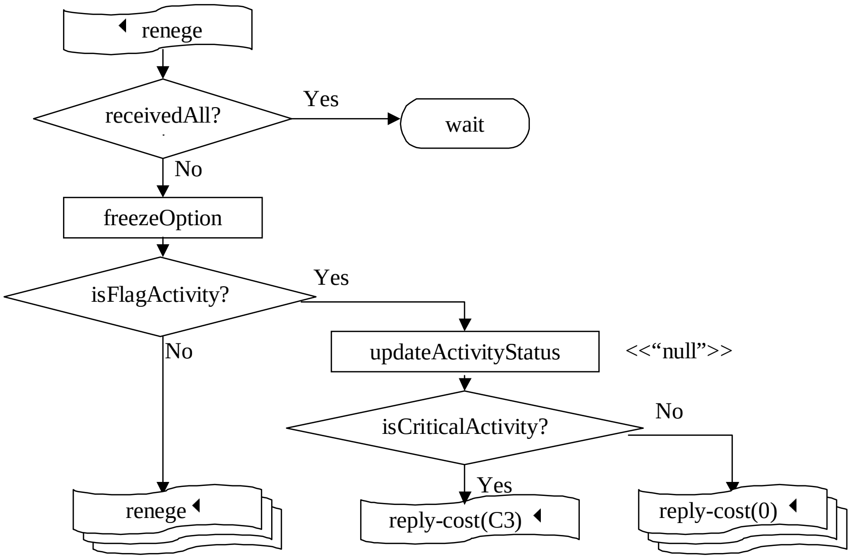


Figure C-8. Handling “renege” messages

If the activity is the ‘flag’ activity, it selects the C3 and sends “reply-cost” messages as like handling “confirm” messages. The other procedures are same as handling “confirm” messages except sending “renege” messages.

C.9 Handling “accept-all” Message

As shown in Figure C-9, when an agent receives an “accept-all” message for its activity from an agent that has a preceding activity, it checks whether all the possible “accept-all” or “reject-all” messages for the activity are received. When the agent has received all the possible “accept-all” or “reject-all” messages, it handles the messages according to the position of its activity.

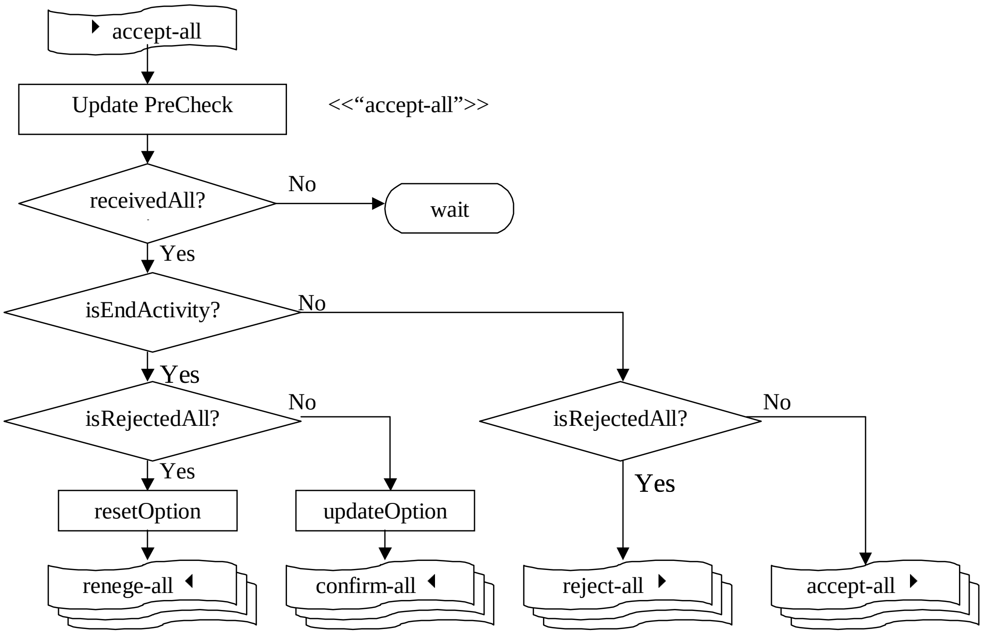


Figure C-9. Handling “accept-all” messages

If the activity is the ‘end’ activity, the agent checks whether a “reject-all” message is received and sends one of two messages to the agents that have sent “accept-all” or “reject-all” messages: a “confirm-all,” which means the agent updates the selected option for replying costs; or a “renege-all” message, which means the agent has to go back to the original option. For other ‘middle’ activities, the agent checks whether a “reject” message is received and sends “accept-all” or “reject-all” messages to the agents that have succeeding activities. Note that any “reject-all” message will cause the agent to nullify the “accept-all” messages, which results in consensus decisions.

C.10 Handling “reject-all” Messages

As shown in Figure C-10, when an agent receives a “reject-all” message for its activity from an agent that has a preceding activity, it checks whether all the possible “accept-all” or “reject-all” messages for the activity are received. When the agent has received all the possible “accept-all” or “reject-all” messages, it handles the message according to the position of its activity.

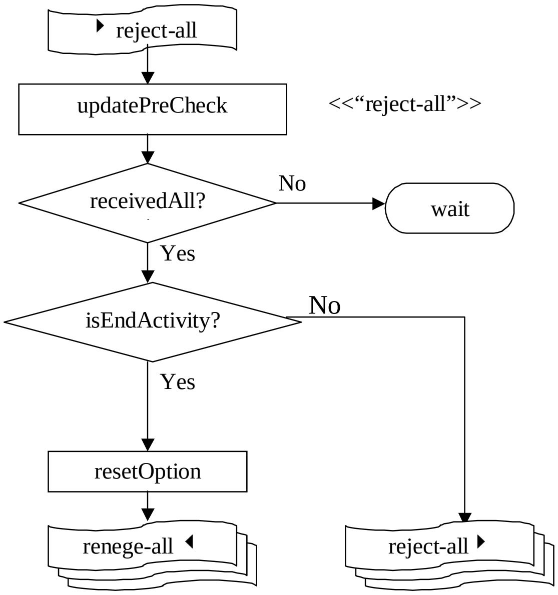


Figure C-10. Handling “reject-all” Message

If the activity is the ‘end’ activity, without checking whether any “reject-all” message is received, the agent send “renege-all” messages to the agents that have succeeding activities, which means the agent has to go back to the original option. Then the agent updates its schedule change options accordingly. For other ‘middle’ activities, without checking whether any “reject-all” message is received, the agent sends “reject-all” messages. Although any “reject-all” message will cause the agent to nullify the “accept-all” messages, “reject-all” messages should be propagated to the end activity in order to avoid deadlocks, instead of sending “renege-all” messages immediately.

C.11 Handling “confirm-all” Message

As shown in Figure C-11, when an agent receives a “confirm-all” message for its activity from an agent that has a succeeding activity, it checks whether all the “confirm-all” messages for the activity have been received. When the agent has received all the “confirm-all” messages, it handles the message according to the position of its activity. Although any “confirm-all” message implies that all the “confirm-all” messages will be received, the agent should wait until all the “confirm-all” messages are received in order to avoid deadlocks.

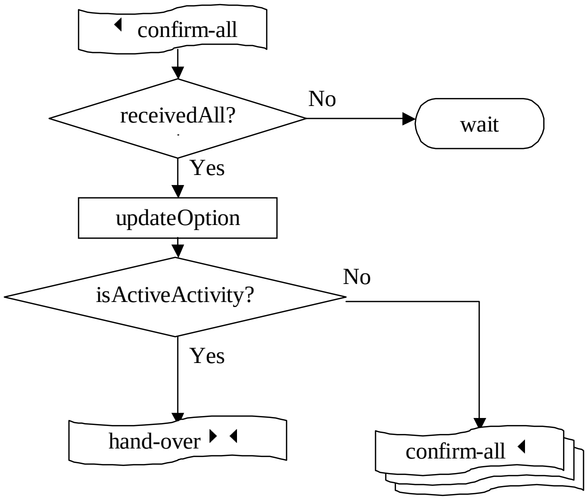


Figure C-11. Handling “confirm-all” Message

If the activity is the ‘active’ activity, it sends a “hand-over” message to itself to initiate another cycle of negotiation because any “confirm-all” message implies that its schedule change options are changed. Then the agent updates its status to ‘null’, which means the agent is no longer the ‘active’ activity. For other ‘middle’ activities, the agent sends “confirm-all” messages to the agents that have preceding activities.

C.12 Handling “renege-all” Messages

As shown in Figure C-12, when an agent receives a “renege-all” message for its activity from an agent that has a succeeding activity, it checks whether all the “renege-all” messages for the activity are received. When the agent has received all the “renege-all” messages, it handles the messages according to the position of its activity. Although any “renege-all” message implies that all the “renege-all” messages will be received, the agent should wait until all the “renege-all” messages are received in order to avoid deadlocks.

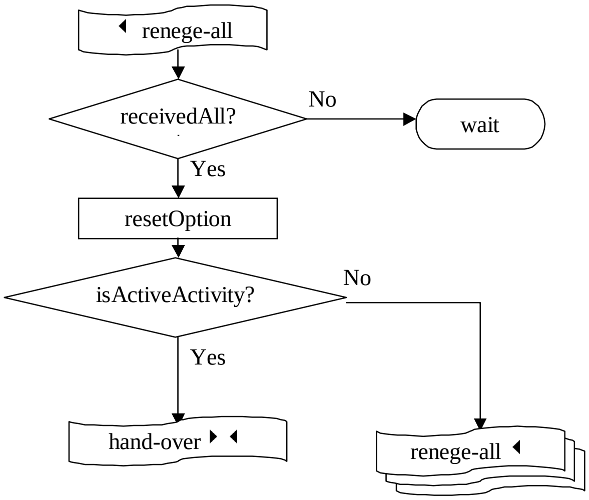


Figure C-12. Handling “renege-all” messages

If the activity is the ‘active’ activity, it sends a “hand-over” message to the agents that have the succeeding activities because any “renege-all” message implies that its schedule change options are not changed. Then the agent updates its status to ‘null’, which means the agent is no longer the ‘active’ activity. For other ‘middle’ activities, the agent sends “renege-all” messages to the agents that have preceding activities.

C.13 Handling “hand-over” Message

As shown in Figure C-13, when an agent receives a “hand-over” message for its activity from an agent that has a preceding activity, it checks whether all the “hand-over” messages for the activity are received. When the agent has received all the “hand-over” messages, which is necessary in order to ensure the synchronous negotiation, it updates the status as “active” and handles the messages according to the position of its activity.

If the activity is the ‘end’ activity, the agent sends a “final” message to its human subcontractor to inform the result and send “done” messages to the agents that have preceding agents. It also updates its status as “unlock” to allow its human subcontractor to provide inputs. For the ‘middle’ activities, the agent finds the best option to pursue and sends “ask-cost” messages to the agents that have succeeding activities if AC is positive, which means the agent has an opportunity to save money, or sends “hand-over” messages to the agents that have the succeeding activities, which means the agent decides to keep the original schedule.

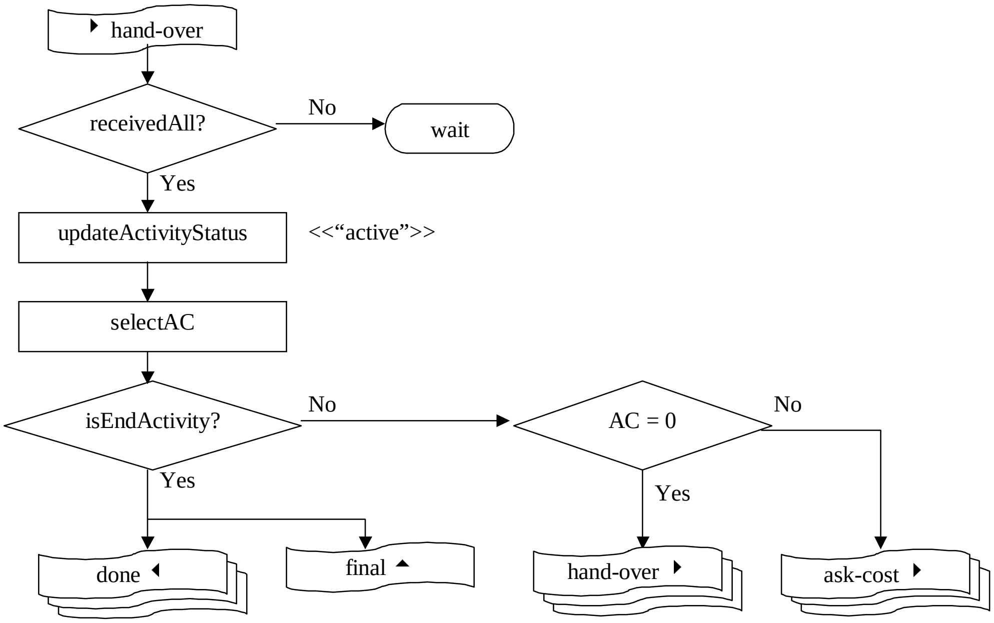


Figure C-13. Handling "hand-over" messages

C.14 Handling "done" Message

As shown in Figure C-14, when an agent receives a "done" message for its activity from an agent that has a succeeding activity, it checks whether all the "done" messages for the activity are received. When the agent has received all the "done" messages, it updates its status as "unlock" and handles the messages according to the position of its activity. Although any "done" message implies that all the "done" messages will be received, the agent should wait until all the "done" messages are received in order to avoid deadlocks.

If the activity is the 'start' activity, it sends a "final" message to its human subcontractor. For other 'middle' activities, the agent sends "done" messages to the agents which have preceding activities and sends a "final" message to its human subcontractor.

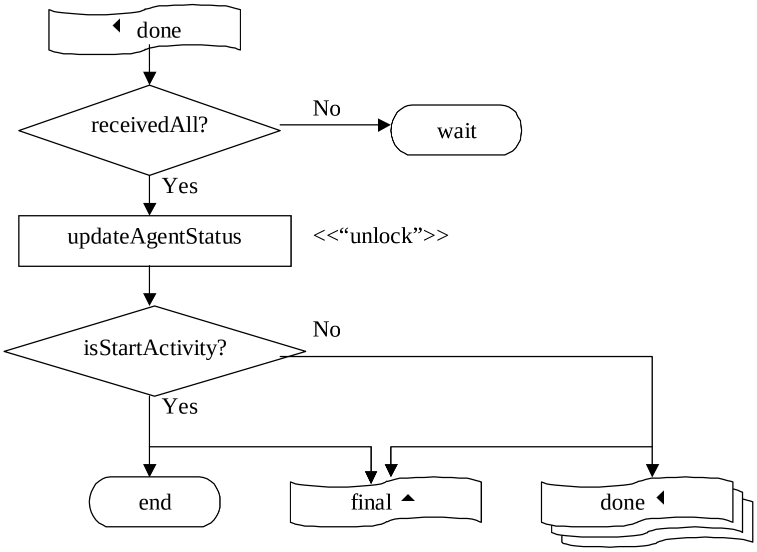
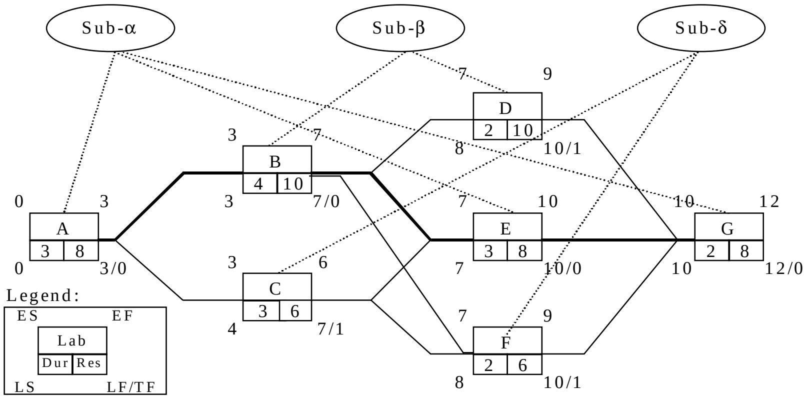


Figure C-14. Handling “done” messages

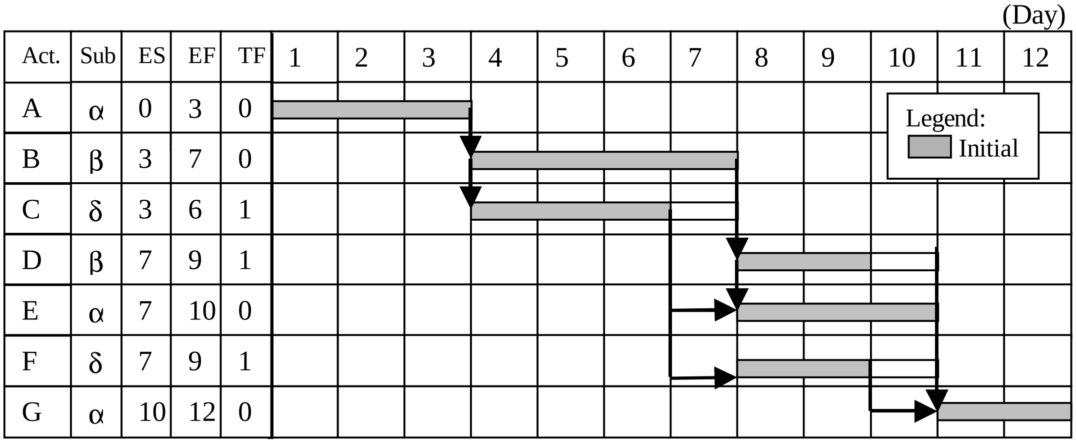
APPENDIX-D. STEP-BY-STEP ABCN ON DSAS

A step-by-step Agent-based Compensatory Negotiation (ABCN) methodology on Distributed Subcontractor Agent System (DSAS) example will verify that the DSAS demonstrates the effectiveness of the distributed coordination of project schedule changes.

Step-0: Consider the example network in Figure D-1(a). The results of conventional CPM calculations appear on the diagram. The resource requirement for each activity appears on the diagram in Figure D-1(a). Activities (A, E and G) are assigned to Sub-α. Activities (B and D) are assigned to Sub-β. Activities (C and F) are assigned to Sub-δ.



(a)



Act.	Sub	ES	EF	TF
A	α	0	3	0
B	β	3	7	0
C	δ	3	6	1
D	β	7	9	1
E	α	7	10	0
F	δ	7	9	1
G	α	10	12	0

(b)

Figure D-1. Example network and ERS schedule

The resource histogram in Figure D-2 indicates the initial resource requirements.

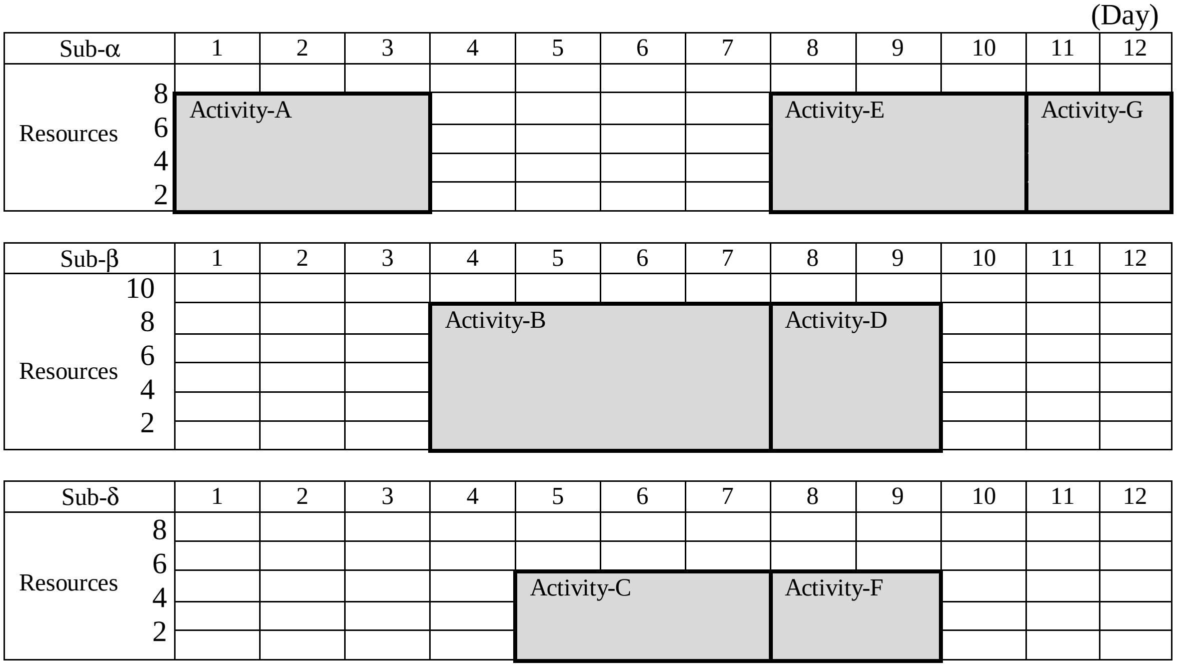


Figure D-2. Resource requirement histogram

The resource histogram in Figure D-3 indicates the available resources.

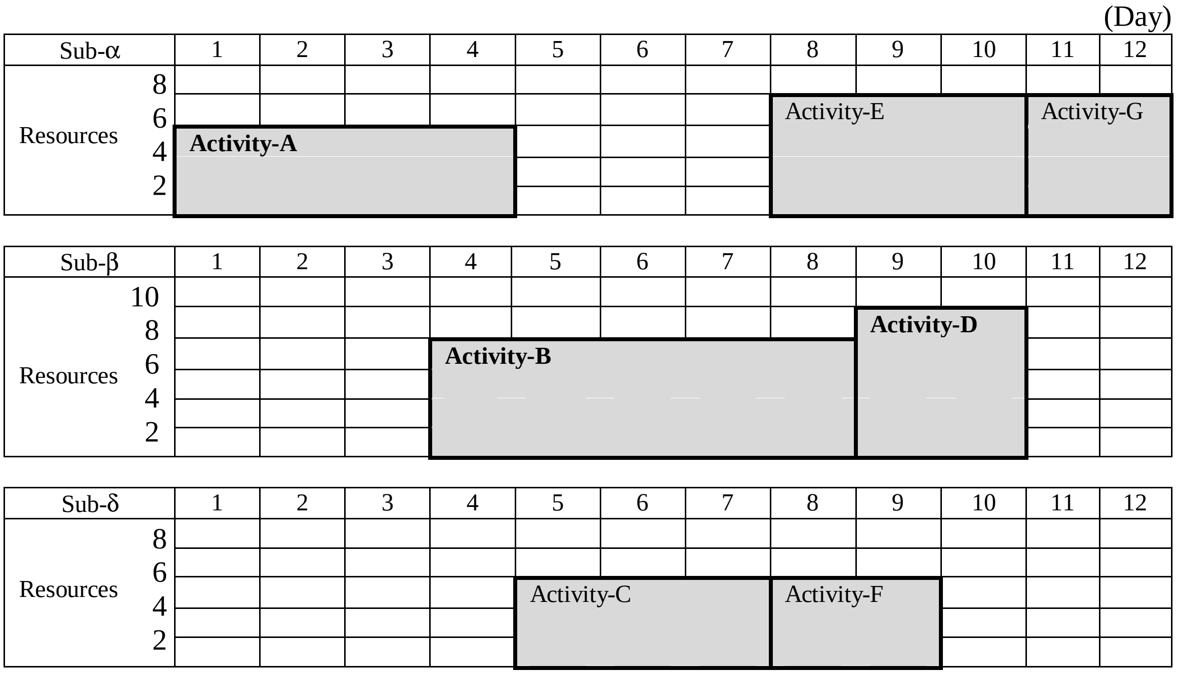


Figure D-3. Available resource histogram

This resource histogram implies that some subcontractors' schedules differ from their original ones. Based on these available resource histograms, some of the subcontractors' preferred schedules shift, as shown by the diagonal pattern in Figure D-4.

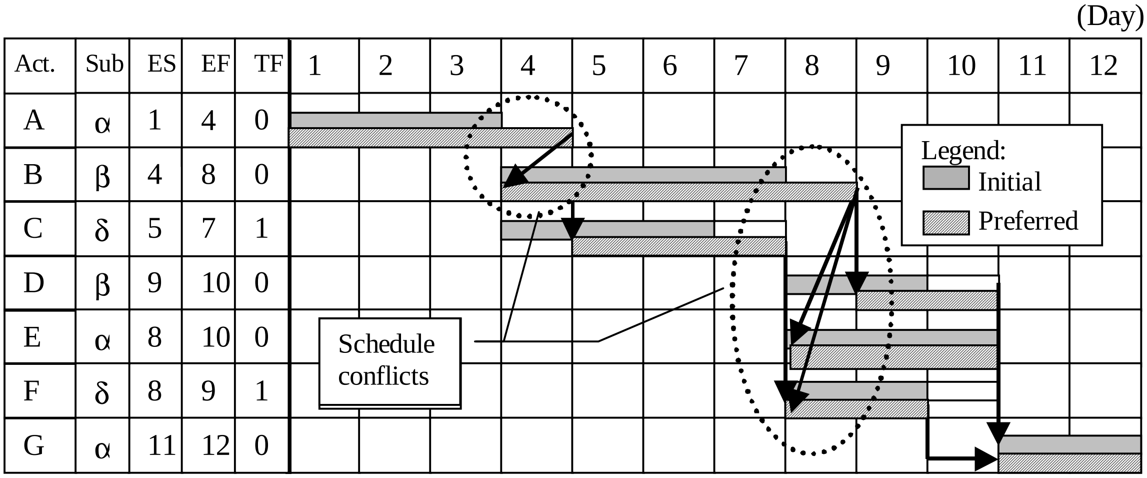


Figure D-4. Subcontractors' preferred schedule

Human subcontractors provide their agents with agent and activity information by sending "input" messages. Table D-1 shows the initial schedule-change options in the format of (startDate(day) endDate(day) extraCost($)). Note that options marked '*' are initially available options, which are feasible because the start date of an activity is later than the end dates of the preceding activities. The option marked '**'includes liquidated damages ($4,000) for a 2-day project delay.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480)	(1 4 0)			
B	(4 7 1920)	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0)				
D	(8 10 0)	(9 10 0)	(10 11 960)		
E	(8 10 0)	(9 10 640)	(10 12 0)		
F	(8 9 0)	(9 10 384)	(10 11 768)		
G	(11 12 0)	(13 14 4512)**			

Table D-1. Initial schedule change options at Step-0

Step-1: The Distributed Coordination of Project Schedule Changes (DCPSC) starts with "ready" messages from its human subcontractors through distributed Graphic User Interfaces (GUIs). Agents propagate the "ready" messages backward to the agents that have preceding activities, based on the project plan.

Step-2: The agents propagate "ready" messages backward to the agents that have the preceding activities.

Step-3: The agents propagate "ready" messages backward to the agent that has the start activity.

Step-4: After receiving all "ready" messages, the agent (Sub-α) that has the start activity (A) selects an option (A2) to be explored and finds AC ($480), the difference between the selected option ($0) and the initial option ($480). To find the extra cost for the succeeding activities from the change, Sub-α sends "ask-cost" messages to agents (Sub-β and Sub-δ) that have succeeding activities (B and C), which ask the succeeding activities to start at Day 5.

Step-5: After receiving all "ask-cost" messages, the agents (Sub-β and Sub-δ) select the lowest-cost possible options (B5 and C1) and find the DC, which are the extra costs of selected options. They then send "ask-cost" messages to the agents (Sub-α, Sub-β, and Sub-δ) that have succeeding activities (D, E, and F). Note that Sub-β and Sub-δ send messages to themselves, but for different activities.

Step-6: The "ask-cost" messages are propagated forward to the end activity, with selecting options (E3, D3, F3).

Step-7: After receiving all "ask-cost" messages, the agent (Sub-α) that has the end activity (G) selects the lowest possible option (G2) as DC. The agent also finds which activity is a critical activity (E), which asks (G) to start on the latest date. And then Sub-α sends "reply-cost" messages with DC to the agent (Sub-α) that has critical activity (E) and sends zero (0) cost to the other agents (Sub-β and Sub-δ). Table D-2 shows the selected schedule-change options, which is marked "✓."

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480)	(1 4 0) ✓			
B	(4 7 1920)	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0) ✓
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0)	(10 11 960) ✓		
E	(8 10 0)	(9 10 640)	(10 12 0) ✓		
F	(8 9 0)	(9 10 384)	(10 11 768) ✓		
G	(11 12 0)	(13 14 4512)** ✓			

Table D-2. Selected schedule change options at Step-7

Step-8: After receiving all "reply-cost" messages, the agents (Sub-α, Sub-β, and Sub-δ) accumulate DC and compare them with AC, which can be finished within the initial end date. In this case, there are no other possible options; therefore, the agents send "accept" messages to the agent (Sub-α) that has a succeeding activity (G).

Step-9: After receiving all "accept" messages, the agent (Sub-α) sends "confirm" messages to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F).

Step-10: After receiving all "confirm" messages, the agents (Sub-α, Sub-β, and Sub-δ) send "reply-cost" messages with accumulated DC to the agents (Sub-β and Sub-δ) that have preceding activities (D, E, and F).

Step-11: Similar to Step-8. But Sub-β finds the lower-cost AC ($5,760) for Activity-B than the accumulated DC ($6,240). Therefore, Sub-β sends "reject" messages to the agents (Sub-α and Sub-β) that have succeeding activities (D and E). On the other hand, Sub-δ cannot find another option for Activity-C and sends "accept" messages to the other agents (Sub-α and Sub-δ).

Step-12: After receiving all "accept/reject" messages, the agents (Sub-α, Sub-β, and Sub-δ) send "reject" messages to the agent (Sub-α) that has succeeding activities (G). Note that "accept" messages are sent only if all messages are "accept", which makes them consensus agreements.

Step-13: After receiving all “reject” messages, the agent (Sub-α) sends “renege” messages to the agents (Sub-α, Sub-β, and Sub-δ) that have sent “accept/reject” messages for activities (D, E, and F).

Step-14: The “renege” messages propagate backward to the agents (Sub-β and Sub-δ) that initially sent “accept” or “reject” messages.

Step-15: After receiving all “renege” messages, the agents (Sub-β and Sub-δ) send “reply-cost” messages with their AC to the agent (Sub-α) that has sent “ask-cost” messages for activity (A). Table D-3 shows the selected schedule-change options.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480)	(1 4 0) ✓			
B	(4 7 1920)	(4 8 0)	(5 7 5760) ✓	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0) ✓	(9 10 0)	(10 11 960)		
E	(8 10 0) ✓	(9 10 640)	(10 12 0)		
F	(8 9 0) ✓	(9 10 384)	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table D-3. Selected schedule change options at Step-15

Step-16: After receiving all “reply-cost” messages, the agent (Sub-α) accumulates DC and compares them with its AC, which can be finished within the initial end dates. In this case, DC ($5,760) is more than AC ($480). Therefore it sends “reject-all” messages to the agents (Sub-β and Sub-δ) that have succeeding activities (B and C). Note that “reject-all” messages are authorative compared to tentative “reject” messages.

Step-17: The agents (Sub-β and Sub-δ) propagate “reject-all” messages forward to the agents (Sub-α, Sub-β, and Sub-δ), which have activities (E, E, and F).

Step-18: The agents (Sub-α, Sub-β, and Sub-δ) propagate “reject-all” messages forward to the agent (Sub-α), which has the end activity (G).

Step-19: After receiving all “reject-all” messages, the agent (Sub-α) sends “renege-all” messages to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F).

Step-20: The agents (Sub-α, Sub-β, and Sub-δ) propagate “renege-all” messages backward to the agents (Sub-β and Sub-δ), which have activities (B and C).

Step-21: The agents (Sub-β and Sub-δ) propagate “renege-all” messages backward to the agent (Sub-α) that initially sent a “reject-all” message. Table D-4 shows the selected schedule-change options. Note that all agents have to stick with the initial options.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920) ✓	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0) ✓	(9 10 0)	(10 11 960)		
E	(8 10 0) ✓	(9 10 640)	(10 12 0)		
F	(8 9 0) ✓	(9 10 384)	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table D-4. Selected schedule change options at Step-21

Step-22: After receiving all “renege-all” messages, the agent (Sub-α) fixes its option (A1) and sends “hand-over” messages to the agents (Sub-β and Sub-δ) that have succeeding activities (B and C).

Step-23: After receiving all “hand-over” messages, the agents (Sub-β and Sub-δ) select an option (B2 and C1) to be explored and find AC. Sub-β finds AC ($1,980) and sends “ask-cost” messages to agents (Sub-α, Sub-β, and Sub-δ) that have succeeding activities (D, E, and F), which ask the succeeding activities to start at Day 9. Meantime, Sub-δ sends “hand-over” messages to agents (Sub-α and Sub-δ) that have succeeding activities (E and F) because its AC is zero (0), which means there is no better option.

Step-24: After receiving all “ask-cost” messages, the agents (Sub-α, Sub-β, and Sub-δ) select the options that are the lowest possible options (E2, D2, and F2). And then sends “ask-cost” messages to the agent (Sub-α) that has succeeding activities (G). Note that Sub-α and Sub-δ handle “ask-cost” messages even though they have also received “hand-over” messages.

Step-25: After receiving all “ask-cost” messages, the agent (Sub-α), which has the end activity (G), selects the lowest possible option (G1). And then Sub-α sends “reply-cost” messages to the agents (Sub-α, Sub-β, and Sub-δ), which have sent “ask-cost” messages for activities (D, E, and F).

Step-26: After receiving all “reply-cost” messages, the agents (Sub-α, Sub-β, and Sub-δ) accumulate DC and compare them with AC, which can be finished within the initial end date. In this case, there are no other possible options; therefore, the agents send “accept” messages to the agent (Sub-α) that has a succeeding activity (G).

Step-27: After receiving all “accept” messages, the agent (Sub-α) sends “confirm” messages to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F).

Step-28: After receiving all “confirm” messages, the agents (Sub-α, Sub-β, and Sub-δ) send “reply-cost” messages with accumulated DC to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F). Table D-5 shows the selected schedule-change options.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920)	(4 8 0) ✓	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0) ✓	(10 11 960)		
E	(8 10 0)	(9 10 640) ✓	(10 12 0)		
F	(8 9 0)	(9 10 384) ✓	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table D-5. Selected schedule change options at Step-28

Step-29: After receiving all "reply-cost" messages, the agent (Sub-β) accumulates DC and compares them with AC, which needs to finish within the initial end date. In this case, DC ($1,024) is less than AC ($1,960). Therefore, it sends "accept-all" messages to the agents (Sub-α, Sub-β, and Sub-δ) that have succeeding activities (D, E, and F).

Step-30: The agents (Sub-α, Sub-β, and Sub-δ) propagate "accept-all" messages forward to the agent (Sub-α), which has the end activity (G).

Step-31: After receiving all "accept-all" messages, the agent (Sub-α) sends "confirm-all" messages to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F).

Step-32: The agents (Sub-α, Sub-β, and Sub-δ) change their option from (E2 (9 10 640), F2 (9 10 384)) to (E2' (9 10 0), F2' (9 10 0)) and propagate "confirm-all" messages backward to the agent (Sub-β), which has activity (B). Note that sending "confirm-all" messages means contract binding between them.

Step-33: After receiving all "confirm-all" messages, the agent (Sub-β) changes its option from (B2 (4 8 0)) to (B2' (4 8 1024)) and sends "hand-over" messages to its agent (Sub-β) that has the same activity (B) because schedule-cost options change after the compensation. Table C-6 shows the selected schedule-change options. Note that options (B2, E2, and F2) change after compensation. Table D-6 shows the selected schedule-change options.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920)	(4 8 1024) ✓	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0) ✓	(10 11 960)		
E	(8 10 0)	(9 10 0) ✓	(10 12 0)		
F	(8 9 0)	(9 10 0) ✓	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table D-6. Selected schedule change options at Step-33

Step-34: After receiving the “hand-over” message, Sub-β selects an option (B5) to explore and finds AC. Sub-β finds AC ($1,024) and sends “ask-cost” messages to agents (Sub-α, Sub-β, and Sub-δ) that have succeeding activities (D, E, and F), which ask the succeeding activities to start at Day 10.

Step-35: After receiving the “ask-cost” message, the agents (Sub-α, Sub-β, and Sub-δ) select the options that are the lowest possible options (E3, D3, and F3). And then sends “ask-cost” messages to the agent (Sub-α) that has succeeding activities (G).

Step-36: After receiving all “ask-cost” messages, the agent (Sub-α), which has the end activity (G), selects the lowest possible option (G2). And then Sub-α sends “reply-cost” messages to the agents (Sub-α, Sub-β, and Sub-δ), which have sent “ask-cost” messages for activities (D, E, and F).

Step-37: After receiving all “reply-cost” messages, the agents (Sub-α, Sub-β, and Sub-δ) accumulate DC and compare them with AC, which can be finished within the initial end date. In this case, there are no other possible options; therefore, the agents send “accept” messages to the agent (Sub-α) that has a succeeding activity (G).

Step-38: After receiving all “accept” messages, Sub-α sends “confirm” messages to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F).

Step-39: After receiving all “confirm” messages, the agents (Sub-α, Sub-β, and Sub-δ) send “reply-cost” messages with accumulated DC to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F). Table D-7 shows the selected schedule-change options.

Step-40: After receiving all “reply-cost” messages, Sub-β accumulates DC and compares them with an AC that can be finished within the initial end dates. In this case, DC ($6,240) is more than AC ($1,024). Therefore it sends “reject-all” messages to the agents (Sub-α, Sub-β, and Sub-δ) that have succeeding activities (D, E, and F).

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920)	(4 8 0)	(5 7 5760)	(5 8 1920)	(5 9 0) ✓
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0)	(10 11 960) ✓		
E	(8 10 0)	(9 10 640)	(10 12 0) ✓		
F	(8 9 0)	(9 10 384)	(10 11 768) ✓		
G	(11 12 0)	(13 14 4512)** ✓			

Table D-7. Selected schedule change options at Step-39

Step-41: The agents (Sub-α, Sub-β, and Sub-δ) propagate "reject-all" messages forward to Sub-α, which has the end activity (G).

Step-42: After receiving all "reject-all" messages, Sub-α sends "renege-all" messages to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities (D, E, and F).

Step-43: The agents (Sub-α, Sub-β, and Sub-δ) propagate "renege-all" messages backward to Sub-β, which has an activity (B). Table D-8 shows the selected schedule-change options.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920)	(4 8 1024) ✓	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0) ✓	(10 11 960)		
E	(8 10 0)	(9 10 0) ✓	(10 12 0)		
F	(8 9 0)	(9 10 0) ✓	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table D-8. Selected schedule-change options at Step-43

Step-44: After receiving all "renege-all" messages, Sub-β sends "hand-over" messages to the agents (Sub-α, Sub-β, and Sub-δ) that have succeeding activities (D, E, and F).

Step-45: After receiving all "hand-over" messages, the agents (Sub-α, Sub-β, and Sub-δ) select an option (E2', D2 and F2') to be explored. However, since their ACs are zero (0), agents (Sub-α, Sub-β, and Sub-δ) send “hand-over” messages to Sub-α that has the succeeding activity (G).

Step-46: After receiving all “hand-over” messages, Sub-α, which has the end activity, sends a “final” message that informs its human subcontractor of the negotiation result, which includes new activity duration, activity benefit from the negotiation, receivable, payable, and new schedule-change options. Also Sub-α sends “done” message to the agents (Sub-α, Sub-β, and Sub-δ) that have preceding activities.

Step-47: After receiving all “done” messages, the agents (Sub-α, Sub-β, and Sub-δ) send “final” messages that informs their human subcontractors of the negotiation results, and propagate “done” messages backward to the agents (Sub-β and Sub-δ), which have activities (B and C).

Step-48: After receiving all “done” messages, the agents (Sub-β and Sub-δ) send “final” messages that informs their human subcontractors of the negotiation results, and propagate “done” messages backward to Sub-α, which has the start activity (A).

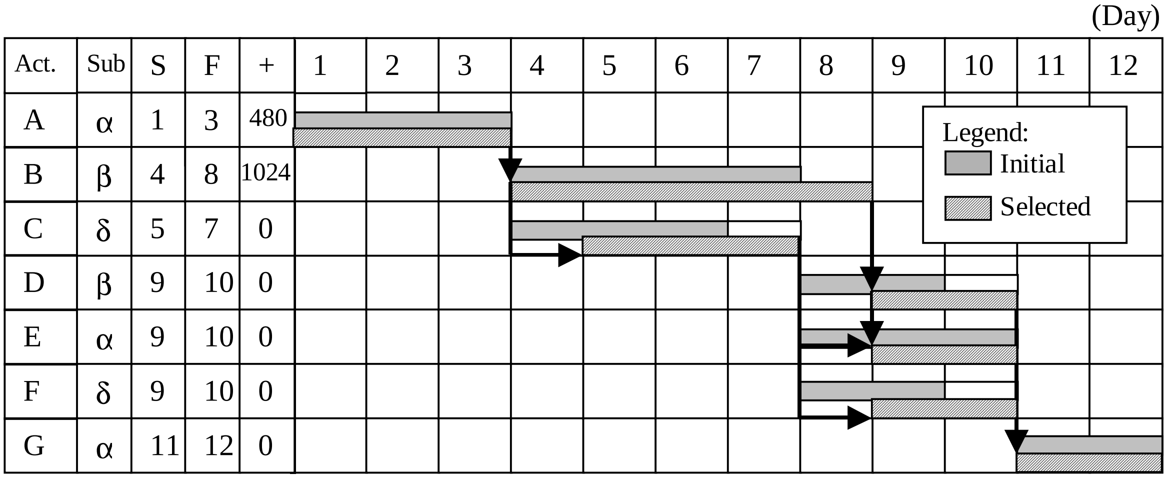
Step-49: After receiving all “done” messages, Sub-α, which has the start activity, sends a “final” message to its human subcontractor.

Step-50: A cycle of negotiation is finished. Table D-9 shows the selected schedule-change options.

Activity	Option 1*	Option 2	Option 3	Option 4	Option 5
A	(1 3 480) ✓	(1 4 0)			
B	(4 7 1920)	(4 8 1024) ✓	(5 7 5760)	(5 8 1920)	(5 9 0)
C	(5 7 0) ✓				
D	(8 10 0)	(9 10 0) ✓	(10 11 960)		
E	(8 10 0)	(9 10 0) ✓	(10 12 0)		
F	(8 9 0)	(9 10 0) ✓	(10 11 768)		
G	(11 12 0) ✓	(13 14 4512)**			

Table D-9. Selected schedule change options at Step-50

Diagonal patterns in Figure D-5 shows the final revised schedule at Step-50.



Act.	Sub	S	F	+
A	α	1	3	480
B	β	4	8	1024
C	δ	5	7	0
D	β	9	10	0
E	α	9	10	0
F	δ	9	10	0
G	α	11	12	0

Figure D-5. Final revised schedule

Figure D-6 shows the final revised resource histogram at Step-50. The diagonal pattern indicates the overtime or importing extra resources.

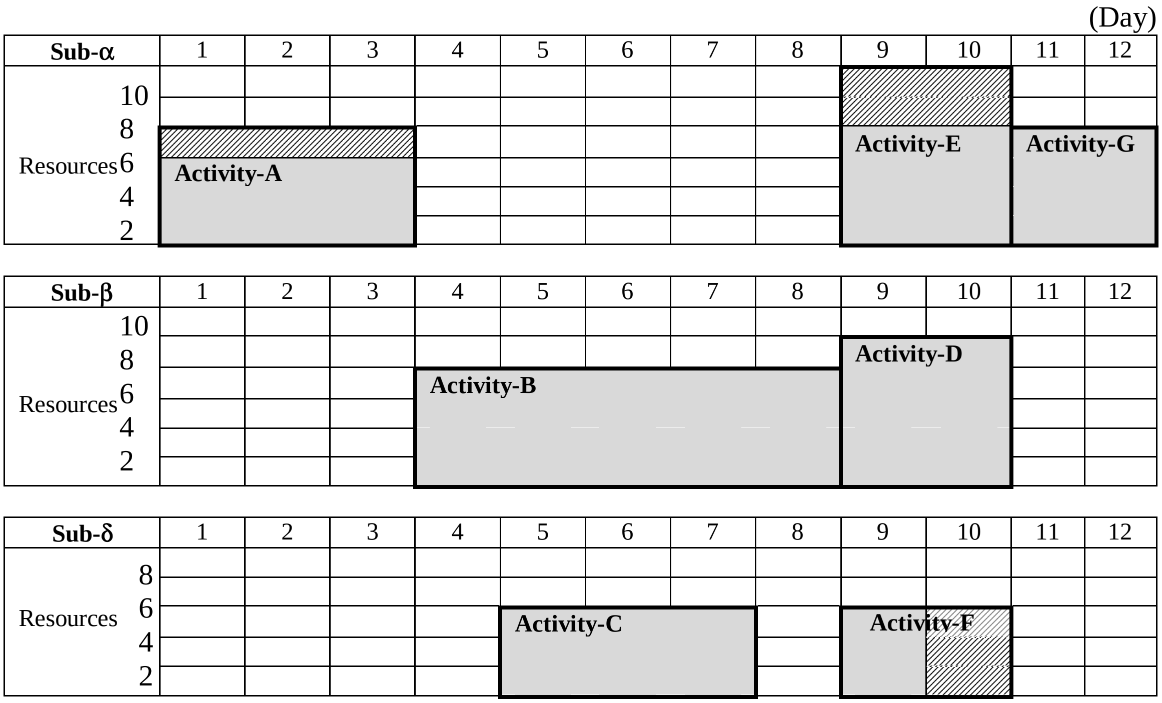


Figure D-6. Final revised resource histogram

APPENDIX-E. MATHEMATICAL PROOFS OF EVALUATION RESULTS

E.1 Definitions of Total Cost Overruns of TCC, LCC, and DCPSC

This section shows the definitions of total cost overruns of TCC (C_{TCC}), LCC (C_{LCC}), and DCPSC (C_{DCPSC}).

E.1.1 Definition of total cost overruns of TCC

A naïve formulation of the total cost overruns (C_{TCC}) after TCC is as follows:

$$C_{TCC} = \sum_{i=1}^{n} AC_i , \quad AC_i \geq 0$$

Where AC_i = the cost for expediting the i^{th} activity, over and above the original cost, denoted "$C_{i1} – C_{i0}$" on page 40.

However, examining the actual process of getting AC_i on any project plan produces a more complex formulation as below:

$$C_{TCC} = \sum_{k\geq 1 \cap Wave_k \neq \{\}} \left(\sum_{i \in Wave_k} AC_i^k \right), \quad AC_i^k \geq 0$$

Where k is the order of *Wave*, which is a set of activities to be considered at the same time. I.e., $Wave_1$ = the first (only one) activity and $Wave_i$ = {succeeding activities of $Wave_{(i-1)}$} for $i \geq 2$;

AC_i^k = the cost for expediting the i^{th} activity, over and above the original cost within the k^{th} Wave.

Since $\forall k, AC_i^k = AC_i$, because there is only one *AC* for each activity, the following formula generalize the total cost overruns (C_{TCC}) after TCC:

$$C_{TCC} = \sum_{k\geq 1 \cap Wave_k \neq \{\ \}} \left(\sum_{i \in Wave_k} AC_i^k \right) = \sum_{i=1}^{n} AC_i , \quad AC_i^k , AC_i \geq 0$$

E.1.2 Definition of total cost overruns of LCC

A naïve formulation of the total cost overruns (C_{LCC}) after LCC is as follows:

$$C_{LCC} = \sum_{i=1}^{n} LCC_i = \sum_{i=1}^{n} CB_i, \quad CB_i \geq 0$$

Where $LCC_i = CBi$ = the cost for delaying the i^{th} activity, over and above the original cost, denoted "$C_{i2} - C_{i0}$" on page 40 for i=1 and "$C_{i4} - C_{i0}$" on page 41 for i>2. This includes the cost of unscheduled resources and liquidated damages.

However, examining the actual process of getting CB_i on any project plan produces a more complex formulation as below:

$$C_{LCC} = \sum_{k \geq 1 \bigcap Wave_k \neq \{\}} LCC^k = \sum_{k \geq 1 \bigcap Wave_k} \left(\sum_{i \in Wave_k} LCC_i^k \right) = \sum_{k \geq 1 \bigcap Wave_k \neq \{\}} \left(\sum_{i \in Wave_k} \left(CB_i^k + CD_i^k \right) \right),$$

$$CD_i^k = \sum_{j \in \{succeeding\_activities_i\}} \left(CB_j^k + CD_j^k \right), \quad CB_i^k, CD_i^k, CB_j^k, CD_j^k \geq 0$$

Where LCC^k = the cost for delaying all the activities with the k^{th} wave;

LCC_i^k = the cost for delaying the i^{th} activity with the the k^{th} wave;

CB_i^k = the cost for delaying the i^{th} activity, over and above the original cost within the k^{th} wave;

CD_i^k = the cost for delaying succeeding activities of the i^{th} activity, over and above the original cost within the k^{th} Wave.

Since CB_i is the cost for delaying the i^{th} activity, which accumulates all the CB_i^k, $\sum_{k_{wave}} CB_i^k = CB_i$. Therefore, the following formula generalize C_{LCC}:

$$C_{LCC} = \sum_{k \geq 1 \bigcap Wave_k \neq \{\}} LCC^k = \sum_{k \geq 1 \bigcap Wave_k} \left(\sum_{i \in Wave_k} LCC_i^k \right) = \sum_{k \geq 1 \bigcap Wave_k \neq \{\}} \left(\sum_{i \in Wave_k} \left(CB_i^k + CD_i^k \right) \right)$$

$$= \sum_{i=1}^{n} \left(\sum_{k_{wave}} CB_i^k \right) = \sum_{i=1}^{n} CB_i = \sum_{i=1}^{n} LCC_i,$$

$$CD_i^k = \sum_{j \in \{succeeding\_activities_i\}} \left(CB_j^k + CD_j^k \right), \quad CB_i^k, CD_i^k, CB_i, CB_j^k, CD_j^k \geq 0$$

E.1.3 Definition of total cost overruns of DCPSC

The following formula generalizes the total cost overruns (C_{DCPSC}) after DCPSC:

$$C_{DCPSC} = \sum_{k \geq 1 \cap Wave_k \neq \{\}} DCPSC^k = \sum_{k \geq 1 \cap Wave_k} \left(\sum_{i \in Wave_k} DCPSC_i^k \right) =$$

$$\sum_{k \geq 1 \cap Wave_k \neq \{\}} \left(\sum_{i \in Wave_k} \min\left(AC_i^k, \left(CB_i^k + DC_i^k \right) \right) \right), \quad AC_i^k, CB_i^k, DC_i^k \geq 0,$$

$$DC_i^k = \sum_{j \in \{succeeding\_activities_i\}} \min\left(AC_j^k, \left(CB_j^k + DC_j^k \right) \right), \quad AC_j^k, CB_j^k, DC_j^k \geq 0$$

Where $DCPSC^k$ = the minimum cost between the cost for expediting and the costs for delaying activities with the kth wave;

$DCPSC_i^k$ = the minimum cost between the cost for expediting and the costs for delaying activities from the *i*th activity with the kth wave;

DC_i^k = the minimum cost between the cost for expediting and the costs for delaying succeeding activities of the *i*th activity, over and above the original cost within the *k*th Wave.

E.2 Comparisons of Total Cost Overruns of TCC, LCC, and DCPSC

Theorem 1: $C_{DCPSC} \leq C_{TCC}$

Because $\min\left(AC_i^k, \left(CB_i^k + DC_i^k \right) \right) \leq AC_i^k$ by the definition of *min*,

$$C_{DCPSC} = \sum_{k \geq 1 \cap Wave_k \neq \{\ \}} \left(\sum_{i \in Wave_k} \min\left(AC_i^k, \left(CB_i^k + DC_i^k \right) \right) \right)$$

$$\leq \sum_{k \geq 1 \cap Wave_k \neq \{\ \}} \left(\sum_{i \in Wave_k} AC_i^k \right) = \sum_{i=1}^{n} AC_i = C_{TCC}$$

$$AC_i^k, CB_i^k, DC_i^k, AC_i \geq 0 \quad Q.E.D.$$

Theorem 2: $C_{DCPSC} \leq C_{LCC}$

LEMMA 1: $C_{DCPSC} \leq C_{LCC}$ when the activities follow a linear sequence, which means the n^{th} activity follows the $(n\text{-}1)^{th}$ activity, so on, as shown in Figure E-1.

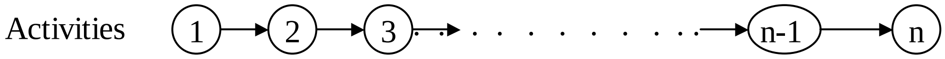


Figure E-1. Activity precedence network

Because $Wave_k$ = the k^{th} activity, then $i = k$. Then we can simplify C_{LCC} as follows:

$$C_{LCC} = \sum_{k=1}^{n} LCC^{k} = \sum_{k=1}^{n} LCC_{k}^{k} = \sum_{k=1}^{n}\left(\sum_{i=k}^{n} CB_{i}^{k}\right),$$

$$CB_{k}^{k}, DC_{k}^{k}, CB_{i}^{k}, CB_{i} \geq 0,$$

With the same procedure, we can simplify C_{DCPSC} as follows:

$$C_{DCPSC} = \sum_{k=1}^{n} DCPSC^{k} = \sum_{k=1}^{n} DCPSC_{k}^{k}$$

$$= \sum_{k=1}^{n} \min\left(AC_{k}^{k}, \left(CB_{k}^{k} + DCPSC_{k+1}^{k}\right)\right), \quad AC_{k}^{k}, CB_{k}^{k}, DC_{k}^{k} \geq 0,$$

Now we compare C_{DCPSC} with C_{LCC}.

Basis: $k = n$:

Because $\forall k, DCPSC_{n+1}^{k} = 0$, because there is no succeeding activity, $DCPSC_{n+1}^{n} = 0$.

$$DCPSC^{n} = DCPSC_{n}^{n} = \min\left(AC_{n}^{n}, \left(CB_{n}^{n} + DCPSC_{n+1}^{n}\right)\right) = \min\left(AC_{n}^{n}, CB_{n}^{n}\right)$$

$$\leq CB_{n}^{n} = \sum_{i=n}^{n} CB_{i}^{n} = LCC_{n}^{n} = LCC^{n}$$

When $k = n\text{-}1$:

Because $DCPSC_{n+1}^{n-1} = 0$,

$$DCPSC_n^{n-1} = \min\left(AC_n^{n-1}, \left(CB_n^{n-1} + DCPSC_{n+1}^{n-1}\right)\right) = \min\left(AC_n^{n-1}, CB_n^{n-1}\right),$$

$$\begin{aligned}
DCPSC^{n-1} &= DCPSC_{n-1}^{n-1} = \min\left(AC_{n-1}^{n-1}, \left(CB_{n-1}^{n-1} + DCPSC_n^{n-1}\right)\right) \\
&= \min\left(AC_{n-1}^{n-1}, \left(CB_{n-1}^{n-1} + \min\left(AC_n^{n-1}, CB_n^{n-1}\right)\right)\right) \\
&\le \left(CB_{n-1}^{n-1} + \min\left(AC_n^{n-1}, CB_n^{n-1}\right)\right) \le \left(CB_{n-1}^{n-1} + CB_n^{n-1}\right) = \sum_{i=n-1}^{n} CB_i^{n-1} = LCC_{n-1}^{n-1} = LCC^{n-1}
\end{aligned}$$

When $k = k$:

Given $\forall i,\ \ l \le i \le n,\ \ DCPSC_i^k \le LCC_i^k = \sum_{i=i}^{n} CB_i^k$, we prove that $DCPSC_{l-1}^k \le LCC_{l-1}^k$ by induction on the activity number within the same wave *k*.

$$\begin{aligned}
DCPSC_{l-1}^k &= \min\left(AC_{l-1}^k, \left(CB_{l-1}^k + DCPSC_l^k\right)\right) \\
&\le \left(CB_{l-1}^k + DCPSC_l^k\right) \le \left(CB_{l-1}^k + \sum_{i=l}^{n} CB_i^k\right) = \sum_{i=l-1}^{n} CB_i^k = LCC_{l-1}^k
\end{aligned}$$

The conclusion we draw is that $DCPSC^k$ is less than or equal to LCC^k for any activity number for $k \le i \le n$.

Therefore, $DCPSC^k = DCPSC_k^k \le LCC_k^k = LCC^k$

Then,

$$C_{DCPSC} = \sum_{k=1}^{n} DCPSC^k \le \sum_{k=1}^{n}\left(\sum_{i=k}^{n} CB_i^k\right) = \sum_{i=1}^{n} LCC_i = C_{LCC}, \quad Q.E.D.$$

Therefore, $C_{DCPSC} \le C_{LCC}$ when the activities follow a linear sequence.

LEMMA 2: If $C_{DCPSC} \le C_{LCC}$ for each string of linear activities, then this is so for the x^{th} activity where they fork also.

$$\sum_{j \in all\_suc\_linear\_activities_x} C_{DCPSC_j} \le \sum_{j \in all\_suc\_linear\_activities_x} C_{LCC_j} \qquad C_{DCPSC_j} \le C_{LCC_j}$$

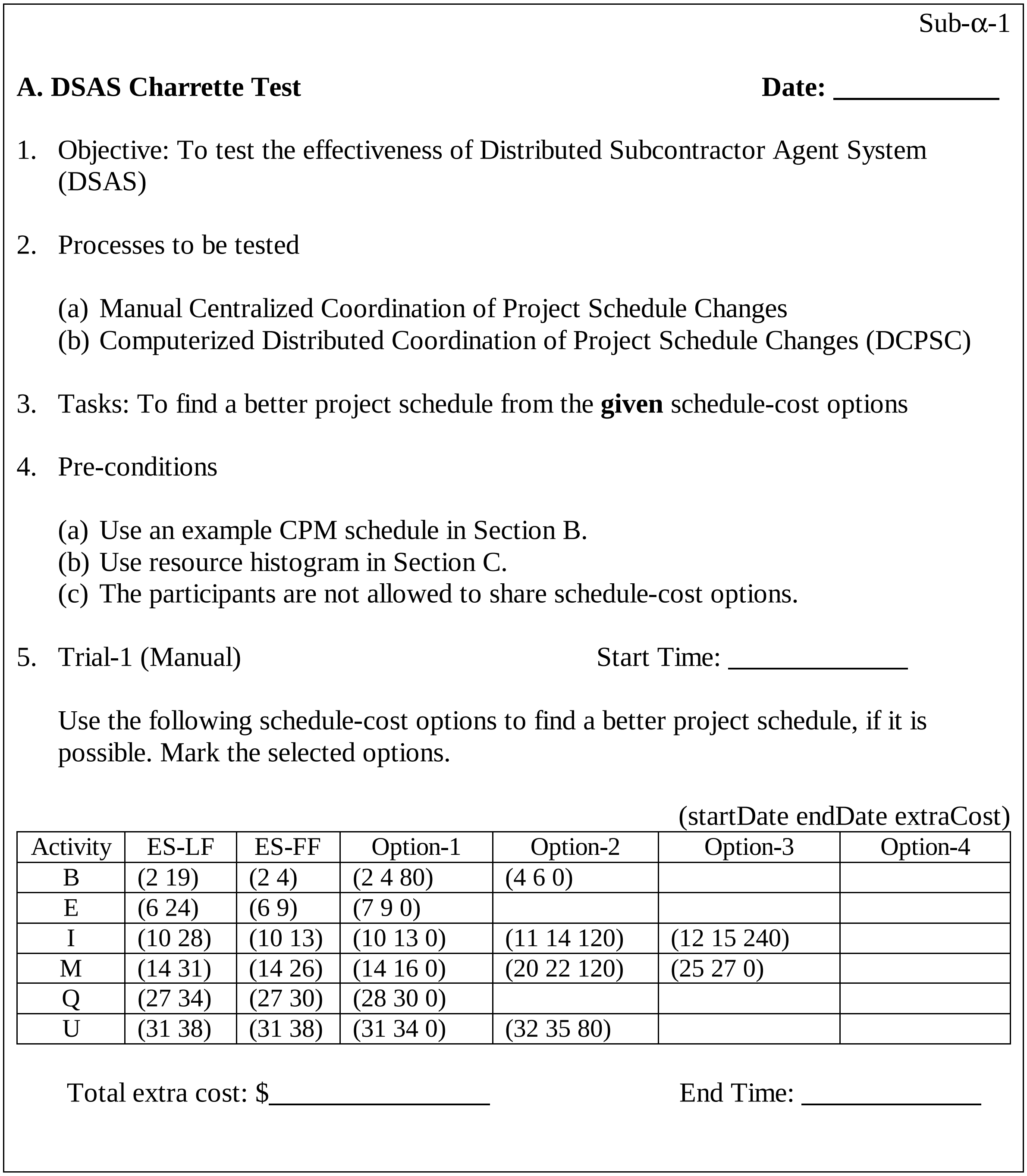
Since $C_{DCPSC} \leq C_{LCC}$ for each string of linear activities, then this is so for the activity where they fork also because the summation of C_{DCPSC} is less than or equal to the summation of C_{LCC} for the x^{th} activity where they fork.

Lemmas 1 and 2 show that $C_{DCPSC} \leq C_{LCC}$ within any project plan where the activities follow a linear sequence and the activity where they fork also.

Theorems 1 and 2 show that DCPSC always finds a solution that is better than or equal to that of any of two centralized coordination methodologies (TCC and LCC) since DCPSC selects the minimum value among the cost of expediting the activity (AC) and the cost for delaying the activity (CB).

APPENDIX-F. HANDOUT FOR DSAS CHARRETTE TEST

Figure F-1 shows a copy of handout for DSAS charrette test, which was prepared for Sub-α.



Sub-α-1

A. DSAS Charrette Test **Date: ____________**

1. Objective: To test the effectiveness of Distributed Subcontractor Agent System (DSAS)

2. Processes to be tested

   (a) Manual Centralized Coordination of Project Schedule Changes
   (b) Computerized Distributed Coordination of Project Schedule Changes (DCPSC)

3. Tasks: To find a better project schedule from the **given** schedule-cost options

4. Pre-conditions

   (a) Use an example CPM schedule in Section B.
   (b) Use resource histogram in Section C.
   (c) The participants are not allowed to share schedule-cost options.

5. Trial-1 (Manual) Start Time: _____________

   Use the following schedule-cost options to find a better project schedule, if it is possible. Mark the selected options.

(startDate endDate extraCost)

Activity	ES-LF	ES-FF	Option-1	Option-2	Option-3	Option-4
B	(2 19)	(2 4)	(2 4 80)	(4 6 0)		
E	(6 24)	(6 9)	(7 9 0)			
I	(10 28)	(10 13)	(10 13 0)	(11 14 120)	(12 15 240)	
M	(14 31)	(14 26)	(14 16 0)	(20 22 120)	(25 27 0)	
Q	(27 34)	(27 30)	(28 30 0)			
U	(31 38)	(31 38)	(31 34 0)	(32 35 80)		

Total extra cost: $________________ End Time: _____________

Figure F-1. Handout for DSAS charrette test (Sub-α)

Sub-α-2

6. Trial-2 (Computerized DSAS)   Start Time: ____________

Based on the tutorials in Section D, use the data in Section E to find a better schedule, if it is possible. Mark and **correct** the selected schedule-cost options based on the "final" messages, if it is changed.

(startDate endDate extraCost)

Activity	ES-LF	ES-FF	Option-1	Option-2	Option-3	Option-4
B	(2 19)	(2 4)	(2 4 80)	(4 6 0)		
E	(6 24)	(6 9)	(7 9 0)			
I	(10 28)	(10 13)	(10 13 0)	(11 14 120)	(12 15 240)	
M	(14 31)	(14 26)	(14 16 0)	(20 22 120)	(25 27 0)	
Q	(27 34)	(27 30)	(28 30 0)			
U	(31 38)	(31 38)	(31 34 0)	(32 35 80)		

Total extra cost: $_______________   End Time: ____________

7. Questionnaire

Please answer the following questions:

7.1 Your backgrounds

	Years?	Novice				Expert
Construction experience	yrs.	①	②	③	④	⑤
Computer experience	yrs.	①	②	③	④	⑤

7.2 How hard do you think were the following trials?

		Easy				Hard
Trial-1	Manual	①	②	③	④	⑤
Trial-2	Computerized DCPSC	①	②	③	④	⑤

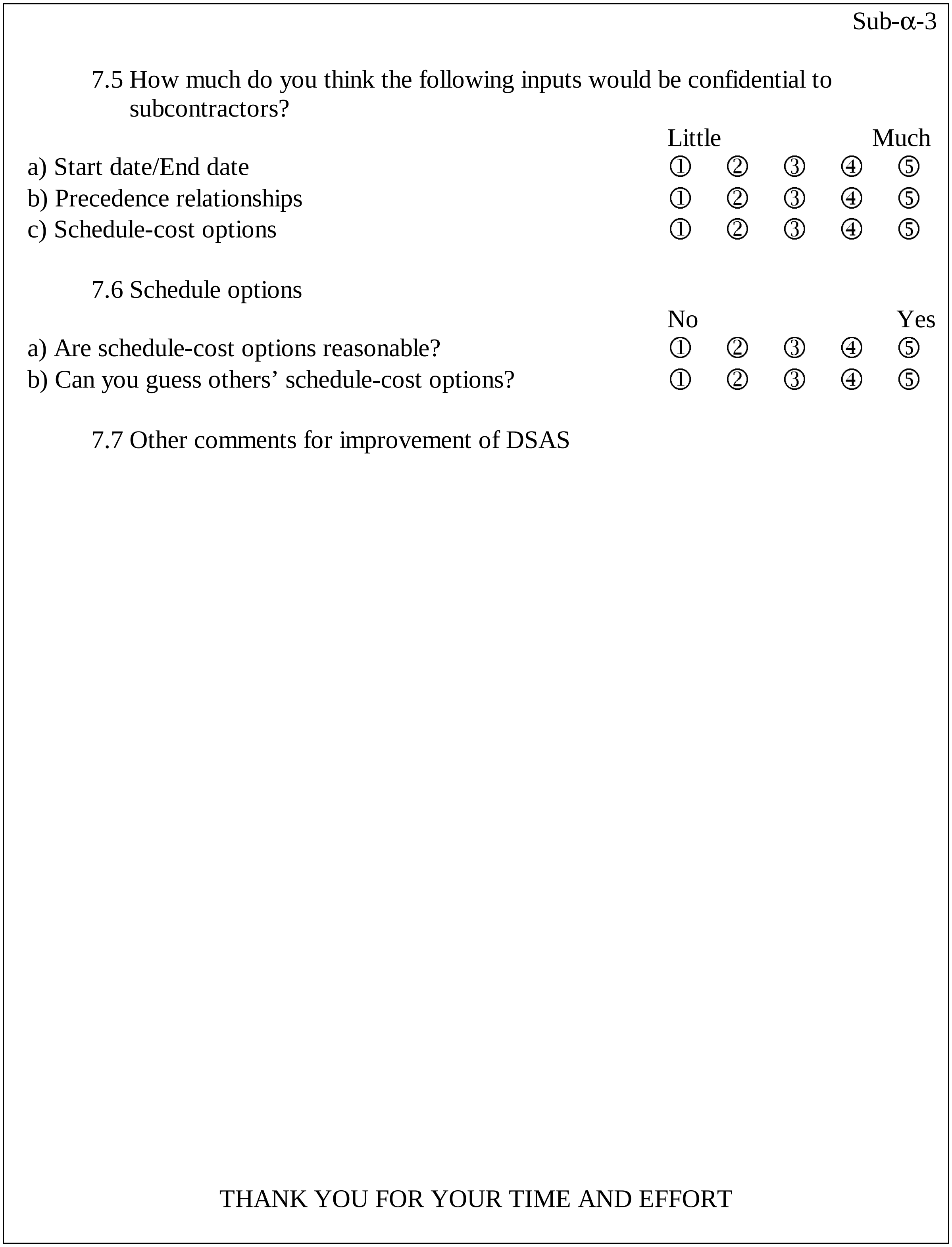
7.3 How well are you convinced on the results from the following trials?

		Little				Much
Trial-1	Manual	①	②	③	④	⑤
Trial-2	Computerized DCPSC	①	②	③	④	⑤

7.4 How well do you think the subcontractor will accept the results?

		Little				Much
Trial-1	Manual	①	②	③	④	⑤
Trial-2	Computerized DCPSC	①	②	③	④	⑤

Figure F-1. (Continued)



Sub-α-3

7.5 How much do you think the following inputs would be confidential to subcontractors?

	Little				Much
a) Start date/End date	①	②	③	④	⑤
b) Precedence relationships	①	②	③	④	⑤
c) Schedule-cost options	①	②	③	④	⑤

7.6 Schedule options

	No				Yes
a) Are schedule-cost options reasonable?	①	②	③	④	⑤
b) Can you guess others’ schedule-cost options?	①	②	③	④	⑤

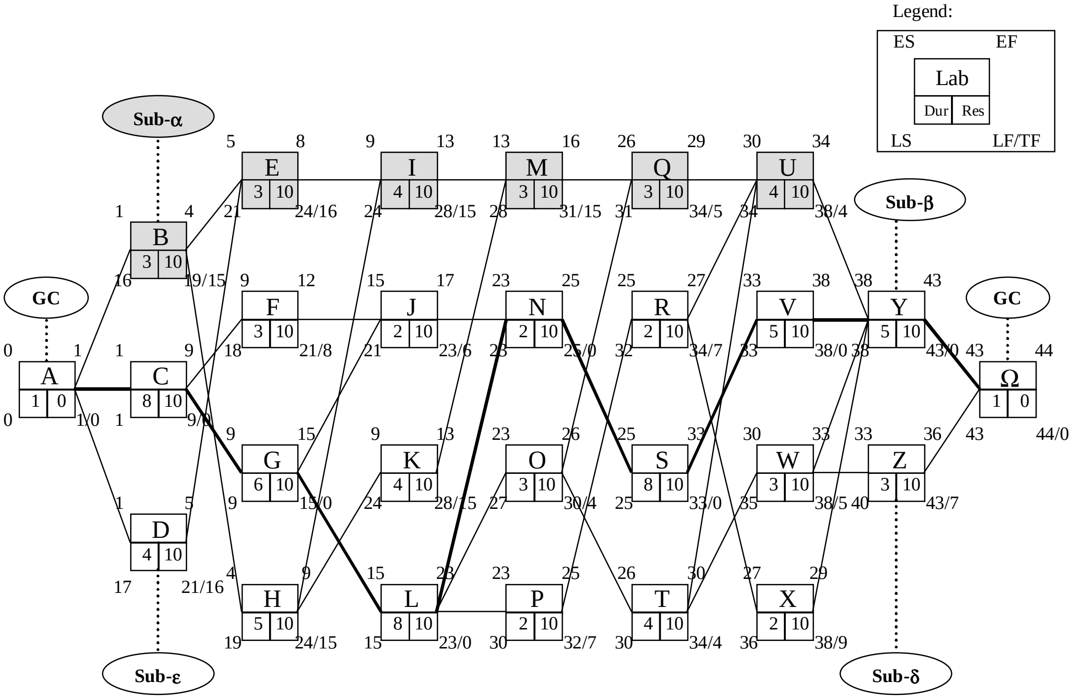
7.7 Other comments for improvement of DSAS

THANK YOU FOR YOUR TIME AND EFFORT

Figure F-1. (Continued)

Sub-α-4

B. CPM Schedule



C. Resource Histogram

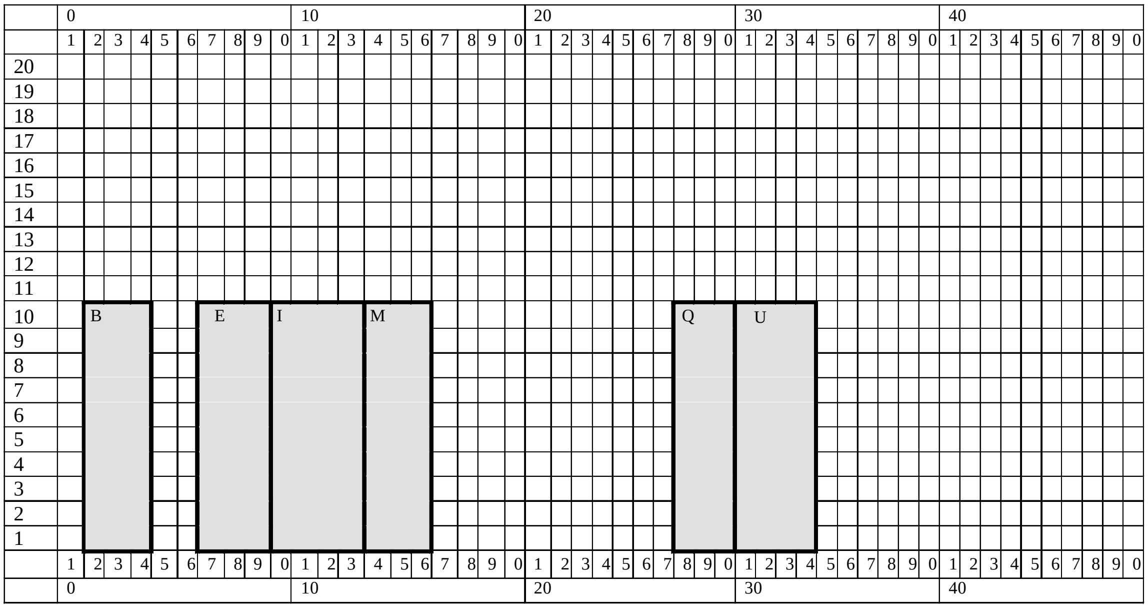


Figure F-1. (Continued)

Sub-α-5

D. Tutorial

This DSAS tutorial explains step-by-step instructions to run DSAS for Trial-2.

1. Flow Chart

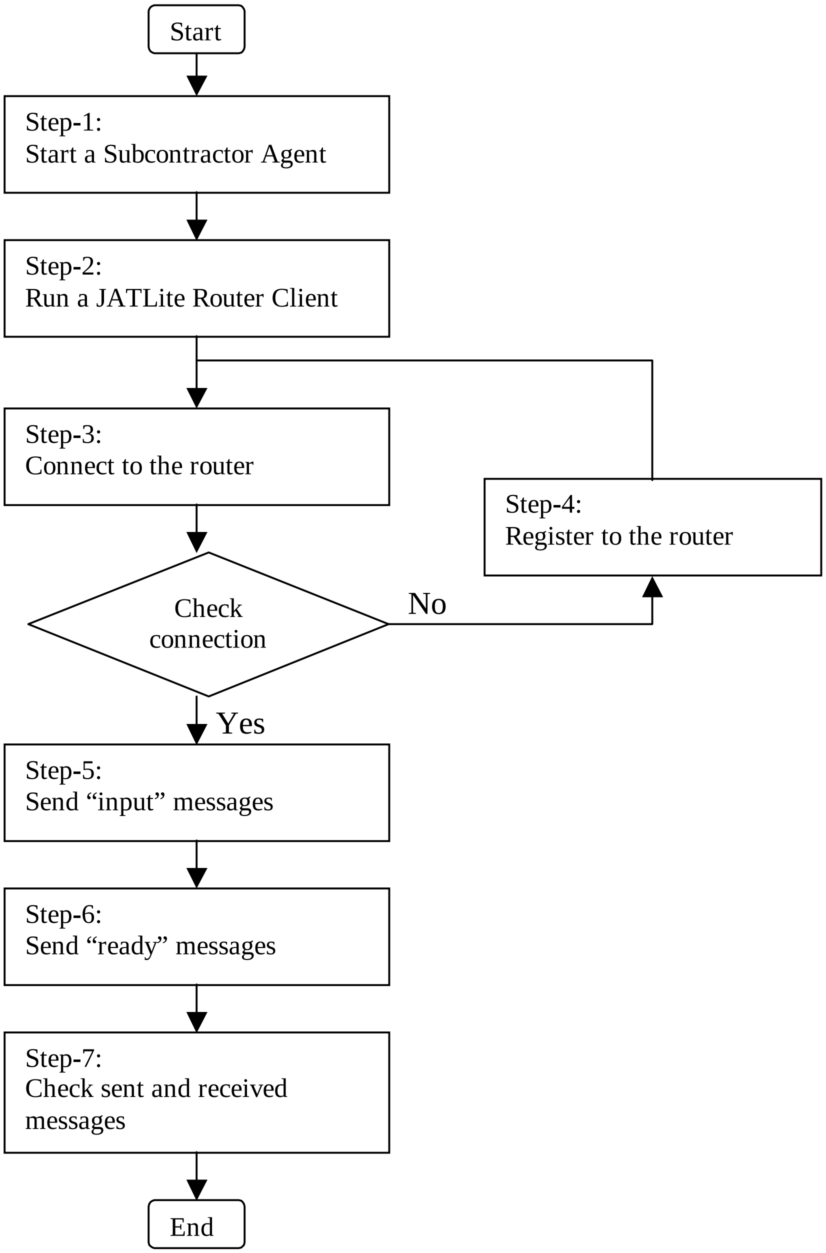


Figure F-1. (Continued)

Sub-α-6

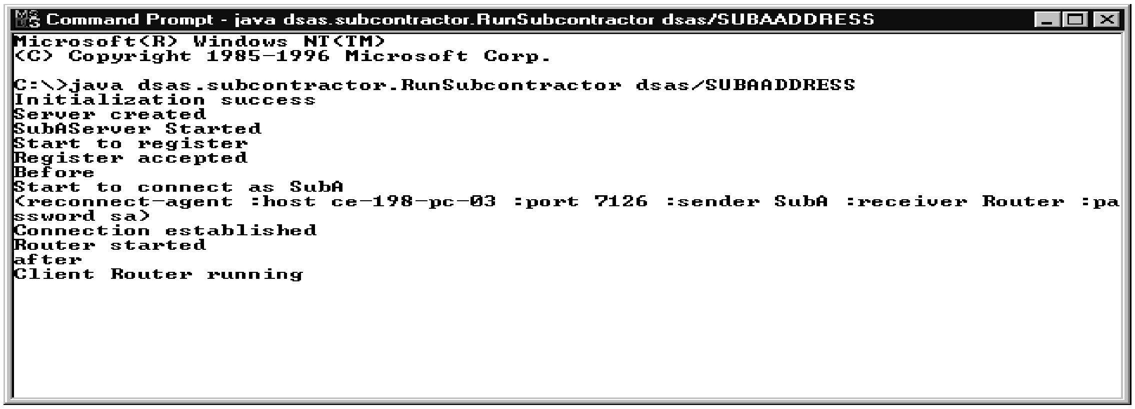
2. Step-by-Step Instructions

Step-1: Start a Subcontractor Agent

- Click **Start**, point to **Programs**, and then click **MS-DOS Prompt**.
- Type the following command in the root directory.

```
C:/>java dsas/subcontractor/RunSubcontractor dsas/SUBAADDRESS
```

- You will see following message (the last four lines) on the **Command Prompt** window.



Command Prompt - java dsas.subcontractor.RunSubcontractor dsas/SUBAADDRESS

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>java dsas.subcontractor.RunSubcontractor dsas/SUBAADDRESS
Initialization success
Server created
SubAServer Started
Start to register
Register accepted
Before
Start to connect as SubA
(reconnect-agent :host ce-198-pc-03 :port 7126 :sender SubA :receiver Router :pa
ssword sa)
Connection established
Router started
after
Client Router running
```

Step-2: Run a JATLite IPLayer Router Client

- Click **Start**, point to **Programs**, point to **JATLiteBeta**, and then click **IPRCApplet**.
- If successful, you can see **JATLite IPLayer Router Client** window as below.

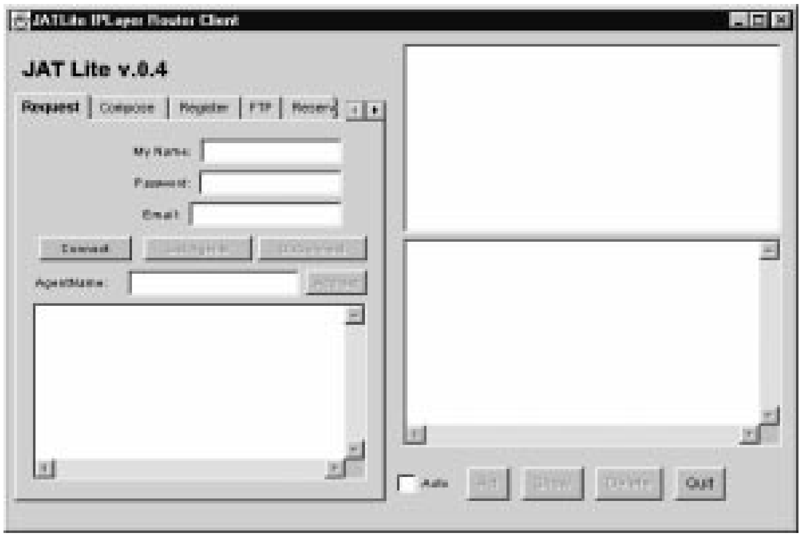


Figure F-1. (Continued)

Sub-α-7

Step-3: Connect to the router in the JATLite IPLayer Router Client window

- In ‘Request’ panel, fill in AgentName (**koo**) and Password (**koo**) and click ‘Connect’ button to connect to router.
- If successful, you will see ‘`Connection established`’ message is received. Then skip Step-4 and go to Step-5.

Step-4: Register to the router in the JATLite IPLayer Router Client window

- Go to ‘Register’ panel and fill in AgentName (**koo**) and Password (**koo**). AgentName and Password are case sensitive and there should be no white space.
- Click ‘Register’ button
- Go to ‘Request’ panel and make sure that ‘`Register accepted`’ message is received in the text area.

Step-5: Send “input” messages in the JATLite IPLayer Router Client window

- Go to ‘Compose’ panel and fill in Performative (**input**) and Receiver (**Sub-α**).
- Copy and paste one of DSAS input data in Content field from the top. Note that you do not need to type in open and close parenthesis in the content filed.
- Click ‘Send’ button and check the **Command Prompt** window.
- Repeat the previous two processes until there is no new input data.
- Check whether all “input” messages are sent.

Step-6: Send “ready” messages in the JATLite IPLayer Router Client window

- Change Performative (**ready**) in the ‘Compose’ panel.
- Copy and paste one of DSAS ready data in Content field from the top. Note that you do not need to type in open and close parenthesis in the content filed.
- Click ‘Send’ button and check the **Command Prompt** window.
- Repeat the previous two processes until there is no new input data.
- If completed, the subcontractor will start negotiation and does not allow any input until the negotiation cycle is finished, when your subcontractor agent will send ‘final’ messages to your IPRCApplet.

Step-7: Check sent and received messages in the JATLite IPLayer Router Client window

- Click once to select the message in the list box (upper right). Then click “show” button.
- Click once to un-select the read message.
- Click ‘final’ messages to check the results
- If **New Start Date**/ **New End Date** differs from the Option-1, check **Receivable**, **Payable**, and **Schedule Options**.

Figure F-1. (Continued)

Sub-α-8

E. Data

1. input human subcontractor → agent

(input
:sender koo
:receiver Sub-α
:content (AGENT(projStartDate projEndDate projPenalty subName))
(ACTIVITY(activityName startDate endDate){(preSubName preActivityName)*}{(sucSubName sucActivityName)*}{(startDate endDate extraCost)*})

Test inputs:
[**Sub-α**]---

AGENT(1 44 2000.0 koo)
ACTIVITY(B 2 4){(GC A)}{(Sub-α E)(Sub-ε H)}{(2 4 80)(4 6 0)}
ACTIVITY(E 6 9){(Sub-α B)(Sub-ε D)}{(Sub-α I)}{(7 9 0)}
ACTIVITY(I 10 13){(Sub-α E)(Sub-ε H)}{(Sub-α M)}{(10 13 0)(11 14 120)(12 15 240)}
ACTIVITY(M 14 25){(Sub-α I)(Sub-δ K)}{(Sub-α Q)}{(14 16 0)(20 22 120)(25 27 0)}
ACTIVITY(Q 27 30){(Sub-α M)(Sub-δ O)}{(Sub-α U)}{(28 30 0)}
ACTIVITY(U 31 38){(Sub-α Q)(Sub-β R)(Sub-ε T)}{(Sub-β Y)}{(31 34 0)(32 35 80)}

2. ready human subcontractor → agent

(ready
:sender koo
:receiver Sub-α
:content activityName)

Test inputs:
[**Sub-α**]---

B
E
I
M
Q
U

Figure F-1. (Continued)

APPENDIX-G. ANAYSIS OF RUNNING TIME

I estimate that the worst-case computational complexity of DSAS. The worst-case schedule is the schedule where all activities are sequential. This means that the nth activity follows the (n-1)th activity, so on, as shown in Figure G-1.

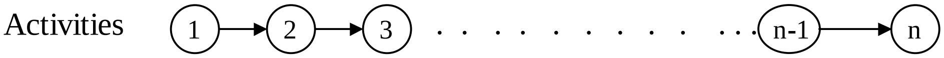


Figure G-1. Activity precedence network

Now I can use the inductive rules to analyze the approximate running time of a program, which is measured number of messages as a function of the number of activities, *n*, in a schedule. For this analysis of running time, I consider only the following message by agents: *ready*, *ask-cost*, *reply-cost*, *accept/reject*, *confirm/renege*, *accept-all/renege-all*, *hand-over*, *done*, and *final*. I exclude the *input* and *ready* messages because those messages are human-dependent. I also exclude *inform* messages because the *inform* messages are dependent to the negotiation messages.

Basis: *n* = 1:

Figure G-2. 1-activity precedence network

Message type	No. of message	Relationship with *n*
ready	0	
ask-cost	0	
reply-cost	0	
accept/reject	0	
confirm/renege	0	
accept-all/reject-all	0	
confirm-all/renege-all	0	
hand-over	0	
done	0	
final	1	n

Table G-1. 1-activity case analysis

Next n = 2:

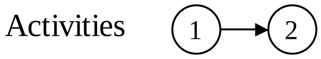


Figure G-3. 2-activity precedence network

Message type	No. of message	Relationship with *n*
ready	1	(n-1)
ask-cost	1	(n-1)+T(n-1)
reply-cost	1	(n-1)+T(n-1)
accept/reject	0	
confirm/renege	0	
accept-all/reject-all	1	(n-1)
confirm-all/renege-all	1	(n-1)
hand-over	1	(n-1)
done	1	(n-1)
final	2	n

Table G-2. 2-activity case analysis

Next n = 3:

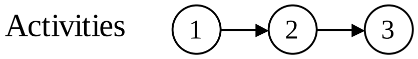


Figure G-4. 3-activity precedence network

Message type	No. of message	Relationship with *n*
ready	2	(n-1)
ask-cost	3	(n-1)+T(n-1)
reply-cost	3	(n-1)+T(n-1)
accept/reject	1	(n-2)+T(n-1)
confirm/renege	1	(n-2)+T(n-1)
accept-all/reject-all	3	(n-1)+T(n-1)
confirm-all/renege-all	3	(n-1)+T(n-1)
hand-over	2	(n-1)
done	2	(n-1)
final	3	n

Table G-3. 3-activity case analysis

Next n = 4:

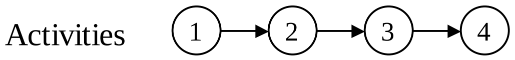


Figure G-5. 4-activity precedence network

Message type	No. of message	Relationship with *n*
ready	3	(n-1)
ask-cost	6	(n-1)+T(n-1)
reply-cost	6	(n-1)+T(n-1)
accept/reject	4	{(n-3)+(n-2)}+T(n-1)
confirm/renege	4	{(n-3)+(n-2)}+T(n-1)
accept-all/reject-all	6	(n-1)+T(n-1)
confirm-all/renege-all	6	(n-1)+T(n-1)
hand-over	3	(n-1)
done	3	(n-1)
final	4	n

Table G-4. 4-activity case analysis

Next n = 5:

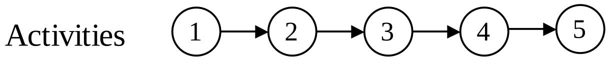


Figure G-6. 5-activity precedence network

Message type	No. of message	Relationship with *n*
ready	4	(n-1)
ask-cost	10	(n-1)+T(n-1)
reply-cost	10	(n+1)+T(n-1)
accept/reject	10	{(n-4)+(n-3)+(n-2)}+T(n-1)
confirm/renege	10	{(n-4)+(n-3)+(n-2)}+T(n-1)
accept-all/reject-all	10	(n-1)+T(n-1)
confirm-all/renege-all	10	(n-1)+T(n-1)
hand-over	4	(n-1)
done	4	(n-1)
final	5	n

Table G-5. 5-activity case analysis

Next n = 6:

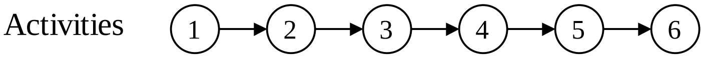


Figure G-7. 6-activity precedence network

Message type	No. of message	Relationship with *n*
ready	5	(n-1)
ask-cost	15	(n-1)+T(n-1)
reply-cost	15	(n-1)+T(n-1)
accept/reject	20	{(n-5)+(n-4)+(n-3)+(n-2)}+T(n-1)
confirm/renege	20	{(n-5)+(n-4)+(n-3)+(n-2)}+T(n-1)
accept-all/reject-all	15	(n-1)+T(n-1)
confirm-all/renege-all	15	(n-1)+T(n-1)
hand-over	5	(n-1)
done	5	(n-1)
final	6	n

Table G-6. 6-activity case analysis

Next n = 7:

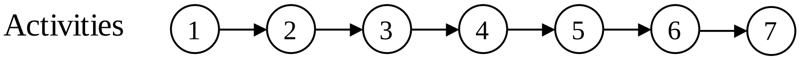


Figure G-8. 7-activity precedence network

Message type	No. of message	Relationship with *n*
ready	6	(n-1)
ask-cost	21	(n-1)+T(n-1)
reply-cost	21	(n-1)+T(n-1)
accept/reject	35	{(n-6)+(n-5)+(n-4)+(n-3)+(n-2)}+T(n-1)
confirm/renege	35	{(n-6)+(n-5)+(n-4)+(n-3)+(n-2)}+T(n-1)
accept-all/reject-all	21	(n-1)+T(n-1)
confirm-all/renege-all	21	(n-1)+T(n-1)
hand-over	6	(n-1)
done	6	(n-1)
final	7	n

Table G-7. 7-activity case analysis

By this point, I can guess for n = n:

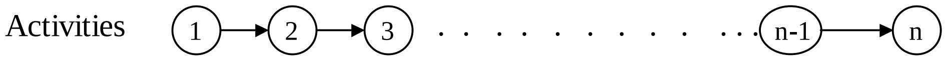


Figure G-9. N-activity precedence network

Message type	No. of message	Relationship with *n*
ready	(n-1)	(n-1)
ask-cost	$\sum_{i=2}^{n}(i-1)$	(n-1)+T(n-1)
reply-cost	$\sum_{i=2}^{n}(i-1)$	(n-1)+T(n-1)
accept/reject	$\sum_{i=3}^{n} g(i), g(i) = \sum_{j=1}^{i-2} j$	{(1)+(2)+(3)+ . . . .+(n-2)}+T(n-1)
confirm/renege	$\sum_{i=3}^{n} g(i), g(i) = \sum_{j=1}^{i-2} j$	{(1)+(2)+(3)+ . . . .+(n-2)}+T(n-1)
accept-all/reject-all	$\sum_{i=2}^{n}(i-1)$	(n-1)+T(n-1)
confirm-all/renege-all	$\sum_{i=2}^{n}(i-1)$	(n-1)+T(n-1)
hand-over	(n-1)	(n-1)
Done	(n-1)	(n-1)
Final	N	n

Table G-8. N-activity case analysis

Therefore,

T(n) for *ready* = (n-1) = O(n);

T(n) for *ask-cost* = (1 + n)(1 + (n –1))/2 = O(n^2);

T(n) for *reply-cost* = (1 + n)(1 + (n –1))/2 = O(n^2);

T(n)for *accept/reject* = (2 + n)[{(n - 2)(1 + (n – 2))/2}+1]/2 = O(n^3);

T(n) for *confirm/renege* = (2 + n)[{(n - 2)(1 + (n – 2))/2}+1]/2 = O(n^3);

T(n) for accept-all/reject-all = (1 + n)(1 + (n –1))/2 = O(n^2);

T(n) for *confirm-all/renege-all* = (1 + n)(1 + (n –1))/2 = O(n^2);

T(n) for *hand-over* = (n-1) = O(n);

T(n) for *done* = (n-1) = O(n);

T(n) for *final* = (n) = O(n).

Therefore, the sum of T(n) is O(n^3) because T(n) for “accept/reject” or “confirm/renege” grows rapidly so that we can neglect other lower O(n^2) or O(n).