The Pennsylvania State University

The Graduate School

Department of Industrial Engineering

Market-based Dynamic Resource Control of Distributed Multiple Projects

A Thesis in

Industrial Engineering

by

Yong-Han Lee

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August, 2002

We approve the thesis of Yong-Han Lee.

Date of Signature

Soundar R. Tirupatikumara
Professor of Industrial Engineering
Thesis Adviser
Chair of Committee

Kalyan Chatterjee
Distinguished Professor of Management Science and Economics

Natarajan Gautam
Assistant Professor of Industrial Engineering

Timothy W. Simpson
Assistant Professor of Industrial Engineering

Richard J. Koubek
Professor of Industrial Engineering
Head of the Department of Industrial Engineering

Abstract

Project oriented workflow is common in modern industry. Recently, the proliferation of Internet technology and globalization of business environment give rise to the advent of dynamic virtual alliances among complementary companies, thereby forming an *extended virtual enterprise.* Resource scheduling and rescheduling of multiple projects carried out in this situation cannot be effectively supported by existing centralized project scheduling approaches, mainly due to its dynamic and decentralized nature. The need for supporting such organizationally, geographically, and computationally distributed multiple projects (DMP) calls for a novel solution approach, which is distributed, collaborative, and flexible. In this research we have developed a market-based approach to address the DMP problem.

In the market-based DMP scheduling approach, the agents - representing project managers, tasks, resource managers, and resources - constitute a dynamic virtual economy, where multiple *local markets* are established and cleared over time. Like any other resource-constrained scheduling problem, the DMP scheduling problem also has to handle *precedence constraints* and *resource constraints.* In each local market, resource time slots are evaluated throughout by a *combinatorial auction mechanism*, generating resource-feasible local schedules for each resource. Meanwhile, in order to generate precedence-feasible schedules for each project, a *tâtonnement* type procedure is applied to the temporary economy, which is a set of local markets. This iterative mechanism, called P-TÂTO, generates convergent and globally efficient DMP schedules. P-TÂTO is run in a multiagent based information system infrastructure. We design and implement the multiagent system (MAS) for the DMP control economy. The MAS design and implementation issues - including individual agent model, agent communication, and DMP control ontology - are thoroughly discussed. The solution qualities are verified through an intensive empirical analysis. The analysis results support the high levels of solution quality and computational efficiency of the mechanism.

Our approach can be directly applied to many project-oriented, decentralized scheduling problems. In addition, our problem formulation can also be considered as a decentralized extension of general resource allocation or scheduling problems. Application of our problem-solving model to other decentralized resource allocation and scheduling applications, including military logistics, supply chain management, and survivability analysis of networks are potential areas of future research.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my gratitude to my advisor Dr. Soundar Kumara for his support and encouragement throughout my Ph.D. study at the Pennsylvania State University. He is an enthusiastic researcher, a patient teacher, and most of all a warm-hearted person. I have tried to learn all of his virtues for the last four years since I started working with him. I would also like to thank my committee members Dr. Kalyan Chatterjee, Dr. Natarajan Gautam and Dr. Timothy Simpson, for their invaluable guidance and suggestions during my research.

In finishing my "Ph.D." study, I would like to give special thanks to three "Ph.D.'s" who deeply influenced me in many ways. They are Dr. Byoung-Kyu Choi (at KAIST), Dr. Woo-Jong Lee (in Daewoo Motor), and the late Dr. Inyong Ham (at Penn State). When I got troubles during my Ph.D. study, I used to imagine myself one of them and think what to do. Most of the cases, I could get confident answers. Although they have not given me a word of advise on my Ph.D. research, they indeed made a great contribution toward my finishing Ph.D. study.

I am indebted to my wife Sunghee and daughter Hayeon for their patience. They had to sacrifice many aspects of their lives due to my Ph.D. study in the U.S. They would never know how much I feel sorry and how much I love them. Finally, I would like to thank my parents and parents in law, from the bottom of my heart, for their prayers and confidence in me. I am really proud of them all.

Chapter 1

Introduction

The rapid change in both technology and the marketplace in recent years has called for a new paradigm for managing large distributed projects. Globalized product development projects fall into this category. In order to compete in the highly competitive marketplace, modern enterprises are forced to compress their product development lead time. In addition, projects have become more and more distributed due to economic reasons, and project fulfillment processes overlap. Therefore project management has become more difficult. In these situations, traditional project management techniques cannot provide the required functionality to react effectively to the changes and to support collaborative project management processes in dispersed project environments. For the sake of clarity of problem definition, we wish to characterize a representative project environment in modern enterprises, called *Distributed Multiple Projects (DMP)*. As the name implies, DMP type enterprises run multiple projects, product-development projects for example, simultaneously by using distributed functional divisions. The unique characteristics of DMP environments are as follows:

1. *Frequent Change Inevitability*: Due to the dynamic changes in the market situation and a company's marketing strategy, product designs and product launch schedules frequently change. As a result, original plans, which are based on estimation, are never accomplished as planned and daily re-planning becomes a routine managerial activity rather than an exception.

2. *Tightly Coupled Tasks*: The DMP type of projects usually have tight resource constraints, unlike other process or task centered projects such as R&D projects, construction projects and ship building projects. Due to these resource constraints along

with precedence constraints in individual projects, the task schedules are tightly coupled, meaning that any change in a task can cause the needs to change other tasks.

3. *Disciplinary Self-interestedness*: A project goal is achieved throughout a series of functional divisions - they can be either internal divisions or contract based external organizations. Usually the divisions have their own self-interests, for example, all *project-oriented divisions*[1] want their products or component parts to be scheduled and finished on time; *resource-oriented divisions*[2] are more interested in efficient resource utilization and work load leveling. As a result, a great deal of coordination is required to solve conflicts between self-interested divisions. According to Petrie (1998), a result from a survey at Boeing showed that 60% of the labor is devoted to operational task coordination.

4. *Information/control Distribution*: As divisions get distributed organizationally, geographically and/or computationally, local divisions maintain local information - not only factual data but also process and planning knowledge. In addition, as the scale of a project grows, more and more decisions are made by local organizations. The local organizations can obviously make more beneficial decisions for their local goals because they have up-to-date and precise information on their local situations (although their decisions can conflict with others').

1.1 Drawbacks of Traditional Approaches

Existing project management tools fall short of supporting the management of DMP. Following are some significant functional deficiencies of the current project management tools for supporting DMP environments.

[1]The functional divisions that are dedicated to specific projects.

[2]The functional divisions that are shared by different projects by supporting specific functionality using their resources.

1. *Lack of Distributed Management Support*: Most of the current project support tools (software + algorithms) work only over a local area network and they cannot support globally distributed project environments. Even though some of the project management tools support a distributed environment through the network, they are still based on a single-user model of planning and control and are not capable of supporting distributed decision makers (Drabble, 1995; Petriea et al., 1999).

2. *Lack of Coordination Support*: Because of the self-interestedness of the organizations, a great deal of coordination (or negotiation when conflicts occur) effort is required in DMP planning and control. In this coordination problem, criticality of a task and the resulting global benefit of assigning higher priority to the task must be balanced. This kind of quantitative analysis, however, is often missed during the negotiation process because current project management tools do not support negotiation mechanisms and do not provide for sufficient level of *what-if* analysis. Thus, the coordination processes usually rely on human interactions without any computational support.

3. *Lack of Task Level Control*: Most project management systems consider a project as a set of activities to achieve a goal within the precedence and resource constraints, maintaining a single huge project network. However, as the size and number of projects increase, the traditional project management tools can hardly support the detailed task level control due to the huge number of tasks. As a result, they group the tasks to reduce the number of elementary activities in a project. This aggregate project management will work for the initial/master plan, but not for operational resource control. In addition, these aggregate-based project management system causes another significant problem - the *lack of interoperability*. Namely, they can hardly be integrated with other information systems such as *work flow management* and *product data management*, due to different levels of information and aggregation.

4. *Lack of Planning and Control Knowledge Maintenance Support*: The efficiency of the DMP planning and control, which is very labor intensive and time consuming, depends on the human planner's knowledge, which is gathered from experience. This is because project management tools do not maintain the planning and controlling

rationale (or more fundamentally these tools do not plan by themselves). As a result, the enterprise-level objectives can hardly be applied to everyday operational decision making in a consistent way.

Due to these drawbacks, the existing project management tools support the DMP management problem only in limited ways such as: (i) supporting just a single user who is in charge of project planning and control, causing the system to be used as a personal assistant by planners, rather than as a cooperative management tool; (ii) managing projects in an aggregate manner, thereby, losing detailed task level control; (iii) not being able to maintain consistent project management rationale and (iv) relying on face-to-face conflict resolution, even for daily operational re-planning decisions. The above discussion points to the need for developing approaches for supporting effective management of DMP.

1.2 Problem Discussion

In this thesis we will address the problem of dynamic resource allocation in the DMP execution mode. We will denote this problem as *DMP resource control*[3]. Though this can be thought of as rescheduling of distributed projects, we make a clear distinction in the sense that short-term rescheduling in DMP environment is more like execution control. In our context, control is more a frequent and routine activity compared to occasional rescheduling. The formal problem specification is given in Chapter 2. In the following sections of this chapter, we illustrate the planning and control problem of DMP by means of an example and present our research objectives.

1.2.1 A DMP Example

Globalized automotive manufacturing is a classic example of a DMP type of enterprise. Consider an automotive manufacturing company having four technical centers in Korea, Italy, England and Germany, say TCK, TCI, TCE and TCG respectively. Every technical center has multiple resource divisions internally and the functionality of each

[3]We often use *DMP control* rather than DMP resource control for convenience in this thesis.

technical center is not mutually exclusive among the four technical centers, meaning that their resource types overlap. Figure 1.1 shows an example of the DMP organizational structure. On the basis of the company's platform line-up, four basic car programs are being carried out simultaneously (projects a, b, c and d). Each car platform has *major-change* projects once every four years and *minor-change* projects in between, meaning that a new project starts every six months, and the project periods significantly overlap.

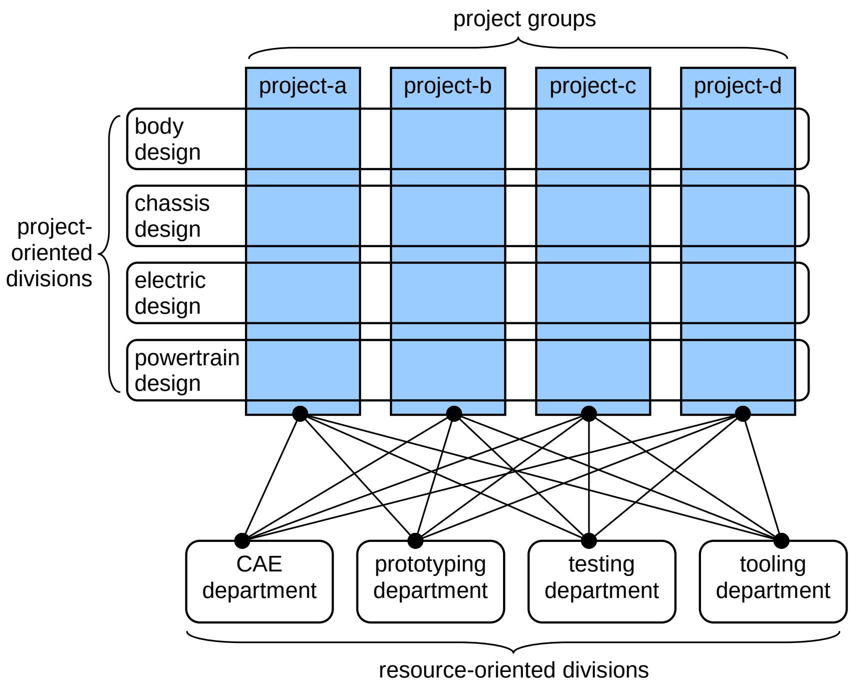


Fig. 1.1. An example DMP organization

- *Planning for new projects*: At the initial stage of each project, the top management decides two basic goals from resource allocation problem's point of view - numbers of *milestones* and the *assignment of divisions* in an aggregate manner. On the basis of these fundamental goals and constraints, the project group, which consists of a project manager, planning staff and other project oriented divisions such as chassis design or body design divisions, starts planning the project. At this point, the resource-oriented divisions, who have already been supporting ongoing project(s), start to schedule and/or reschedule their resources to support the new project. These decisions are made mostly by a fewer number of planning staff by taking estimations and strategic/operational restrictions into account.

- *Control of ongoing projects*: Other than the new project planning situation, the project groups and resource oriented divisions keep facing changes with respect to the plans and have to adapt their plans to these changes. For example, a project's milestones may be changed for some strategic reason, a part must be redesigned due to the failure in a test, or a prototyping process may take far more time than expected because of a machine failure. Sometimes just a few adjustments may be enough, but it may generate more complicated situations that affect other projects' schedules.

1.2.2 Dynamic Resource Control of DMP

The DMP resource control problem, which this thesis mainly addresses, is an $\mathcal{NP}$-hard[4] problem, as far as optimality of the control procedure is concerned. The main difficulty of the problem arises from the tight coupling between elementary tasks via *precedence* and *resource* constraints. Hence, in practice, relying on face-to-face negotiation, decisions are made based on: (1) the bounded (local) information about the resultant effect of the re-allocation and (2) inconsistent allocation rules. In addition, decision making requires coordination efforts and time delays.

The control problem arises due to the self-interests of divisions conflicting with each other, no matter whether they are project-oriented or resource-oriented. Namely, each project group tries to secure enough resources to achieve their goals with a higher probability. However, due to resource restrictions we can rarely obtain a solution satisfying every group. In addition, it is even unclear which choice (in every day operational decision making) is better than the others from the enterprise's perspective. The problem is how to *fairly* and *consistently* reschedule the tasks at the operational level.

[4]DMP resource control problem, which is formally specified in Chapter 2, is a resource-constrained project schedule problem (RCPSP), which is known as $\mathcal{NP}$-hard (Kolisch and Padman, 2001).

From the top management's perspective the relative importance or urgency of each project (ultimately those of each task) could clearly be determined. The problem is how such an overall priority can be deployed in a consistent way all the way down to the task level operational scheduling and rescheduling activities.

Now think about how one can handle such a dynamic resource control problem in such an uncertain situation? In order to solve the DMP resource control problem, a distributed, collaborative, and adaptive resource control approach, which guarantees the fairness with respect to the global objectives is required.

1.3 Market / Multiagent based Solution Approach to DMP Problem

Due to the nature of the DMP resource control problem as discussed in the previous section, it is very natural to model this problem as a negotiation process between competitive participants who need to acquire some resources to achieve their individual goals. In such situations, the value (or price) of the commodities must be determined based on the market principle - the higher the demand, the higher the price. In other words, the DMP resource control problem can be formulated as a *market-based resource allocation problem.* In order to facilitate the market based approach it is necessary to have a computational architecture. The DMP can be facilitated through a *multiagent system* (MAS). The important observations regarding these two approaches are:

- Market-base mechanisms and multiagent systems are inherently distributed in nature, so the DMP decision makers and decision making process can be naturally implemented by these approaches.
- Each project group and resource-oriented division can easily be modeled as a self-interested autonomous decision maker, who try to maximize its own objective.
- The simple mechanism based on bidding and auction can reduce the communication burden in this distributed negotiation environment and increase the system robustness.

- The modularity and adaptability of the market-based and multiagent approach reduces the implementation and maintenance efforts in the frequently changing DMP environment.

1.4 Research Objectives

The research objectives in the proposed research are: (1) development of a multiagent-based information system model, and (2) development of market-based DMP resource control mechanism, which works within the information system infrastructure.

1. *The development of a multiagent-based information system model includes*:
   - An organizational model such as the role definition of individual agents in the DMP resource control system;
   - Design of an individual agent's architecture, which is adequate for DMP resource control problem solution model; and
   - Agent communication including the definitions of agent-agent interactions and DMP project resource control ontology.
2. *Development of a market-based DMP resource control mechanism involves*:
   - A virtual economy model for resource allocation, which includes the definitions of goods, buyers and sellers;
   - An utility function that incorporates the relative value of a good from the individual project's or task's perspective; and
   - Detailed market protocols consisting of an auction mechanism and bidding policies.

1.5 Organization of the Thesis

In Chapter 2, the DMP resource control problem is formally defined. In Chapter 3, we briefly review the background literature on the distributed resource allocation in the context of project management and market-based mechanism. A market-based planning and control mechanism for DMP are presented in Chapter 4 in detail. A multiagent-based information system model is presented in Chapter 5. Chapter 6 deals with testing and verification of the proposed approach for solving the DMP resource control problem. Chapter 7 concludes the research followed by future research recommendations. Figure 1.2 summarizes the road map of research and this thesis.

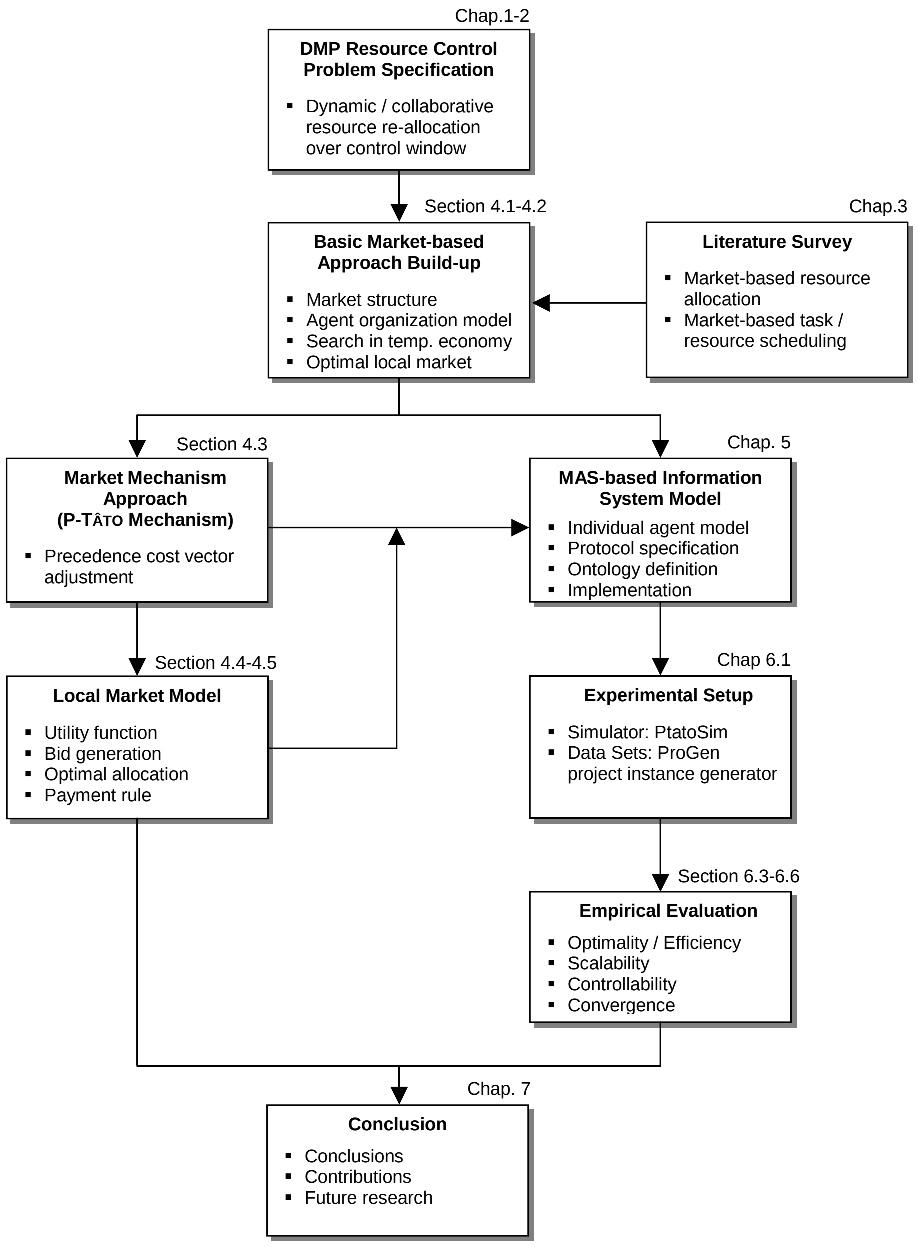
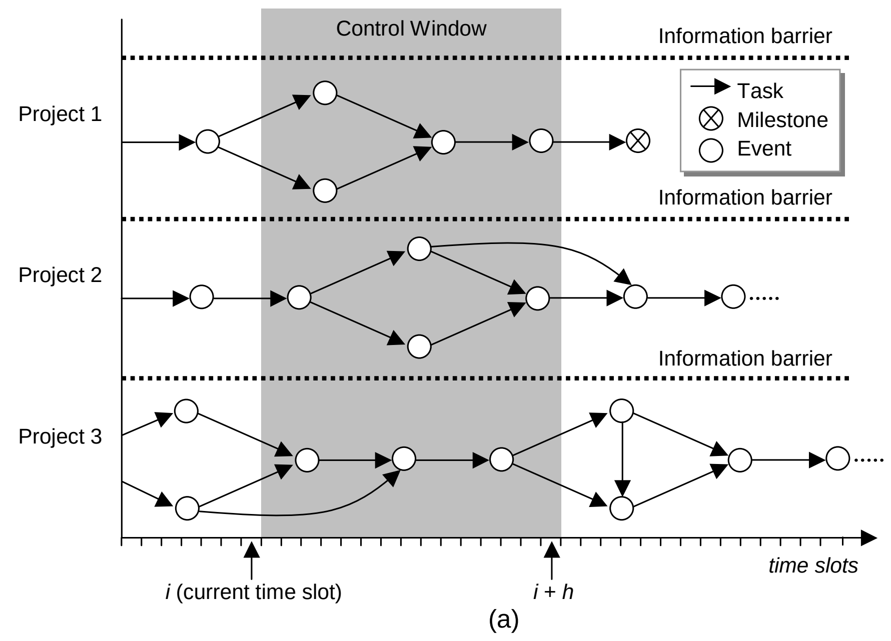


Fig. 1.2. Research road map and thesis organization

Chapter 2

DMP Resource Control Problem

In Chapter 1 we roughly discussed the background and objectives in DMP resource control problem. Figure 2.1 illustrates the overall structure of the DMP resource control problem. From the project groups' view points (Figure 2.1(a)), we have multiple simultaneous projects, which consist of hundreds or thousands interrelated project activities (or *tasks*) organized by a project activity network. There are informational and managerial barrier among projects and precedence constraints among individual tasks (defined by the project activity network). Meanwhile, the resources are to be shared by the tasks as shown in Figure 2.1(b). The resource time over the given (moving) control window must be dynamically re-allocated throughout the DMP resource control process. As shown in the figure, the continuous resource time is divided into equilength time slots, which is a common practice both in resource scheduling literature and in real practice in industry. The typical time-slot size is one hour in an operational manufacturing scheduling situation. In the following sections of this chapter we formally define fundamental terms, and ultimately the difinitoin of DMP resource control problem. In addition, a centralized Integer Programming formulation is explained, which is used for the purpose of solution optimality comparison. At the end of this chapter, we discuss solution performance measures of the DMP resource control problem formulation.



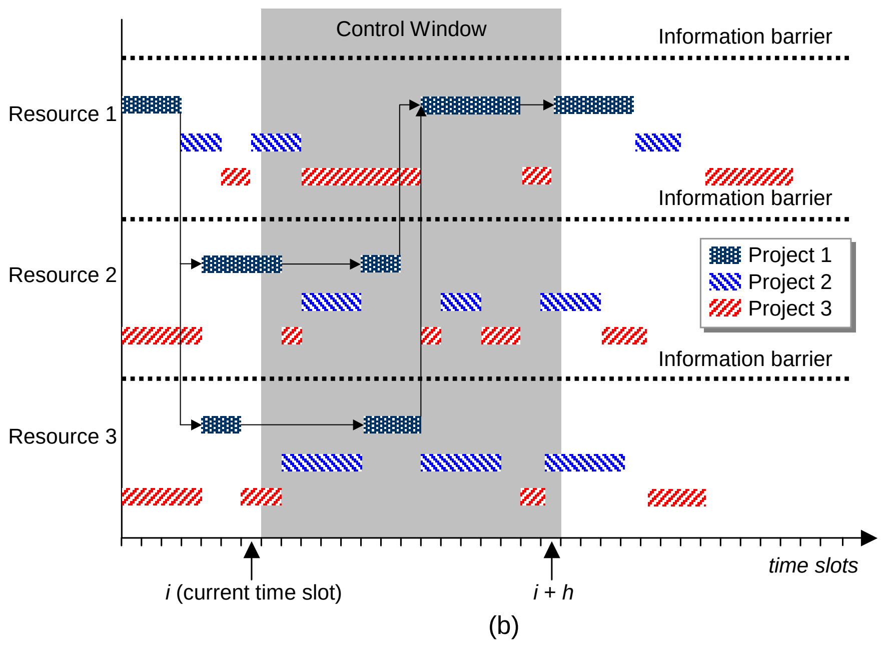


Fig. 2.1. A simplified DMP resource control problem: (a) project view, (b) resource view

2.1 Problem Specification

2.1.1 DMP environment

We define two components that constitute a DMP problem environment: (a) a set of projects and (b) a set of resource divisions. First, we define a project as a set of tasks, milestones and precedence constraints in the following definition. As shown in the definition, a project $\mathcal{P}_i$ may have multiple milestones or goals, $\mathcal{G}_i = \{G_{i1}, G_{i2}, ..., G_{i|\mathcal{G}_i|}\}$, including the target project completion date G_{i1}. For the sake of simplicity $|\mathcal{G}_i| = 1\ \forall i$ is assumed in this thesis. The precedence constraints can be defined by task pairs (predecessor, successor), i.e., $\mathcal{N}_i = \{(T_{ij}, T_{ik}) : j \neq k,\ T_{ij}, T_{ik} \in \mathcal{T}_i\}$.

Definition 2.1. (project) A *project* $\mathcal{P}_i$ is a 3-tuple $\langle \mathcal{G}_i, \mathcal{T}_i, \mathcal{N}_i \rangle$, where $\mathcal{G}_i$ is a set of milestones, $\mathcal{T}_i$ is a set of all tasks to be processed in the project, and $\mathcal{N}_i$ is the set of the precedence constraints defined on $\mathcal{T}_i$. □

On the other hand, we have multiple resource divisions, which have resources under their control as defined in the following definition. As shown in the definition, each resource R_{jk} has a capability to perform a specific task type $q \in \mathcal{Q}$, where $\mathcal{Q}$ is the universal set of task types.

Definition 2.2. (resource division) A *resource division*, $\mathcal{D}_j$, is an organizational unit that comprises a set of resources, $\mathcal{R}_j = \{R_{j1}, R_{j2}, ..., R_{j|\mathcal{R}_j|}\}$. All R_{jk} in $\mathcal{D}_j$ are located at a common physical place and under a common control exclusively, i.e., every $\mathcal{D}_j$ has its own manager or control. □

Based on these two definitions, a DMP environment can be defined as a combined set of multiple projects and multiple resource groups as follows.

Definition 2.3. (distributed multiple project) A *Distributed Multiple Projects (DMP)* environment, Π, is defined by a set of projects, $\mathcal{P} = \{\mathcal{P}_i\}$, and a set of resource divisions, $\mathcal{D} = \{\mathcal{D}_j\}$, i.e., $\Pi = \langle \mathcal{P}, \mathcal{D} \rangle$. □

2.1.2 Deviation Cost Function and Project Weight Distribution

In order to define the concept of a *project schedule*, we need to define beforehand a measure that decides on the superiority of one project schedule with respect to another. For this purpose, we define a solution quality measure of a project schedule, called *deviation cost function*, which is a generalization of project tardiness cost functions.

Definition 2.4. (deviation cost function) A *deviation cost function*, $\phi_i : \mathbb{R} \to \mathbb{R}_+$, of a project $\mathcal{P}_i$ is a function in the time domain, representing the penalty cost due to the deviation of the project completion time from the original completion target, G_{i1}. $\square$

Obviously a smaller value is preferable to a bigger value in this function, i.e., if the completion times of two project schedules (see Definition 2.9) $\mathcal{E}'_i$ and $\mathcal{E}''_i$ for a project $\mathcal{P}_i$ are $t(\mathcal{E}'_i)$ and $t(\mathcal{E}''_i)$ respectively, $\mathcal{E}'_i \succ \mathcal{E}''_i$ if and only if $\phi_i(t(\mathcal{E}'_i) - G_{i1}) < \phi_i(t(\mathcal{E}''_i) - G_{i1})$, i.e., project schedule $\mathcal{E}'_i$ is preferable to $\mathcal{E}''_i$. Many different forms of deviation cost functions can be designed. In general, project scheduling models take into account the following three criteria: (1) earliness, (2) tardiness and (3) risk of failure. These three can be easily incorporated in a deviation cost function, as shown in Figure 2.2 for example.

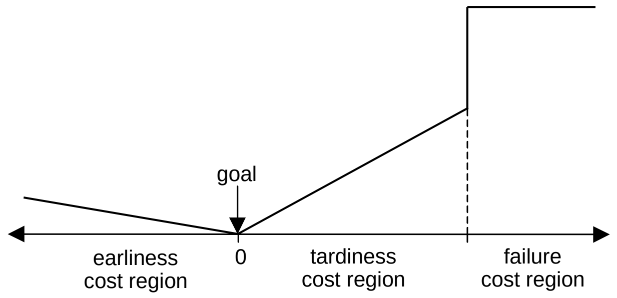


Fig. 2.2. A deviation cost function

As mentioned in Section 1.2.2, the relative importance or urgency of each project is determined and given to each project by the top management, and this weight distribution must be consistently applied to the resource control process. Following is the formal definition of a *project weight* and the *project weight distribution*.

Definition 2.5. (project weight distribution) A *project weight*, w_i, of a project $\mathcal{P}_i$ defines the relative importance or urgency of the project. So, for a DMP resource control problem, the relative weights of the projects are given in the form of *project weight distribution*, $\mathcal{W} = \{w_i\}$, where $\sum_i w_i = 1$. □

2.1.3 Project schedules

Before defining *project schedules*, we need to define *control window*, which is a time-window in which the resources are to be rescheduled when significant changes happen in the DMP environment. As explained at the beginning of this chapter, the resource time is divided into equilength time slots for scheduling purposes; hence, the control window is also defined by time slots as follows:

Definition 2.6. (control window) A *control window*, $h = [h_1, h_2]$, of a DMP resource control problem is a set of consecutive time slots between two time slots h_1 and h_2 inclusively, where $h_1 \leq h_2$. □

The following three definitions define an *engagement*, a *project schedule* and a *DMP schedule*, which are the outcomes of the resource control mechanism.

Definition 2.7. (engagement) An *engagement*, $\mathbf{e}_{il} = (j, k, \Omega)$ is a commitment of a resource R_{jk} to allocate its resource-time period Ω to the task T_{il} in $\mathcal{P}_i$, where $\Omega = [\omega_1, \omega_2]$ defines the time window between the time-slots ω_1 and ω_2, over which the task is to be allocated. □

Definition 2.8. (project schedule) A *project schedule* of $\mathcal{P}_i$, denoted by $\mathcal{E}_i$, is a set of engagements that cover all the tasks in $\mathcal{P}_i$, i.e., $\mathcal{E}_i = \{\mathbf{e}_{i1}, \mathbf{e}_{i2}, ..., \mathbf{e}_{i|\mathcal{T}_i|}\}$, and *precedence-conflict free*, i.e., $\forall (T_{ij},\ T_{ik}) \in \mathcal{N}_i, \omega_{j_2} < \omega_{k_1}$ where $\mathbf{e}_{ij} = (*, *, \Omega_j)$, $\mathbf{e}_{ik} = (*, *, \Omega_k)$, $\Omega_j = [\omega_{j_1}, \omega_{j_2}]$ and $\Omega_j = [\omega_{k_1}, \omega_{k_2}]$. □

Definition 2.9. (DMP schedule) A *DMP schedule*, denoted by $\mathcal{A} = \langle \mathcal{E}, h \rangle$, is a set of project schedules within the DMP control window, i.e., $\mathcal{E} = \{\mathcal{E}_i\}$, $h = [h_1, h_2]$, and *resource-conflict free*, i.e., $\forall\ (k, l, \Omega_i),\ (k, l, \Omega_j) \in \mathcal{E}$ where $i \neq j$, $1 \leq l \leq |\mathcal{R}_j|$, $1 \leq k \leq |\mathcal{D}|$, $\Omega_i = [\omega_{i_1}, \omega_{i_2}] \subseteq h$ and $\Omega_j = [\omega_{j_1}, \omega_{j_2}] \subseteq h$, $\omega_{i_2} < \omega_{j_1}$ or $\omega_{i_1} > \omega_{j_2}$. □

2.1.4 DMP Resource Control

Now we define the *DMP resource control* problem, which is the problem we try to solve. As mentioned in Section 1.2, the *planning* procedure is carried out every time a new project starts; *control* is a persistent procedure for maintaining the projects as planned. As just defined, a valid DMP schedule is a set of resource and precedence conflict-free engagements over a given control window. Associating this concept of *DMP schedule*, we define the DMP resource control problem as follows:

Definition 2.10. (DMP resource control) Given DMP environment Π, control window h, and project weight distribution $\mathcal{W}$, *DMP resource control* problem Ψ is a persistent procedure to maintain a DMP schedule, $\mathcal{A}$, *valid*[1] all the time, i.e., $\Psi : (\Pi, h, \mathcal{W}) \rightarrow \mathcal{A}$, where $\mathcal{A} = \langle \mathcal{E}, h \rangle$ is a valid DMP schedule. □

2.2 Integer Programming Formulation

There have been many different Integer Programming (IP) models for project scheduling problems. We formulate the multiple project scheduling problem for minimizing weighted tardiness based on Drexl (1991). We assume that the deviation cost function is a simple linear function, $\phi(t) = t$. As shown in the following formulation, the number of variables increase rapidly according to the increase of scheduling horizon, the number of tasks, and the number of resources. This formulation is basically a centralized formulation, and it cannot be directly applied to DMP resource control problems. Instead this formulation is used for comparison purposes, especially to compare the P-TÂTO[2]-generated DMP schedule to an optimal schedule when the problem is solved in a centralized way. In the IP formulation, constraint (2.1) ensures that any task is assigned only once, (2.2) ensures precedence relations, and (2.3) ensures no resource is allocated to more than one task at a time. Table 2.1 summarizes the notation for the IP formulation.

[1]Validity of a DMP schedule means satisfaction of both precedence and resource constraints.

[2]P-TÂTO mechanism is our proposed market-based mechanism explained in Chapter 4.

$$x_{i,j,t} = \begin{cases} 1 & : \text{ task } j \text{ of project } i \text{ is finished by resource } r_{ij} \text{ in time slot } t \\ 0 & : \text{ otherwise} \end{cases}$$

$$\textit{minimize} \qquad \sum_{i=1}^{I} W_i \sum_{t=1}^{T} t \cdot x_{iJ_it}$$

$$\textit{subject to}$$

$$\sum_{t=\underline{w}_{ij}+\delta_{ij}-1}^{\overline{w}_{ij}} x_{ijt} = 1 \quad \forall i, j \tag{2.1}$$

$$\sum_{t=\underline{w}_{ih}+\delta_{ih}-1}^{\overline{w}_{ih}} t \cdot x_{iht} \leq \sum_{t=\underline{w}_{ij}+\delta_{ij}-1}^{\overline{w}_{ij}} (t-\delta_{ij}) \cdot x_{ijt} \quad \forall i, j \quad , \quad h \in P_{ij} \tag{2.2}$$

$$\sum_{i,j:r_{ij}=k}^{J_i} \sum_{q=t}^{t+\delta_{ij}-1} x_{ijq} \leq 1 \quad \forall k, t \tag{2.3}$$

Table 2.1. Notations for the IP formulation

notation	description
I	total number of projects
J_i	number of tasks in i-th project
T_{ij}	j-th of i-th project, $i = 1, \ldots, I$ and $j = 1, \ldots, J_i$
δ_{ij}	time duration of T_{ij}
r_{ij}	resource index for task T_{ij}
K	number of resources
P_{ij}	set of the predecessors of T_{ij}
T	time horizon, $t \in 1, \ldots T$
W_i	priority (weight) of project i
$\underline{w}_{ij}$	earliest start time (ES) of task T_{ij}
$\overline{w}_{ij}$	latest finish time (LF) of task T_{ij}

2.3 Solution Quality of a DMP Schedule

In this chapter we formally defined a DMP schedule and the DMP resource control problem. As seen in the Definition 2.9, a DMP schedule is defined in the level of feasibility (both in resource and precedence). When developing a DMP resource control mechanism, we need a definition of the DMP schedule *quality* (i.e., the definition of “good” DMP schedule), so that we can evaluate the DMP resource control system with respect to the quality of the output DMP schedules. The following are solution quality measures which DMP resource control mechanisms try to achieve:

- **Efficiency**: For each resource, the resource schedule is efficient if the engagements cannot be updated in a way that any of the task utility can be increased without decrease of any other task's utility. If all the resource schedule is efficient, the DMP schedule is also efficient.

- **Optimality**: For each resource, the resource schedule is optimal if the engagements maximize the aggregate utility values of the corresponding tasks. If all the resource schedules in the DMP schedule (which is already feasible by definition) is optimal, then the DMP schedule is said optimal[3]. If a local resource schedule or a DMP schedule is optimal, they are efficient too.

- **Reflectiveness**: If the DMP schedule more clearly reflects the project weight distribution, the DMP schedule is more desirable. This property is directly related to the *controllability* (see Section 6.5) of a mechanism.

[3]This is corresponding the equilibrium state of the DMP economy explained in Section 4.1.1

Chapter 3

Related Work

In this chapter, we review background literature related to the DMP resource control problem. As explained, the DMP resource control problem is a short-term scheduling problem. Instead of discussing the scheduling literature, however, we mainly discuss distributed resource allocation literature. General notion of resource allocation denotes a *one-shot* resource distribution to multiple agents (, processes, or tasks). However, we map the DMP resource control problem to a *distributed resource allocation* problem by converting the continuous resource time to discrete time slots. By allocating discrete commodities (resource time slots in our case), the scheduling problem is solved. In Section 3.1 a general notion of market-based distributed resource allocation literature is briefly reviewed, and they are elaborated with respect to the market configuration-based classification in Section 3.2. In Sections 3.3 and 3.4, we discuss some problems which have closely related-problem domains and methodologies.

3.1 Market-based Distributed Resource Allocation

Market-based automated negotiation, or more commonly *market-based control*, is a paradigm for controlling complex system that would otherwise be very difficult to handle, by taking advantage of some desirable features of a market (especially a free market) including decentralization, interacting agents, and some notion of resources that need to be allocated (Clearwater, 1996). This approach has been applied to a wide range of fields such as dynamic computer resource allocation (Ferguson et al., 1988; Waldspurger et al., 1992), supply chain management (Hinkkanen et al., 1997; Sauter et al., 1999), vehicle routing (Sandholm, 1993), workflow management (Jennings and Vulkan, 2000), manufacturing scheduling (Tilley, 1996; Baker, 1996)(Kutanoglu and Wu, 1997; Walsh et al., 1998), project planning and control (Qian, 1998; Lee and Kumara, 2000), and process control (Jose and Ungar, 1998). Depending on the situation that the problems deal with, solution methodologies can be classified as shown in Figure 3.1.

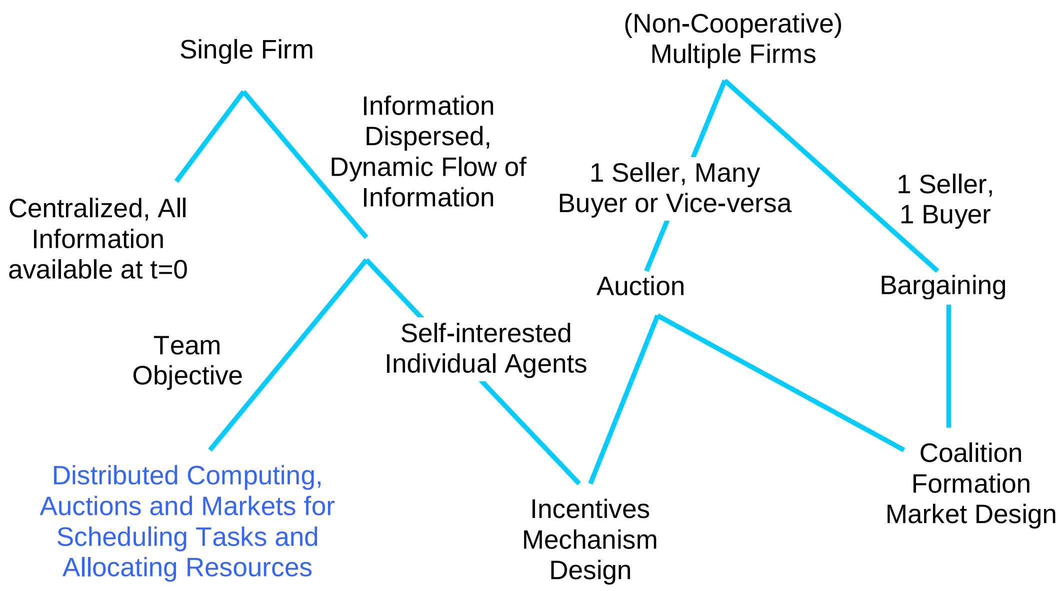

Single Firm
Centralized, All Information available at t=0
Information Dispersed, Dynamic Flow of Information
Team Objective
Self-interested Individual Agents
Distributed Computing, Auctions and Markets for Scheduling Tasks and Allocating Resources
(Non-Cooperative) Multiple Firms
1 Seller, Many Buyer or Vice-versa
1 Seller, 1 Buyer
Auction
Bargaining
Incentives Mechanism Design
Coalition Formation Market Design

Fig. 3.1. Taxonomy of resource allocation mechanisms (Chatterjee, 2002).

The DMP resource control problem domain and our proposed mechanism (explained in Chapter 4) can be classified into the case of distributed resource allocation having a team objective in Figure 3.1. However, our virtual market model and multiagent-based information infrastructure (explained in Chapters 4 and 5) can incorporate the cases requiring an incentive mechanism design.

Fundamental notions of *resource allocation* and *scheduling* are different in the sense that the scheduling problem has to handle an extra dimension - *time*- while resource allocation usually deals with a one-shot resource distribution. However, scheduling problem can be transformed into a one-shot resource allocation problem by converting the continuous time into countable discrete time-slots, which are allocated to the processes or tasks. In this case, we can utilize a variety of study (especially in economics literature) on distributed resource allocation in order to solve distributed scheduling problems, including the DMP resource control problem. Also by defining the time-slots as commodities that are bought and sold, the scheduling problem can be converted into a market-based resource allocation problem. However, none of the distribute resource allocation mechanisms in economics literature can be directly applied to the DMP resource control problem, mainly because of the DMP problem's tasks structure - specifically, precedence constraints among tasks.

3.2 Market Configuration

Market-based control is a *metaphoric* approach, meaning that the original problem is mapped into a *virtual* market place where buyers and sellers trade commodities until an equilibrium state is reached. Hence, we can classify market-based problem-solving models with respect to the structures of (or internal relationship among) the market components: task structure, commodity, and market organization.

3.2.1 Task Structure: Hierarchical vs. Procedural

In a distributed resource allocation problem, the set of tasks (which requires resources) have some structure in general. Typical task structures are *hierarchical* and *procedural* structures. SPAWN (Waldspurger et al., 1992) is one of the earliest market-based control systems for dynamic resource allocation in a network of distributed computer workstations. This system assumes a *hierarchical task structure* of tasks. The concept of *funding* works well with such a task structure to control the priority among tasks. Task allocation problem addressed in (Walsh and Wellman, 1998) has also a hierarchical task structure; however, many other practical problems have *procedural task structure.* Manufacturing scheduling/control (Tilley, 1996; Baker, 1996; Kutanoglu and Wu, 1997; Walsh et al., 1998) or workflow coordination problems (Jennings and Vulkan, 2000) usually have simpler, having fewer operations and precedence relations, while a project scheduling/control problem (Qian, 1998; Lee and Kumara, 2000) has a complicated large activity network so that more coordination effort is required.

3.2.2 Commodity: Bid for Tasks vs. Bid for Resources

Many *task allocation* problems assume no resource boundedness, or do not concern themselves with resource utilization. Instead they are more concerned about outputs or service quality. Many workflow coordination problems and military logistics problems fall into this category. On the other hand many resource allocation problems are involved in bounded (or scarce) resources. In some cases, the resources (or resource organizations) have their own interest, for example their own utilization maximization. Such resource boundedness is a major criterion for one of the most fundamental market mechanism-design decisions: *bid for task* or *bid for resource.* In a more resource-bounded situation, bid-for-resource is more a natural approach than bid-for-task. For example, the market-based contract nets (see Section 3.3.1) by Baker (1996) or Tilley (1996) are the typical cases of bid-for-task approach, and the market-based distributed scheduling (see Section 3.3.2) by Walsh et al. (1998) is of bid-for-resource approach.

3.2.3 Participants and their Organization

The individual units of a distributed organization (including a DMP environment) make decisions using local information and knowledge due to information barriers and boundedness, which are caused by their generic organizational structure. So depending on the structure of the mixture of private information and common information, quite different solution mechanisms can be designed. The private information in a competitive situation is not revealed due to strategic reasons, while local information is held locally for saving communication burdens and expediting decision making processes. Information boundedness directly points to information exchange between the participants (i.e., agents). On one hand, all information (e.g., about resources needed by a participant) could be posted to a common bulletin board visible to all participants. On the other extreme, information is exchanged purely between pairs of participants. Tan and Harker (1999) analyzed workflow efficiency of coordination problems by comparing the two organizational structures: hierarchical organization and (market-based) decentralized organization, and proposed some decision rules for adapting market-based approach.

3.3 Market-based Task / Resource Scheduling

As mentioned previously, market-based control has been applied to a wide range of fields such as computer resource allocation, network bandwidth control, power grid control, and shop-floor or project scheduling. In this section, we review some of literature in manufacturing and project scheduling area, because of the similarity between these problems and the DMP resource control problem in problem domain and methodology.

3.3.1 Market-based Contract Net

Baker (1996) and Tilley (1996) showed that market-based control is very applicable to real-time factory scheduling problem in a *heterachical* situation. In their models, as shown in Figure 3.2, each agent controls one or more manufacturing resources such as machines, material handling systems, inventory storage, and manual operations. Responding to the arrivals of new orders, the machine agents recursively bid for the remaining tasks in a quite straightforward way. Hence their research can be seen as a market-based extension of *contract net*[1]. The limitation of their problem formulations is that they did not seriously take into account the fundamental scheduling constraints

[1]Baker actually called his proposed architecture *market-driven contract net* and Tilly called *manufacturing contract net* (MCN).

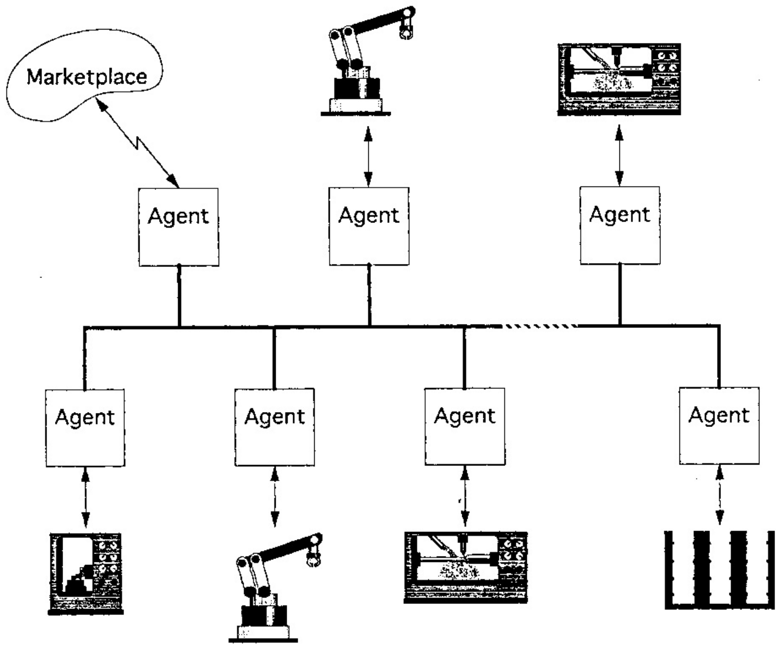


Fig. 3.2. The market-driven contract net manufacturing computer-control architecture (Baker, 1996).

such as strict due dates or resource constraints. In order to incorporate such resource constraints in the market model, it is natural to model the resource as a good, which is bought/sold. They, however, use tasks rather than resource time or machine time as goods. In this sense, the following research may be more appropriate for the resource allocation problem.

3.3.2 Market-based Distributed Scheduling

Walsh et al. (1998) modeled the factory scheduling problem as a discrete allocation problem by seeing the resource's time slots as discrete resources[2]. In their *factory scheduling economy*, the time slots of a factory (as a single machine) are bought by and sold to the agents, who need the factory for a given period of time, through a simple auction mechanisms as shown in Figure 3.3. Their theoretical analysis of auction mechanisms give good theoretical insights. They investigated aspects of two auction

[2]This is very common scheduling formulation approach in operations research. It is usually formulated as an integer program as explained in Section 2.2.

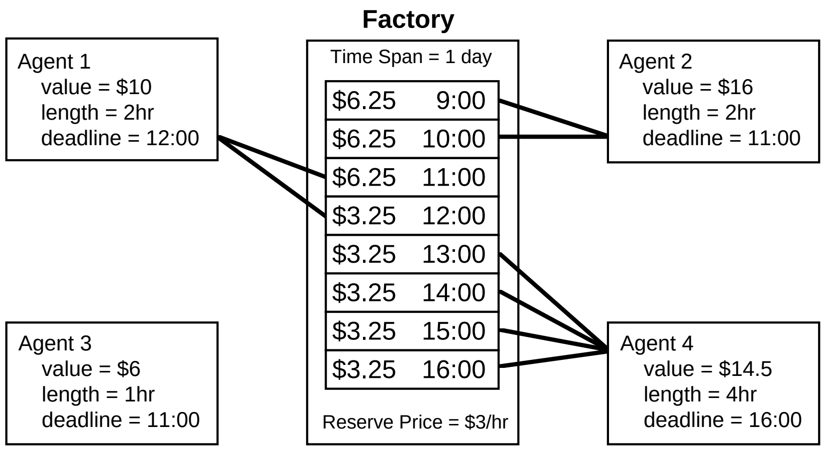


Fig. 3.3. A factory scheduling economy. Lines connecting the agents to the time slots represent one feasible allocation(Walsh et al., 1998).

mechanisms for the factory scheduling economy - a simple ascending auction and a *generalized Vickrey auction*(GVA) (MacKie-Mason and Varian, 1994), for the single unit allocation problem and multiple unit allocation problem, respectively.

3.3.3 Market-based Project Scheduling

Qian (1998) suggested the possibility of market-based software development project scheduling. In the project scheduling economy, he introduced two types of goods: (i) *resource time slots*, which represent the employee's working time just like Walsh et al. (1998) did, and (ii) *project time slots*, which represent the precedence relation between consecutive tasks. The task agents bid for the both types of time slots, and the price of each slot is adjusted every bidding iteration based on the number of bids to the slot. The choice of goods in this research might be still applicable to DMP planning and control problem, but this market-based model has a few critical limitations, which are applicable to the DMP case as follows:

1. The pricing system did not incorporate each project groups' interest; instead, a simple price adjustment mechanism, which is directly proportional to the number of bids in the slot, was used in the algorithm. His model is enough to generate a feasible

solution, but it is not guaranteed to achieve good solution quality in the sense of tardiness, earliness, or risk management.

2. His model was a centralized static scheduling formulation for a traditional deterministic project network, meaning that although the economic metaphor was used in his algorithm, it was more like a heuristic search algorithm to find a feasible solution rather than a market-based coordination or negotiation mechanism, by which global objectives are achieved as an emergent property.

3. The computational performance - both of computation time and solution quality - was too poor to be applied to any practical size problems. The high computational complexity basically arises from the combinatorial property of the multiple unit resource allocation problem and inefficiency of the price adjustment process. The results of the simulation-based computational experiment showed this algorithm may not work in any practical size problems.

3.4 Other Related Work

3.4.1 Decomposition Methods

Resource constrained project scheduling problems are a generalization of the $\mathcal{NP}$-hard job-shop scheduling problems, where the computational requirements for obtaining an optimal solution grow exponentially as the problem size increases. Hence, major efforts to solve this problem in *operations research* (OR) have concentrated on developing heuristic procedures to obtain *near-optimal* solutions. For some specific types of project scheduling problems, efficient solution procedures can be designated by taking full advantage of their desirable features. Luh et al. (1999) and Liu et al. (1998) formulated the scheduling problem of multiple distributed design projects as an *integer programming* (IP) model, and proposed an efficient solution procedure based on *Lagrangian relaxation* (LR) along with a stochastic dynamic programming method and some heuristics. They assume that the subprojects are de-coupled in the *process-centered project* configuration. This assumption makes the problem formulation *separable* so that LR can be successfully applied; however, this assumption cannot be made in the DMP resource control problem.

3.4.2 DAI/MAS for Distributed Project Planning and Control

The distributed Artificial Intelligence (DAI) community has also addressed the distributed project planning and control problem. Chang et al. (1993) modeled the distributed project planning problem as a set of distributed *assumption-based truth maintenance system* (ATMS). Although each ATMS, as a *decision support system (DSS)*, helps a distributed planner to maintain the local plan consistently, conflict resolution among the participants' plans still is based on face-to-face negotiations. Petriea et al. (1999) investigated the project management problem from the viewpoint of *process coordination* and pointed out the importance of *change propagation* in distributed project environment. His research is concentrated on modeling the dependency among the elementary project activities rather than resource planning and control. So, this approach will be a complimentary part of DMP resource control research for supporting overall DMP management. Drabble (1995) pointed out that the desirable framework for intelligent project planning and control tool in *wide area project management* problem would be based on a *multiagent framework*.

3.4.3 MAS for Design Projects

Although MAS-based project planning and control research can hardly be found, MAS applications for design process coordination have been frequently reported. *Multiagent design system* (MADS) is a design applications that incorporates multiple software agents (Lander, 1997). Some researchers demonstrated that MADS can be successfully applied in the large-scale, distributed design environments by integrating human engineers and a set of application software using multiagent framework (Cutkosky et al., 1993; Frost and Cutkosky, 1996; Lander and Corkill, 1996; Jin and Lu, 1998). Although these studies tried to solve design problems, there is some similarity between their approaches and our DMP resource control mechanism because of their main theoretical thrust on coordination mechanisms or dynamic conflict resolution. Representative approaches in this area include: (i) narrowing search space (Parunak, 1999), (ii) modeling design space as a market (Wellman, 1995; Parunak, 1999), (iii) tracking constraints to ensure *Pareto* *optimality* (Petrie et al., 1995) and (iv) sharing information through explicit common storage such as *blackboard* (Lander and Corkill, 1996; Tan et al., 1996).

We have explained the background, problem definition and related literature in the first three chapters. In the following three chapters, we discuss the details of the proposed market-based control mechanism called P-TÂTO (Chapter 4), the multiagent-based information system design (Chapter 5), and experimental analysis results (Chapter 6).

Chapter 4

Market-based DMP Resource Control

In a DMP environment, each project group and resource division are to some extent self-interested, and there are information barriers among project groups and among resource divisions. Project groups compete for shared resources (e.g., resource time slots) while trying to minimize the *estimated deviation cost* by re-scheduling their tasks within the current *control window*, say $[t+1, t+n]$ where t is the current time slot index and n is the control window size. On the other hand, each resource tries to maximize its *resource utilization*, which is a sum of the utilities for allocated tasks within the control window. There are two types of constraints that restrict the objective of each participant: *precedence constraints*[1] and *resource constraints*[2]. In this chapter, a market-based negotiation mechanism called P-TÂTO is presented, which solves the resource constraints in an optimal way and searches for a precedence conflict-free solution throughout a tâtonement type procedure. In order to elaborate the proposed approach, the computational economy of the DMP resource control environment is first defined.

[1] Precedence constraints are defined by project-wise project network

[2] Resource constraint means that any resource time slot cannot be allocated to more than two tasks

4.1 Mechanism Design Overview

4.1.1 Structure of Computational Economy

The DMP resource control environment is modeled as a computational economy, where multiple autonomous software agents buy and sell resource time slots. The overall DMP resource control economy is a *dynamic economy* in the sense that multiple markets are dynamically established and cleared over time. This dynamic economy, as a whole, is called a *DMP economy*. As shown in Figure 4.1, the DMP economy e is an infinite sequence of *temporary economies*, e_t's, each of which is a set of uncleared resource time-slot markets at a specific time t. We call this individual time-slot market *local market*, denoted by m_j, which corresponds to the resource R_j.

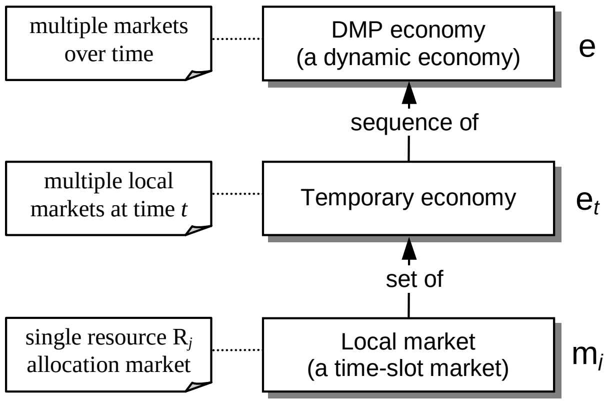

multiple markets over time
DMP economy (a dynamic economy)
e
sequence of
multiple local markets at time t
Temporary economy
e_t
set of
single resource R_j allocation market
Local market (a time-slot market)
m_i

Fig. 4.1. The structure of a DMP resource control economy

If every market in an economy has been cleared at time t, the economy is *stable* (or in an *equilibrium state*) at time t. The DMP resource control problem manages the stability within the control window, or the stability of a temporary economy. As time progresses, either precedence or resource conflicts are detected within the control window at time t, comprising multiple coupled local markets m_j's, where each of m_j is to sell the time slots of the R_j within the control window.

4.1.2 Mechanism Design Strategy

As explained, the DMP economy is just a sequence of temporary economies. Hence our major mechanism-design concern is an effective design of a temporary economy establishing and clearing mechanism. A temporary economy consists of multiple inter-dependent (by precedence and resource constraints) local markets.

1. At the temporary economy level, an overall market clearing mechanism needs to be established. We propose an iterative market mechanism called *Precedence Cost Tâtonnement* (P-TÂTO) mechanism. One of core parts of this mechanism is local market evaluation mechanism.

2. For the local market evaluation, a combinatorial auction mechanism that allocates the resource time slots to tasks in an optimal way with respect to a welfare function (see Section 4.5) needs to be implemented.

In such a computational economy, a local market clearing mechanism guarantees the feasibility and optimality of the resource allocation of the local market, but it does not guarantee the precedence feasibility between the allocated tasks. Local markets hence need to be evaluated repeatedly without physical market clearing until the temporary economy level coordination achieves precedence feasibility. Our market-based resource control approach can be summarized as follows. The details of P-TÂTO mechanism are explained in Section 4.3.1.

- **sellers**: resource agents (RA), each of which is in charge of a resource.
- **buyers**: task agents (TA), each of which is in charge of a task.
- **commodity**: discrete time slots
- **resolving resource constraints**: combinatorial (bundle) auction mechanism
- **resolving precedence constraints**: price tâtonnement process on *precedence cost vector* of each tasks (P-TÂTO).
- **bidding**: bid for resources using combinatorial bids

4.1.3 Precedence Tâtonnement vs. Resource Tâtonnement

The one of the core ideas of our mechanism is precedence cost tâtonnement at the temporary economy level along with resource-wise optimization in the local economy level. Figure 4.2 shows conceptual differences between precedence cost tâtonnement and resource price tâtonnement (we call them *precedence tâtonnement* and *resource tâtonnement*[3], respectively in short) when they are applied to DMP resource control problems.

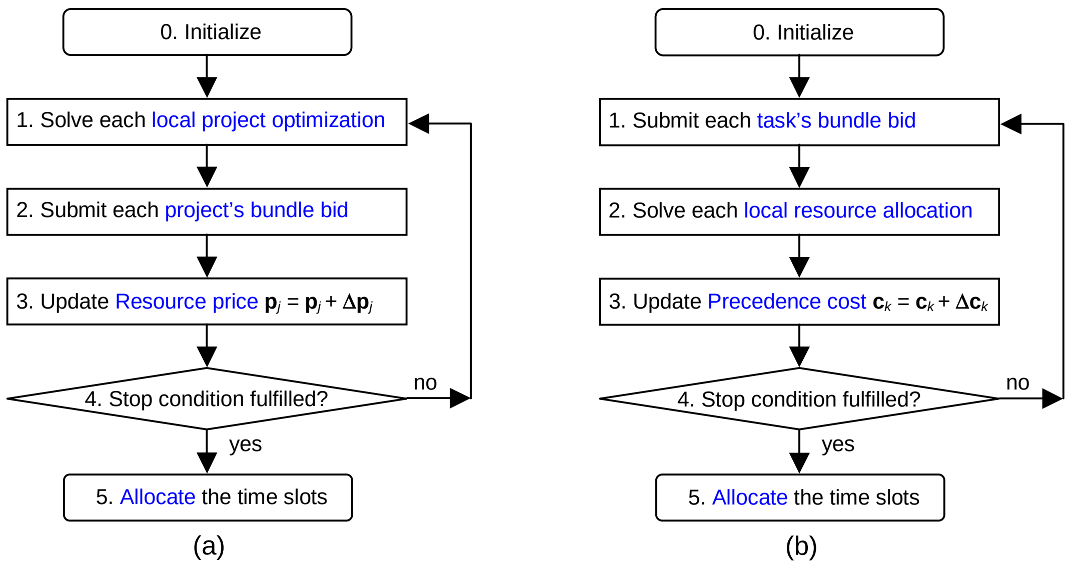

0. Initialize
1. Solve each local project optimization
2. Submit each project's bundle bid
3. Update Resource price $\mathbf{p}_j = \mathbf{p}_j + \Delta\mathbf{p}_j$
4. Stop condition fulfilled?
no
yes
5. Allocate the time slots
(a)
0. Initialize
1. Submit each task's bundle bid
2. Solve each local resource allocation
3. Update Precedence cost $\mathbf{c}_k = \mathbf{c}_k + \Delta\mathbf{c}_k$
4. Stop condition fulfilled?
no
yes
5. Allocate the time slots
(b)

Fig. 4.2. Comparison of overall procedures of (a) resource tâtonnement and (b) precedence tâtonement

The rationale of using the precedence tâtonnement approach can be explained as follows:

- *The numbers of projects and resources*: In DMP environments, the number of shared resources are a lot greater than the number of projects. That means searching

[3]Most of market-based resource allocation approaches including Kutanoglu and Wu (1997)'s falls into this category

for resource-feasible allocation for all resources is a lot harder than searching for precedence-feasible allocation for all projects.

- *Commitment window*[4]: Commitment windows compress the search space for the precedence conflict resolution, meaning that precedence tâtonnement process converges faster than resource tâtonnement.

- *Scarce resources*: Compared to many task allocation problems, resource allocation problems are defined in the situation of scarce resources, meaning that resource time slot domains are crowded with tasks so that the feasible resource allocation could not easily obtained through an iterative search process due to the tight coupling.

- *Local optimization burden*: We can generate much more efficient local resource scheduling algorithm compared to the local project scheduling problem, because a very good initial resource schedule, which is critical for the local optimal allocation heuristics, can be easily generated based on the critical path analysis. An efficient local market clearing algorithm is proposed, which guarantees high level of optimality (see Section 4.5 for details).

- *Resource utilization maximization*: In the resource tâtonnement mechanism, the resource divisions's self-interestedness cannot be realized, while the precedence tâtonnement mechanism maximizes both of the resource utilization and the project groups' utilities.

- *Information barrier*: In a highly distributed (or decentralized) environment, task agents have only bounded local information, such as information on *neighbor* task agents. This means that project groups cannot solve the project-wise optimization problems, which is a core part of a resource tâtonnement approach.

In Section 4.2, the proposed multiagent architecture for implementing the computational economy model in a conceptual level is presented. In Section 4.3 the details

[4] *Commitment window* is a time window where a task can be assigned, which can be defined by two time slots [ES,LF], where ES denotes *earliest start time* and LF denotes *latest finish time.* This window can be generated based on the *critical path analysis* on the project network in the project planning phase.

on the market mechanism (P-TÂTO) is presented. In Section 4.4 we describe the issues on how each TA generates its bid, namely on the individual TA's utility on time slot bundles. In Section 4.5 we present the local market-evaluation mechanism, which is a core element of the whole P-TÂTO mechanism.

4.2 Multiagent Architecture

Cognitive limitations and resource boundedness of individual agents naturally call for *agent organization*, thereby, forming multiagent system (MAS) (Carley and Gasser, 1999). In early MASs, individual agents were designed to have the same social abilities. In this case, however, it may be expensive for each agent to duplicate so-called *matchmaking* activities - an agent's activities for finding appropriate agents with which to interact. In order to overcome this problem, *federated agent architecture*[5] was designated to reducing the matchmaking burdens on task-specific agents. Corkill and Lander (1998), however, pointed out that more specialization over the simple federated architecture is required for the long-lasting and large-scale agent-based systems[6]. According to their argument, the DMP resource control system requires an organizational structure. We discuss the agent organization in this section.

4.2.1 The Agents

The core issue in a multiagent organizational design problem is to define the individual agent *roles* within the organization, which is obviously dependent on the agent *encapsulation* decision, namely, the decision on what represent agents. Many different encapsulation approaches can be considered in different situations, but most of them fall into two major categories: (i) *function-oriented approaches*, where agents are used to encapsulate some functions such as order acquisition, planning, material handling

[5]A relatively simple MAS architecture where special agents, called matchmaker, broker, or facilitator, handle the agent matching problem.

[6]In the paper, they listed the cases where the organizational design is more important, including the agent systems with (i) larger number of agents, (ii) longer duration of agent activities, (iii) more repetitive activities, (iv) more activities requiring shared resources, (v) more collaborative activities, (vi) more specialized agents, (vii) less capable agents and (viii) less slack resources available.

and product distribution; (ii) *physical entity-oriented approaches*, where agents represent physical entities such as managers, workers, machines and components (Shen and Norrie, 1999). The second approach naturally defines distinct sets of state variables that can be managed efficiently by individual agents with limited interactions. Therefore, the second approach is more appropriate for modeling manufacturing environments, where more physical entities are involved compared to transaction-oriented information system domains. The first approach, however, is still useful for specific services even if the overall MAS design follows the second approach.

Besides the general notion of encapsulation, we need to take into account a DMP specific requirement when designing an agent organization model. The project resource control processes require, in general, more strict application of responsibility and authority for every single task. It means that the DMP planning and control system cannot be a fully autonomous system that does not require any human intervention. Instead, the agent organization simulates the physical DMP organization to some extent, in order to incorporate the responsibility and authority structure in physical organizations. On the basis of these arguments, five main agent classes are defined: project manager (PM), task agent (TA), resource manager (RM), resource agent (RA) and a market coordinator (CO). The following are the fundamental roles of these agents and the key information entities they maintain.

1. *Project Manager (PM)*: a PM class agent is in charge of successful accomplishment of a project by coordinating the individual task agents, defined below. Each PM agent maintains the project's milestones ($\mathcal{G}_i$), project network like PERT/CPM chart ($< \mathcal{T}_i, \mathcal{N}_i >$), and each task's resource allocation information (or project plan $\mathcal{A}_i$).

2. *Task Agent (TA)*: a TA class agent is in charge of its own single task (T_{ik}). Each TA maintains the information on resource allocation of the task such as required resource types, task duration, current commitment window and the current schedule.

3. *Resource Manager (RM)*: an RM class agent is in charge of coordinating a set of resource agents, and interacts with other RMs and the CO for temporary economy

initiation and clearing. Each RM maintains the information on resource ($\mathcal{R}_j$) including capability types and current schedules.

4. *Resource Agent (RA)*: an RA class agent is in charge of a single resource such as a machine, a worker or a tier of computer software. An RA, as a seller interacts with TAs (as buyers) in a local market. Each RA maintains its own schedule.

5. *Market Coordinator Agent (CO)*: The CO is in charge of coordinating multiple local markets in the computational economy. The CO is a persistent agent, while other agents are dismissed from service after their goals are achieved.

The first four agent types embody physical DMP organizations approximately, while the CO agent is a virtual entity for market mechanism's purposes.

4.2.2 Organization of Markets

According to the role definitions of each agent class, an MAS implements a computational economy. Figure 4.3 shows an example of the MAS organization consisting of the different types of agents. In the virtual marketplace shown in the figure temporary economies (each of which is a set of multiple simultaneous local markets) are established and cleared over time.

A local market is established for every resource within the control window, meaning that every RA will be an auctioneer (and seller) for selling the bundles of resource time slots to TAs - the bidders (or buyers). Hence RAs and TAs conduct a key role in the market-based mechanism, while other types of agents support them. Figure 4.4 shows a simple example of a resource time slot market, where three TAs participate for time slots of a resource in `division-1`.

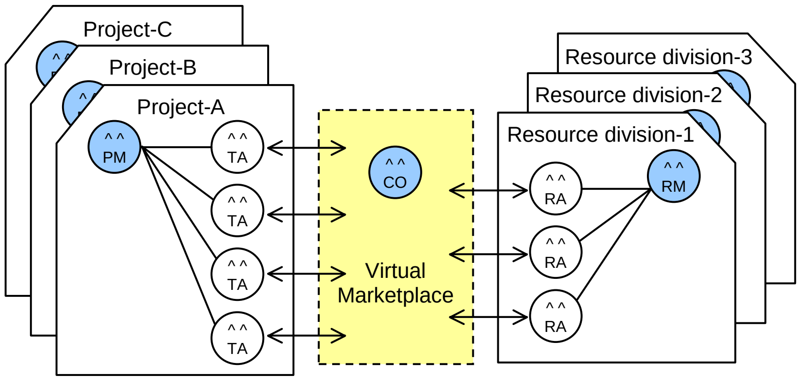


Fig. 4.3. A MAS organization: There are three projects (`Project-A` to `-C`) and three resource divisions (`Resource division-1` to `-3`). `Project-A` has four task agents, and resource `division-1` has three resource agents. The market coordinator is a persistent agent that coordinates the virtual marketplace.

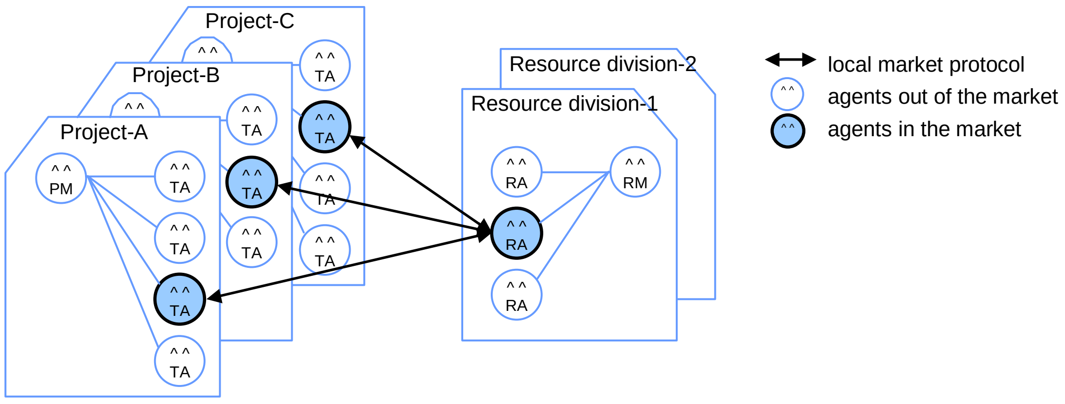


Fig. 4.4. A local market: An resource agent in a resource division (`division-1`) establishes a local market that consists of three task agents, which belong to each of three project groups.

4.3 Precedence Cost Tâtonnement (P-TÂTO) Mechanism

4.3.1 Overall Control Procedure

Figure 4.5 shows the overall control procedure of the P-TÂTO mechanism. As shown in the figure, this control loop is an infinitely repeating loop as DMP economy is defined over an infinite time domain. Each outer iteration (from Step 1 to Step 8) represents the establishment and clearance of a temporary economy, while each internal iteration (from Step 3 to Step 7) represents a cycle of local markets evaluation process. Following are brief explanations of each of the step in Figure 4.5.

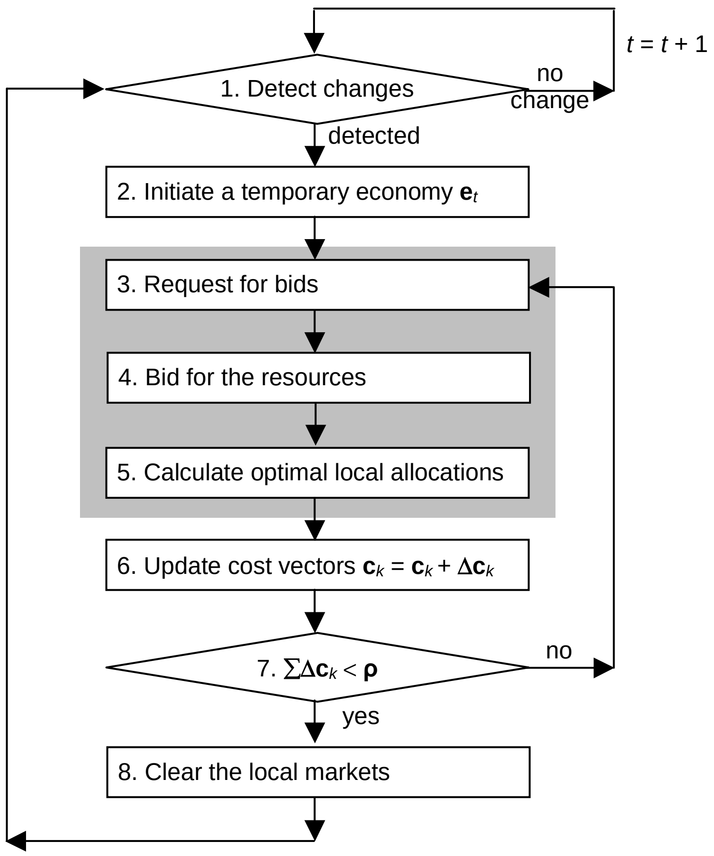

$t = t + 1$
1. Detect changes
no change
detected
2. Initiate a temporary economy $\mathbf{e}_t$
3. Request for bids
4. Bid for the resources
5. Calculate optimal local allocations
6. Update cost vectors $\mathbf{c}_k = \mathbf{c}_k + \Delta\mathbf{c}_k$
7. $\sum\Delta\mathbf{c}_k < \boldsymbol{\rho}$
no
yes
8. Clear the local markets

Fig. 4.5. Overall control procedure of P-TÂTO mechanism: The shaded steps (3 - 5) constitute the local market evaluation process, where multiple simultaneous local market evaluation processes are accomplished.

1. `Detect changes`: As time goes, the moving control window focuses the new set of tasks and their resource allocations. RAs detect any changes in task schedules, which can affect another schedule of tasks, generating conflicts in schedule.

2. `Initiate a temporary economy`: The detected changes are reported to the RM, and the RM informs this to other RMs, initiating a temporary economy. Every RM reports the CO of the information on RAs and TAs within the control window. The CO takes over the temporary economy.

3. `Request for bids`: The CO asks each RA in the temporary economy to establish a local market. Each RA who received the request message sends a *request for bid* (RFB) messages to the TAs whose tasks are currently allocated to the resource within the control window (see Section 4.3.2).

4. `Bid for the resources`: Each TA, M_k, generates a bid for resource time slot bundles. The set of bidding bundles are generated within the current commitment window. The utility $u_k(B)$ for a bundle, B, is calculated by combining the *estimated deviation cost* $\hat{\theta}_k(B)$ and the *aggregate precedence cost* $c_k(B)$ (see Section 4.4).

5. `Calculate optimal local allocations`: Each RA calculates the winning bids of the combinatorial auction. The resource allocation based on the winning bids is optimal with respect to the utility functions of the each bidder (TA). The RA informs the winning bidders of the new task schedule. The local markets are not cleared at this moment (see Section 4.3.2 and Section 4.5).

6. `Update precedence cost vector`: New task schedules are forwarded to TAs so that they can check precedence conflicts between tasks. Each TA updates the *precedence cost vector* $\mathbf{c}_k$ by adding penalties to the overlapped slots, and subtracting the cost from the under-loaded time slots (see Section 4.4.5 for details).

7. `Check the clearing condition`: If the sum of precedence cost changes ($\sum \Delta c_{kl}$) are less than a given tolerance ρ, the current schedule is feasible with respect to

precedence constraints and will not be changed in the subsequent iterations. Non-null vector $\Delta \mathbf{c}_k$ means that either the task k is still overlapped with other task(s) or the task has some room to be compressed.

8. `Clear the local markets`: If the clearing condition is satisfied, the CO announces the temporary market clearance, and accordingly local markets clearance. The RAs fix the schedule based on the current winning bids, and TAs also fix their schedules.

4.3.2 Local Market Evaluation

As shown in Figure 4.5, in every internal iteration local markets within a temporary economy are *evaluated*, meaning that the local markets are not *cleared* through this process. In addition, as mentioned earlier, the resource scheduling problem is transformed into the resource allocation problem by considering discrete time-slots as a resources to be allocated.

There are three major approaches for solving the market-based discrete resource allocation problems: (i) iterative price-adjustment mechanisms for generating *generalized equilibrium*[7] solutions, including Walras' original *tâtonnement algorithm*[8] (Walras, 1954) and its distributed implementation - WALRAS algorithm (Cheng and Wellman, 1998), (ii) simple ascending auction mechanisms for generating competitive equilibrium in an efficient and simple way (see (Walsh et al., 1998) and (Ausubel, 1997)), and (iii) combinatorial auction mechanisms for maximizing revenue for bundles of discrete resources. In a discrete resource time scheduling formulation, the value of each time slot cannot be evaluated independently each other. The first two approaches can not handle this *complementarity* between individual time slots[9]. Hence, the combinatorial auction

[7]A solution for a market where (i) supply meets demand, (ii) consumers maximize their preferences within their budget, and producers maximize profits within their production capabilities (Weiss, 1999).

[8]An iterative search algorithm for finding a general equilibrium. At every iteration, the auctioneer increases the price of goods that were over-demanded, and decrease the price of goods that were under-demanded (Weiss, 1999).

[9]We assume that the time slots in a *feasible* bundle must be consecutive with each other. However, this assumption might be relaxed in some practical situations by allowing *preemption*.

is the obvious choice for formulating the resource time slot market (i.e., a local market) in the DMP resource control problem.

Before presenting the detailed issues of the auction mechanism in the following sections, we formulate a local market as a special type of combinatorial auction. Suppose that we have m TAs, and n consecutive time slots to be allocated in a resource time slot market. Let $X = \{i : i = 1, ..., n\}$ be the set of time slots, where the time slots are ordered in chronological order. We allow combined bids for any set of consecutive time slots (*bundle*), $B_{i,j} = \{x \in X : i \leq x \leq j\}$, which can be interpreted as a time interval. Let $\mathcal{B}$ be the set of all possible bundles so that $\mathcal{B} = \{B_{i,j} : 1 \leq i \leq j \leq n\}$. The winning bundles in the auction must be disjointed because no time slot can be allocated twice. An *allocation* is any $\mathcal{S} \subset \mathcal{B}$ such that $B \cap B' = \emptyset$ for every $B, B' \in \mathcal{S}$.

Let us assume the goal of the RA is to maximize her/his revenue, for example, the sum of winning bids (more general argument on this issue will be presented in Section 4.5). Let $u_i(B)$ be the bid (exactly speaking, the utility of the bid) submitted by a task agent, M_i. Let $w(B) = \max_i\{v_i(B) : M_i \text{ bids for} B\}$. If no bid is submitted for B, set $w(B) = 0$. Using this notation, we can define the goal of the RA, finding an *optimal allocation* $\mathcal{S}^*$ with respect to bidder's utility, such as:

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} \sum_{B \in \mathcal{S}} w(B). \tag{4.1}$$

Example 4.1: (A Local Market: Auction Mechanism) Assume that there is a local market, where the resource agent $\texttt{RA}_\texttt{i}$ and two task agents $\texttt{TA}_\texttt{a}$ and $\texttt{TA}_\texttt{b}$, which are in charge of $\texttt{task}_\texttt{a}$ and $\texttt{task}_\texttt{b}$ respectively as shown in Figure 4.6. The control window $h = [1, 9]$, meaning that 9 time slots are to be sold in the market. The fixed duration time for $\texttt{task}_\texttt{a}$ and $\texttt{task}_\texttt{b}$ are 4 and 3, and the commitment windows are $[1, 5]$ and $[3, 7]$ respectively[10]. $\texttt{TA}_\texttt{a}$ and $\texttt{TA}_\texttt{b}$ send their bids as shown in the Figure 4.6,

[10]We can imagine the following situation. Before the failure of $\texttt{R}_\texttt{i}$, the $\texttt{task}_\texttt{a}$'s commitment window was $[-2, 5]$, and it was scheduled on $[-1, 2]$ for example. However, due to the resource R_i's failure, the commitment window was shrunk to $[1, 5]$

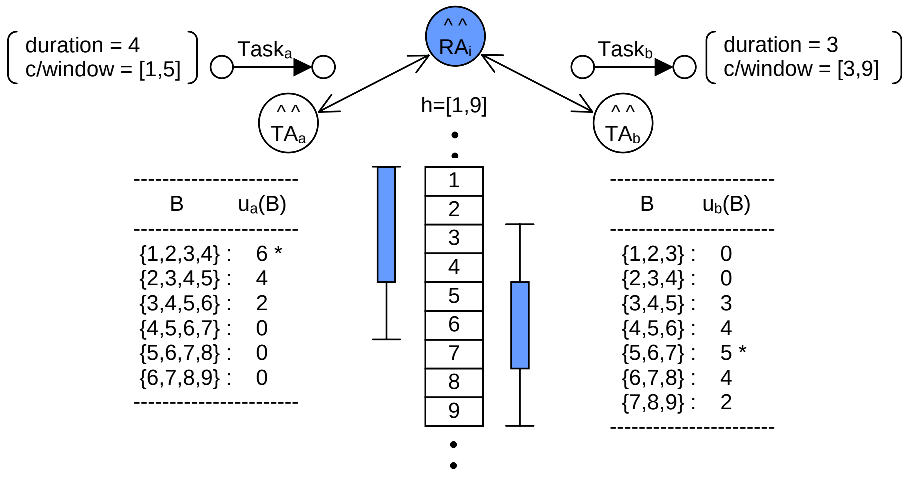


Fig. 4.6. An example of a local market: Two task agent $\mathtt{TA_a}$ and $\mathtt{TA_b}$ compete for the time slots $[1, 9]$ of resource $\mathtt{R_i}$, which is managed by the resource agent $\mathtt{RA_i}$.

and $\mathtt{RA_i}$ decides the optimal allocation using Equation (4.1). The optimal allocation is $\mathcal{S}^* = \{\{1, 2, 3, 4\}, \{5, 6, 7\}\}$ with the aggregate utility $w(\mathcal{S}^*) = 6 + 5 = 11$.

Recall the optimal allocation $\mathcal{S}^*$ is an optimal choice for the single time slot auction market. However, it does not mean that the allocation is optimal or efficient from the global DMP resource control perspective. The following three are key components of an auction mechanism design. By designing these three components carefully, we can control the overall mechanism in a desirable way. The details of the auction mechanism are explained in Section 4.5

- *Bidding police* for individual task agents: This is simply defined by the utility function $u(\bullet)$, which represents the *project group's preference* on a specific bundle (see Section 4.4).
- *Allocation rule*: This represents the *resource division's preference* on bundles. In the example in Figure 4.6, $\mathcal{S}^*$ calculating function is an example of allocation rule defined by Equation (4.1) (see Section 4.5).
- *Payment rule*: This *controls the incentive* for bidders to attend the auction in a desirable manner.

4.4 Utility Function

When a TA, M_i, receives a request for bid (RFB) message from an RA, M_i has to prepare a bid, denoted by $\mathcal{X}_i$. The RFB message contains the information on the re-scheduling horizon (or *control window*) $h = [t+1, t+n]$, within which the tasks will be re-scheduled. Within the control window, M_i generates the bid, which is a set of bundle-utility pairs, i.e., $\mathcal{X}_i = \{(B_{j,k}, u_i(B_{j,k})) : t+1 \leq j \leq k = j + d_i - 1 \leq t+n\}$, where the control window $h = [t+1, t+n]$, and d_i is the duration of the task i. The utility function $u_i(B_{j,k})$ is the measure that RAs use to determine an optimal allocation. In the approach, the utility function, $u_i(B_{j,k})$, is defined by two terms:

$$u_i(B_{j,k}) = v_i(B_{j,k}) - c_i(B_{j,k}) \tag{4.2}$$

The first term, $v_i(B_{j,k})$, is the valuation of the bundle $B_{j,k}$ for M_i and it is the negative value of *estimated deviation cost* $\hat{\theta}$, i.e., $v_i(B_{j,k}) = -\hat{\theta}$. Hence this value is the one that M_i wants to maximize. The second term, $c_i(B_{j,k})$, is the *aggregate precedence cost*, or penalty. Larger values of $c_i(B_{j,k})$ mean $B_{j,k}$ is less valuable for M_i due to precedence constraints with adjacent tasks. $c_i(B_{j,k})$ is the sum of precedence costs of individual time slots in h, which is addressed in Section 4.4.4 in detail. In this section, we discuss details on how the bids (in the form of bundle-utility pair) are generated and updated throughout the control loop iteration.

4.4.1 Task Duration Time

In order to define the estimated deviation cost, we need to define the task duration time model. One simple model is to assume deterministic duration times for each task. Two major stochastic duration time models are to use a uniform distribution and a beta distribution. Let δ depict a random variable representing time duration of a task. Equation (4.3) and Equation (4.4) define the probability density function (pdf) of the uniform and the beta distribution, respectively.

$$\begin{aligned} f(\delta) &= \tfrac{1}{b-a}; \qquad a \le \delta \le b \\ &= 0; \qquad \text{otherwise.} \end{aligned} \tag{4.3}$$

$$\begin{aligned} f(\delta) &= \tfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot \tfrac{1}{(b-a)^{\alpha+\beta-1}} \cdot (\delta-a)^{\alpha-1}(b-\delta)^{\beta-1}; \\ & \text{for} \quad a \le \delta \le b \quad \text{and} \quad \alpha > 0, \quad \beta > 0 \end{aligned} \tag{4.4}$$

4.4.2 Deviation Cost Function

We defined the concept of deviation cost function, denoted by $\theta(t)$, of a project in Chapter 2. Now we define two specific deviation cost function, a linear and an exponential function, as shown in Figure 4.7. These two functions satisfy the requirements of the deviation cost function: (1) increasing and convex with respect to time delay, and (2) no negative cost for the earliness. They are representative deviation cost functions in the sense that the linear function can be applied to cases where project cost is just proportional to tardiness, and the exponential function is good for the cases where the marginal cost of the project is increasing as the tardiness of the project increases. Equations (4.5) and (4.6) represent these functions formally.

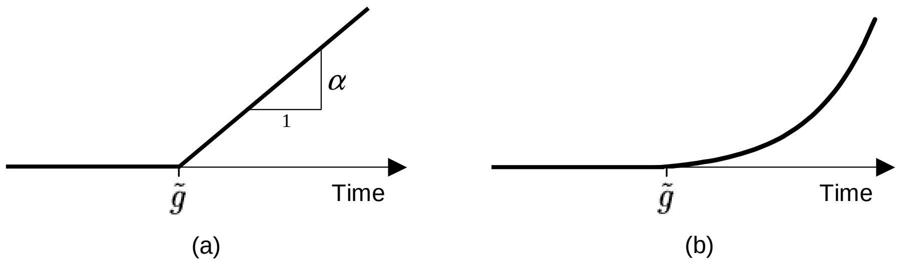


Fig. 4.7. Deviation cost functions: (a) linear function, (b) exponential function; $\tilde{g}$ is the nominal target completion time.

$$\theta(t) = \max\{\alpha(t-\tilde{g}),\ 0\}; \quad \text{for } -\infty \le t \le \infty \quad \text{and} \quad \alpha > 0. \tag{4.5}$$

$$\theta(t) = \max\{\mathrm{e}^{\alpha(t-\tilde{g})} - 1,\ 0\}; \quad \text{for } -\infty \le t \le \infty \quad \text{and} \quad \alpha > 0. \tag{4.6}$$

The deviation cost function is a function of project completion time. Based on the assumption of task level decentralization, each task agent bids for resources independently. In order to do so, each task agent must be able to evaluate time slot bundles based on the bounded information. We can easily assume that task agents can be able to get utility of the successor task agents, which is the minimum level of information exchange among task agents. Each agent calculates an estimated value of the deviation cost function based on the successor's deviation cost function and current schedule. We call this *estimated deviation cost function*, denoted by $\hat{\theta}_i(t)$ where t is the completion time of the task i.

4.4.3 Estimated Deviation Cost Function

Consider two adjacent tasks T_p and T_s in a project. T_p is the only predecessor of T_s (see Figure 4.8(a)). The random variable δ_s denotes T_s's time duration, and it follows a pdf $f_s(\bullet)$, which is defined over $[\underline{\delta}_s, \overline{\delta}_s]$. The commitment window of T_p is $[\underline{t}_p, \overline{t}_p]$. For convenience, assume the continuous time domain rather than time slot based discrete time domain. Then $\hat{\theta}_p(t)$, the estimated deviation cost function of T_p, can be calculated as follows:

$$\hat{\theta}_p(t) = \int_{\underline{\delta}_s}^{\overline{\delta}_s} \hat{\theta}_s(t + \delta_s) f_s(\delta_s) \mathrm{d}\delta_s \tag{4.7}$$

In the case of multiple tasks joining into a task as shown in Figure 4.8(b)), exact bundle evaluation cannot be achieved because of the complementarity among the bids of the agents. Hence, Equation (4.7) is used for this case, ignoring the complementarity among the multiple predecessor tasks. In the case of the task that has multiple successor tasks, as shown in Figure 4.8(c), the equation must be modified slightly. In this case, the $\hat{\theta}_p(t)$ must be define based on the worst case, or the highest estimated deviation cost, as shown in Equation (4.8). This equation is a general form of Equation (4.7), because it is working for case (a) and (b) as well. In Equation 4.8, $\hat{\theta}_s^j(t)$ is the estimated deviation cost function of the j-th successor task T_s^j, where $j \in S = \{1, ..., |S|\}$.

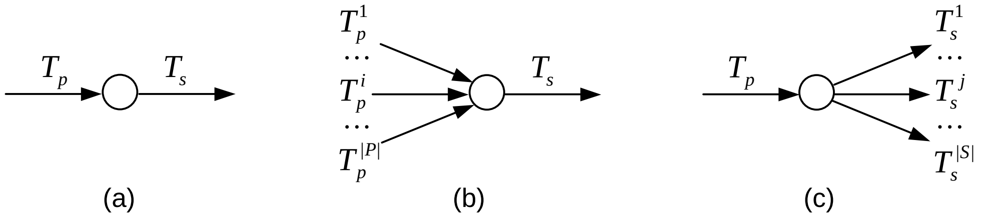


Fig. 4.8. Cases in precedence constraints: (a) one-to-one precedence; (b) many-to-one precedence; and (c) one-to-many precedence.

$$\hat{\theta}_p(t) = \max_{j \in S} \int_{\underline{\delta}_s^j}^{\overline{\delta}_s^j} \hat{\theta}_s^j(t + \delta_s^j) f_s^j(\delta_s^j) \mathrm{d}\delta_s^j \tag{4.8}$$

Assume that the task duration times follow the uniform distribution defined in Equation (4.3). If Equation (4.8) is rewritten as $\hat{\theta}_p(t) = \max_{j \in S}\{\hat{\theta}_p^j\}$, then $\hat{\theta}_p^j$ can be calculated using Equations (4.9) and (4.10) for the linear and exponential deviation cost function cases, respectively. As shown in the equations, we can easily define the recursive formula to generate a estimated deviation cost of the predecessor task from the ones of the successor tasks, by updating the parameters of the pdf's. These recursive equations, however, can only be used when the project network is in the form of a sequence of tasks, because the "$max()$" function makes the $\hat{\theta}_p(t)$ not to be the original pdf form anymore.

$$\begin{aligned} \hat{\theta}_s^j &= \alpha'(t - \tilde{g}') \\ &\quad \text{where } \alpha' = \tfrac{\alpha}{b-a} \text{ and } \tilde{g}' = \tilde{g} - \tfrac{a+b}{2}. \end{aligned} \tag{4.9}$$

$$\begin{aligned} \hat{\theta}_s^j &= \mathrm{e}^{\alpha(t-\tilde{g}')} - 1 \\ &\quad \text{where } \tilde{g}' = \tilde{g} - \ln\left(\tfrac{\mathrm{e}^b - \mathrm{e}^a}{\alpha(b-a)}\right)^{\frac{1}{\alpha}}. \end{aligned} \tag{4.10}$$

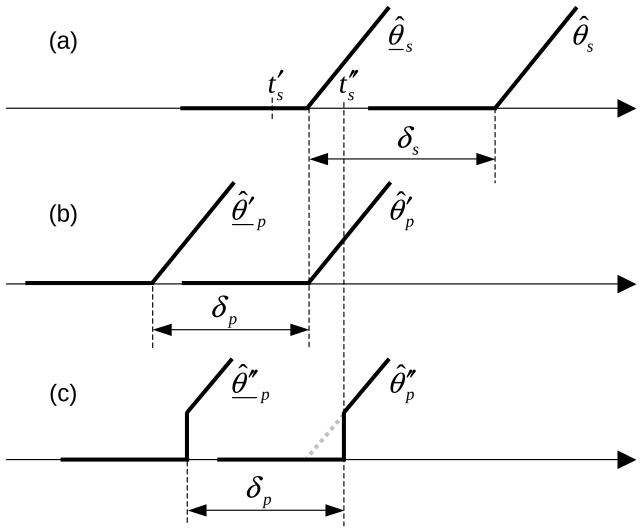
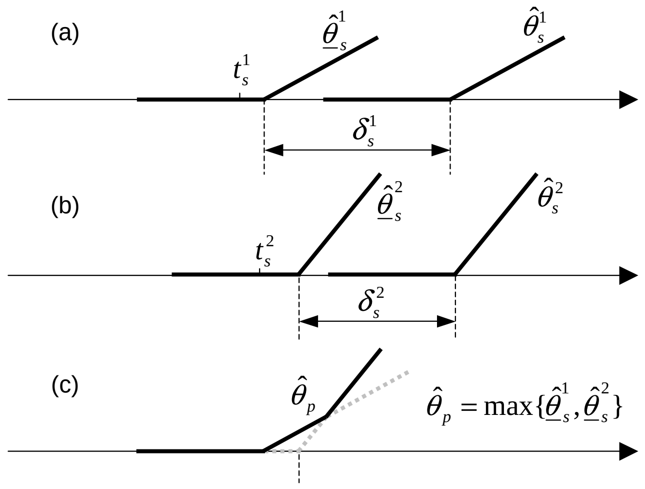


Fig. 4.9. One-to-one precedence case with deterministic duration times: (a) the successor task's estimated deviation cost function; (b) the predecessor tasks's estimated deviation cost function when the successor is scheduled to start at t'_s; (c) the predecessor tasks's estimated deviation cost function when the successor is scheduled to start at t''_s.

In addition to the above recursive relationship between $\hat{\theta}_p(t)$ and $\hat{\theta}_s(t)$, in the temporary economy clearing mechanism we need to take into account the effect of intermediate schedules in an iteration. Namely, each iteration $\hat{\theta}_\bullet(t)$'s must be re-evaluated based on the new resource allocation resulted by local market evaluations. Consider one-to-one and many-to-one precedence relations (see Figure 4.8(a),(b)). Figure 4.9 shows how the predecessor's deviation cost function ($\hat{\theta}_p(t)$) can be determined based on the successor's deviation cost function along with the current schedule (i.e., starting time t'_s or t''_s). This relationship is described by Equation (4.11). In the figure, $\underline{\hat{\theta}}_s$ denotes the estimated deviation cost function of task s when it starts at time t[11].

$$\begin{aligned} \hat{\theta}_p(t) &= \hat{\theta}_s(t+\delta_s); \quad t > t_s \\ &= 0; \qquad\qquad\quad t \le t_s \end{aligned} \tag{4.11}$$

[11]Remember that $\hat{\theta}_s(t)$ is the estimated deviation cost function of task s when it finishes at time t.



(a)
t_s^1
$\underline{\hat{\theta}}_s^1$
$\hat{\theta}_s^1$
δ_s^1
(b)
t_s^2
$\underline{\hat{\theta}}_s^2$
$\hat{\theta}_s^2$
δ_s^2
(c)
$\hat{\theta}_p$
$\hat{\theta}_p = \max\{\underline{\hat{\theta}}_s^1, \underline{\hat{\theta}}_s^2\}$

Fig. 4.10. One-to-two precedence case with deterministic duration times: (a) the first successor task's estimated deviation cost function. This task is scheduled to start at t_s^1; (b) the second successor task's estimated deviation cost function. This task is scheduled to start at t_s^2; (c) the predecessor task's estimated deviation cost function.

In one-to-many case (see Figure 4.8(c)), the predecessor task's estimated deviation cost function $\hat{\theta}_p(t)$ must be calculated by successor tasks' estimated deviation cost functions $\hat{\theta}_s^j(t)$ according to the following Equation (4.12), which is a general form of Equation (4.11). Figure 4.10 illustrates this relationship.

$$\hat{\theta}_p(t) = \max_{j \in S} \tilde{\theta}_s^j(t) \tag{4.12}$$

where:

$$\begin{aligned} \tilde{\theta}_s^j(t) &= \hat{\theta}_s^j(t + \delta_s^j); \quad t > t_s^j \\ &= 0; \qquad\qquad\quad t \le t_s^j \end{aligned} \tag{4.13}$$

4.4.4 Precedence Cost

The precedence cost vector, denoted by $\mathbf{c}_{i,k} = (c_{i,k}^1, ..., c_{i,k}^l, ..., c_{i,k}^n)^T$, is defined over the commitment window $[1, n]$ for each task $T_{i,k}$. Namely, each element $c_{i,k}^l$ in the vector is the cost caused by overlapping with preceding tasks, of the corresponding time slot l in the control window. This vector is to be updated in each iteration after new local optimal allocations are revealed in the way that the cost of the overlapped slots are increased and the cost of non-overlapped slots are decreased. The simplest precedence cost vector updating rule is to increase or decrease the cost of each vector element by a given fixed amount $\bar{c}$ as shown in Equation (4.14). The details on precedence adjustment are addressed in the following section.

$$\begin{aligned} c^l_{i,k} &= c^l_{i,k} + \bar{c}_I; && \text{if slot } l \text{ is overlapped} \\ &= \max(c^l_{i,k} - \bar{c}_D, 0); && \text{if slot } l \text{ is not occupied} \\ &= c^l_{i,k}; && \text{otherwise.} \end{aligned} \tag{4.14}$$

The sum of individual cost $c^l_{i,k}$ for a given time slot bundle is called the *aggregate precedence cost* of the bundle, denoted by $c_{i,k}(B_{b_1,b_2})$ where B_{b_1,b_2} represents the time slot bundle $[b_1, b_2]$.

$$c_{i,k}(B_{b_1,b_2}) = \sum_{l=b_1}^{b_2} c^l_{i,k}. \tag{4.15}$$

This value $c_i(B_{b_1,b_2})$ is added to the estimated deviation cost for generating the total utility of the task for the given bundle B_{b_1,b_2} as shown in Equation (4.2). If there is no change in the precedence cost vector for all the tasks within the control window, the temporary market will be cleared by clearing the individual local markets.

4.4.5 Precedence Cost Adjustment Methods

Precedence cost is used for discouraging allocations that cause precedence conflict between tasks. Unlike resource pricing, which is used to discourage multiple allocations on a single resource time slot, precedence cost adjustment has a clear “direction” for discouraging precedence violating allocations. The standard *tâtonnement* (see Equation (4.14)) has been used for resource price adjustment in literature. Indirection of this type of rule is one of the major factors in a market approach’s undesirable properties, such as cycling and ill-convergence. Figures 4.11 and 4.12 illustrate the standard price adjustment rule for precedence conflict and the proposed precedence cost adjustment rule, respectively.

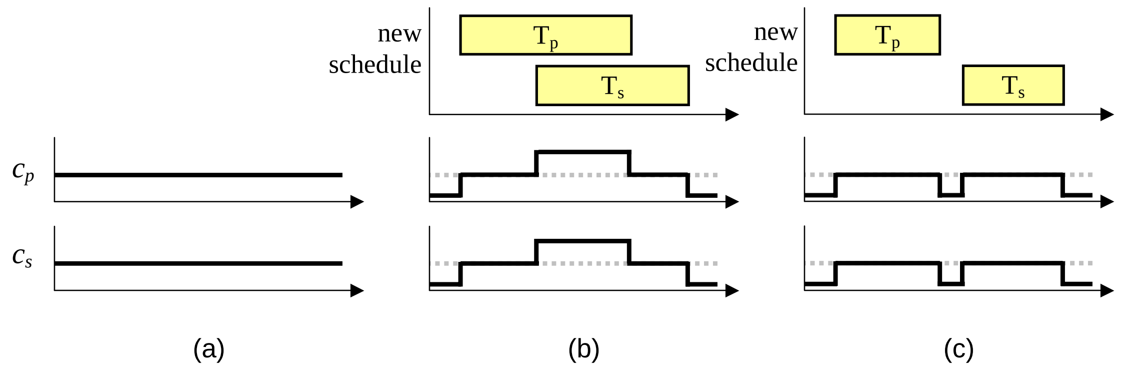


Fig. 4.11. A precedence adjustment rule based on standard tâtonnement: (a) before adjustment; (b) after adjustment in the case of a overlapped schedule; (c) after adjustment in the case of a separated schedule.

Assume that in a project we have a task T_s and its predecessor tasks T_h's, $h \in \mathcal{H}$: set of predecessor task indices. Then P-TÂTO's precedence adjustment rule for a task T_s, as a successor, can be represented by the following rule.

```
For all $h \in \mathcal{H}$ do:
    If  $\underline{t}_s \le \bar{t}_h$ ; where $\bar{t}_h = \underline{t}_h + \delta_h - 1$
        For $l := 1$ to $\bar{t}_h$ do:
            $c_s^l := c_s^l + \bar{c}_I$
For $l = \max_h(\bar{t}_h)$ to $\underline{t}_s - 1$ do:
    $c_s^l := \max(c_s^l - \bar{c}_D, 0)$
For $l = \underline{t}_s + \delta_s$ to $n$ do:
    $c_s^l := \max(c_s^l - \bar{c}_D, 0)$
```

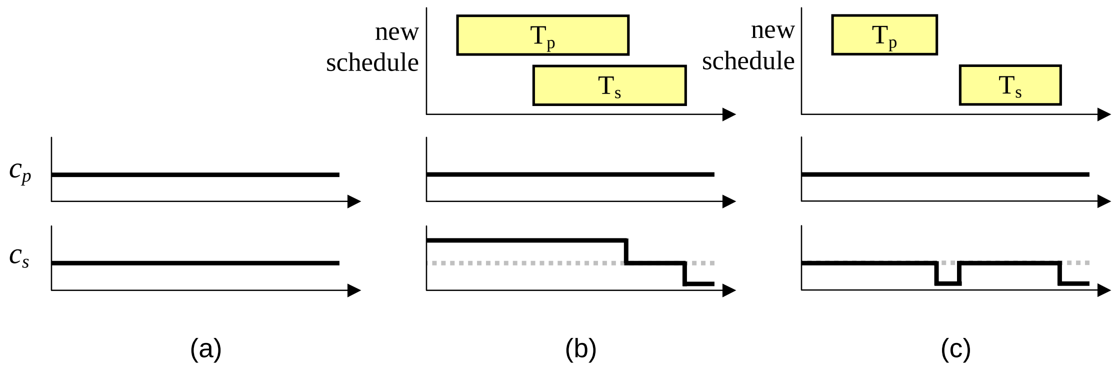


Fig. 4.12. P-TÂTO's precedence adjustment rule: (a) before adjustment; (b) after adjustment in the case of a overlapped schedule; (c) after adjustment in the case of a separated schedule.

In this rule, $\underline{t}_i$ and δ denote the starting time and the duration of the task T_i, respectively. n is the size of the control window. Using $\bar{c}_I \neq \bar{c}_D$, we can prevent possible cycling problem. We use $\bar{c}_D = \alpha\bar{c}_I, 0 < \alpha < 1$ in P-TÂTO implementation. $0.4 \leq \alpha \leq 0.8$ works reasonably well from the viewpoint of convergence speed and final solution quality. Detailed analysis on the effect of α on convergence speed is given in Section 6.6.

Now task agents' bids for resource time slot bundles can be generated and updated by these methods. Based on these methods, the utilities for set of time-slot bundles (or *bid profile*) of a task agent for a resource are dynamically changed over the iterations. Figure 4.13 illustrates the dynamic change of a bid profile.

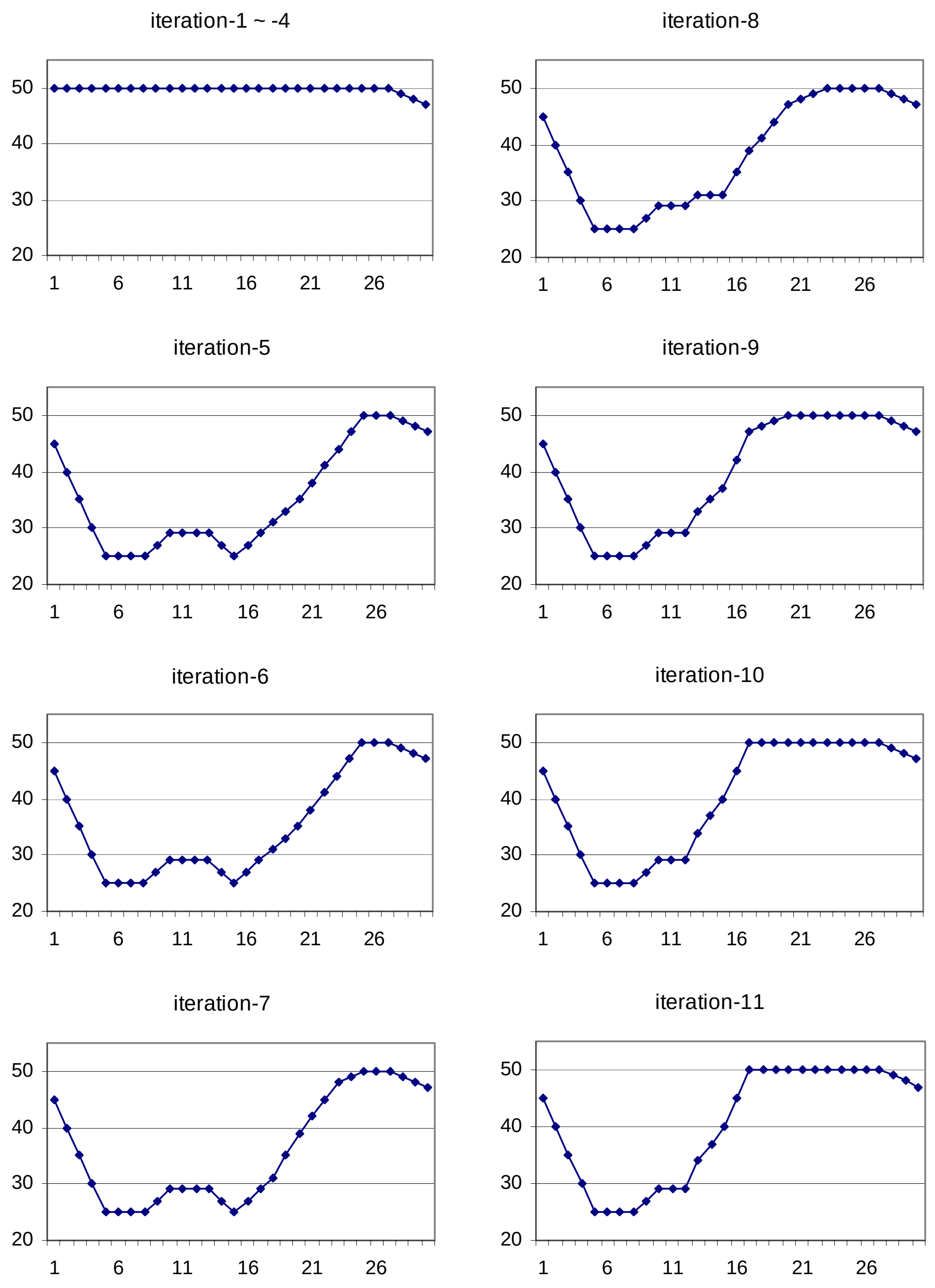


Fig. 4.13. Dynamics of bid profile: an example (Data set: 35J81, Task (2,5))

4.5 Local Market Evaluation Mechanism

Suppose an RA determined the list of TAs $\mathcal{M} = \{M_i\}$ and a scheduling horizon $h = [1, n]$, and sent RFBs to all $M_i \in \mathcal{M}$, thereby, forming a local market. Using the formulation explained in Section 4.3.2, the auction mechanism can be summarized as follows.

(1) Each task agent $M_i \in \mathcal{M}$ generates a bid $\mathcal{X}_i = \{(B_{j,k}, u_i(B_{j,k}) : 1 \le j \le k \le n\}$, using Equation (4.2).

(2) The RA computes the optimal allocation:

$$\mathcal{S}^* = \arg\max_{\mathcal{S} \subset \mathcal{B}} \{W(\mathcal{S}; \mathcal{X}) : B \cap B' = \emptyset \text{ for every } B, B' \in \mathcal{S}\}. \tag{4.16}$$

(3) Inform the winning bid information to each of the tasks.

The bid generation step was already explained in the previous section. Equation (4.16) generalizes Equation (4.1) by introducing a function $W(\mathcal{S}; \mathcal{X})$. The function $W(\mathcal{S}; \mathcal{X})$ defines the preference of the RA. The most desirable goal of the resource division is to maximize an aggregate utility, which can be defined as a function that maps the set of bidders' individual utilities to a single real-valued *social utility.* We call this function, $W(\mathcal{S}; \mathcal{X})$, a *welfare function.* If we choose the welfare function as an increasing function with respect to the individual utilities, the welfare maximizing allocation $\mathcal{S}^*$ is Pareto efficient. Figure 4.14 depicts conditions and relationships among Pareto efficiency, Walrasian (or general) equilibrium, and social welfare maximum (Varian, 1984; Campbell, 1987). Our approach is to maximize the social welfare, which also guarantees Pareto efficiency by defining $W(\mathcal{S}; \mathcal{X})$ as a positive weighted sum of the utilities of bundles. The weights can be easily given in the way that each project has its own weight depending on its importance or urgency, which can be dynamically assigned by the top management depending on the situation.

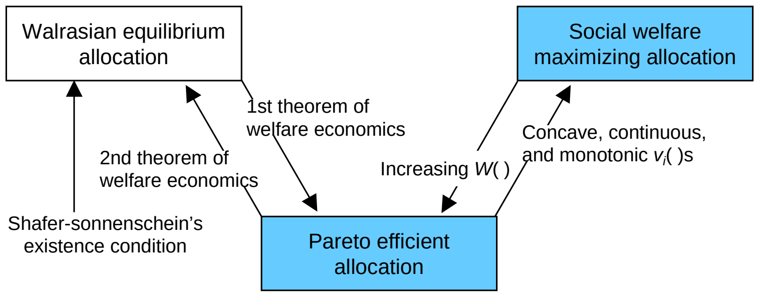

Walrasian equilibrium allocation
Social welfare maximizing allocation
1st theorem of welfare economics
Concave, continuous, and monotonic $v_i(\,)$s
2nd theorem of welfare economics
Increasing $W(\,)$
Shafer-sonnenschein's existence condition
Pareto efficient allocation

Fig. 4.14. The relationship among Pareto efficiency, Walrasian equilibrium, and welfare maximum.

Now let us discuss an efficient algorithm for optimal allocation determination. In a general combinatorial auction problem, the optimization problem to calculate the optimal allocation $\mathcal{S}^*$ is $\mathcal{NP}$-complete[12]. Our problem is not a general combinatorial auction problem. Instead, it is corresponding to a *single-machine scheduling problem with general utility function.* This problem still cannot be solved in a polynomial algorithm. We developed an efficient heuristic algorithm to calculate local market winning bids. The procedure consists of three steps: (1) initial sequencing based on utility distribution, (2) calculate the optimal allocation in the given sequence, (3) repeat pair-wise exchange and Step (2) until no more improvement is made.

4.5.1 Initial Sequencing

Initial sequencing heuristics can be explained as follows:

(1) Get the task-wise best schedule (bundle) independently.

(2) Sort the task list with respect to the best utility values.

(3) Fix the sequence one by one according to the sequence generated by Item (2). If a task cannot be scheduled on the best-utility bundle just put it at the end of the current sequence.

[12]This problem, selecting the winning set of bids, can be formulated as an Integer Program. This formulation is an instance of the *set packing problem*, which is $\mathcal{NP}$-complete (Vries and Vohra, 2000).

Assume that we have a resource, for which three tasks (Task1, Task2 and Task3) bid. The duration times for each task are 3, 2 and 2. Figure 4.15 shows the bid profiles of each task for resource bundles. Remember that the x-axis in the figure denotes the finish time of the tasks, meaning that each y value in the figure is a utility value for the bundle ending with the slot indexed by the x value. In the example, after the first step we get the task-wise best bundles: $B_{1,3}$ for Task1 with $u(B_{1,3}) = 10$, $B_{5,6}$ for Task2 with $u(B_{5,6}) = 8$, and $B_{4,5}$ for Task3 with $u(B_{4,5}) = 7$. Then the tasks are sequenced according to the best utility values: Task1 $\rightarrow$ Task2 $\rightarrow$ Task3. The tasks are to be sequenced one by one according to the previous order. First, Task1 is located at the bundle $B_{1,3}$. Second, Task2 is located at the bundle $B_{5,6}$ without any conflict with already allocated bundles. Lastly, the best utility bundle of Tasks3 conflicts with Task2's allocated bundle. Hence simply add the task bundle at the end of the sequence. The initial sequence is, therefore, Task1 $\rightarrow$ Task2 $\rightarrow$ Task3.

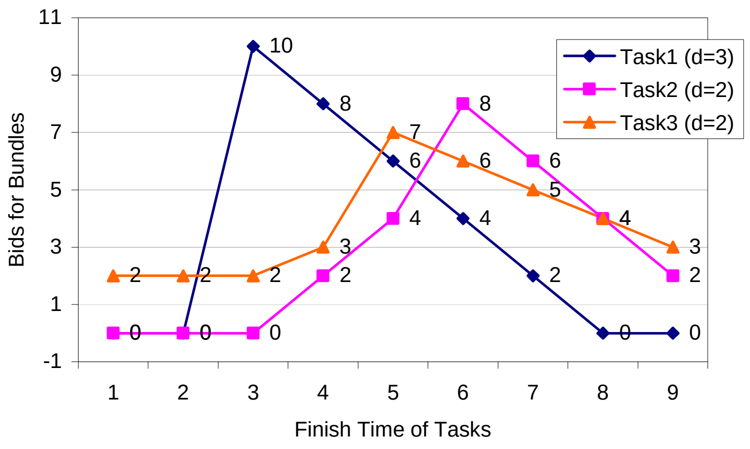


Fig. 4.15. An example of bid profiles for a resource: The duration time for Task1, Task2 and Task3 are 3, 2 and 2, respectively.

4.5.2 Optimal Allocation in a Given Sequence

Given a task sequence, optimal local schedule determination problem can be formulated by Dynamic Programming (DP). Assume that we have a task sequence $\{T_1, ..., T_m\}$ to be scheduled on a single resource over a time horizon $[1, n]$. Each T_i

takes δ_i time slots. Then our problem can be solved using the following DP formulation. The final solution is the task sequence corresponding to the optimal welfare value $S(1, ES_1)$, where ES_1 is the earliest start time of T_1. During the recursive DP problem solution procedure, the intermediate $S(i,t)$'s are stored in a *hash table*, in order to avoid redundant subproblem solving.

$$
\begin{aligned}
\text{OVF}^{a}\text{: } S(i,t) &= \text{Maximum welfare value for the task set } \{T_i, ..., T_m\} \\
&\quad\ \text{over the scheduling horizon } [t, n]. \\
\text{ARG}^{b}\text{: } (i,t) &= i \text{ is the index of the first task to be scheduled;} \\
&\quad\ t \text{ is the index of the starting time slot.} \\
\text{OPF}^{c}\text{: } P(i,t) &= \text{the gap between the starting times of } T_i \text{ and } T_{i+1}. \\
\text{RR}^{d}\text{: } S(i,t) &= \max_{\delta_i \le P(i,t) \le LS_{i+1} - t} \big(S(i+1, t+P(i,t)) + u(T_i, t+\delta_i - 1)\big) \\
&\quad\ \text{where } LS_{i+1} \text{ is the latest starting time of } T_{i+1} \\
\text{BC}^{e}\text{: } S(m,k) &= u(T_m, k), \text{ which is } T_m\text{'s utility for the bundle } [k - \delta_m + 1, k]. \\
\text{ANS}^{f} &= S(1, ES_1).
\end{aligned}
\tag{4.17}
$$

[a] Optimal value function, which assigns the optimal value to each subproblem.

[b] Arguments of OVF, which are symbols that designates a particular subproblem.

[c] Optimal policy function, which associates the best decision with each subproblem.

[d] Recurrence relation, which is the relations among various values of OVF obtained after applying the *principle of optimality*.

[e] Boundary conditions, which are obvious values of OVF for certain argument values obtained from the statement of the problem.

[f] Answer, which is the optimal value and decision for the value of the arguments that represent the whole problem.

4.5.3 Exchanging and Stopping Rule

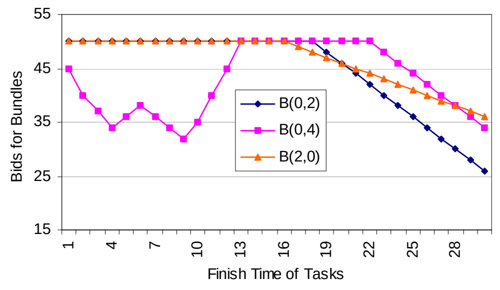
Suppose n tasks are to be scheduled for a resource. Let $currTL$ and $optiTL$ denote lists of tasks. $optiTL$ and W are an optimal task list and its welfare value, respectively.

The function exchange($currTL, i, i+1$) exchanges the i-th task and $i+1$-th task in the $currTL$, and return the new task list. The function getOptimalSchedule($currTL$) finds the optimal schedule within the given sequence $currTL$ and returns the welfare value. The resultant $optiTL$ is a task sequence in which no possible pair-wise exchange can improve the welfare value.

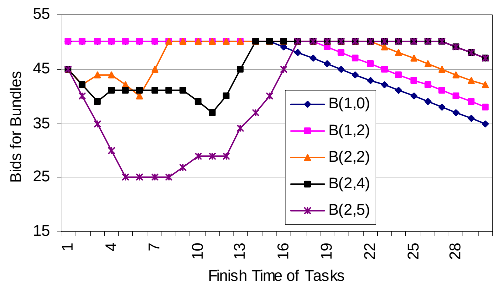
```
$improved := true$
$W := -BigNumber$
While ($improved$) do:
    For $i := 1$ to $n-1$ do:
        $currTL :=$ exchange($currTL, i, i+1$)
        $w :=$ getOptimalSchedule($currTL$)
        $improved := (w > W)$
        If ($improved$) do:
            $W := w$
            $optiTL := currTL$
            break out of For-loop
        Enddo
    Enddo
Enddo
```

The proposed search heuristics dramatically reduce the search space, yet generate a high level of optimality. In an experimental analysis, it is shown that the search space of the algorithm increases linearly according to the increase of the number of tasks, and the optimality ranges between 0.998 to 1.0. Details are explained in Section 6.2.

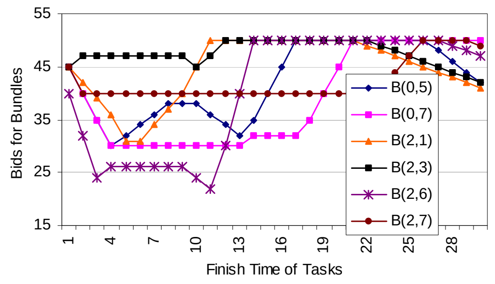
Figure 4.16 shows an example of bid profiles for each resource in a DMP sample, which is investigated previously. Based on this bid profiles local optimal allocation is determined as shown in Figure 4.17



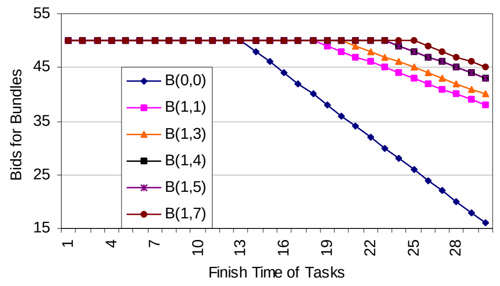
(a) Resource 0 (R0)



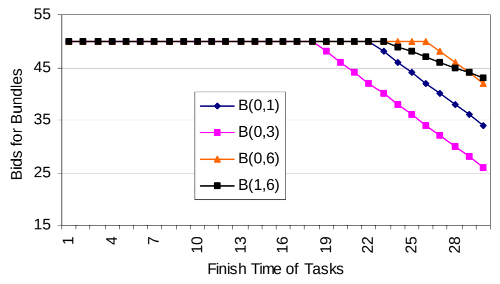
(d) Resource 3 (R3)



(b) Resource 1 (R1)



(e) Resource 4 (R4)



(c) Resource 2 (R2)

Fig. 4.16. An example of bid profiles for each resource (Data Set: 35J81, at the last (11th) iteration).

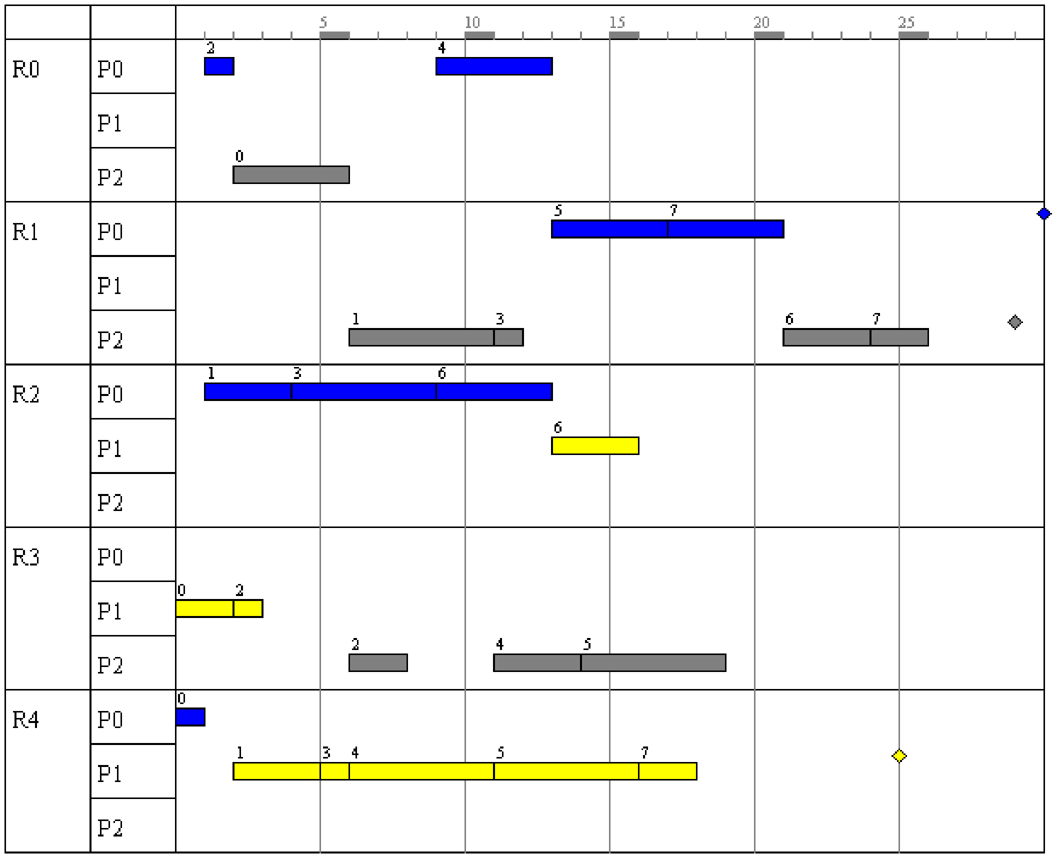


Fig. 4.17. Final schedule based on the bid profiles in Figure 4.16 (the diamond symbols represent the due dates of each project).

4.6 Summary

In this chapter a market mechanism (P-TÂTO) for DMP resource control problem is presented. Because of the distributed and decentralized nature of the DMP resource control problem, the control mechanism must be designed and implemented on top of a distributed infrastructure, in which multiple decision makers collaborate with each other. In this sense, we first defined the MAS based virtual market model in a conceptual level. We proposed a novel virtual economy model, which incorporates both the dynamic and the distributed nature of the DMP resource control problem by organizing the market in three different levels: (1) DMP economy, (2) temporary economy and (3) local market levels. This virtual economy model is embodied by multiple decision makers whose roles are producers, consumers, and mediators in a market situation, thereby constituting a multiagent system. The market-based control approach is a *bottom-up* approach, which minimizes the direct and/or hierarchical control of a "*big brother*". So, the proposed multiagent organization involves more direct interactions among end-buyers and end-sellers, compared to the master-slave organization model, which was proposed by Satapathy (1999) for a procurement problem (see Figure 4.18).

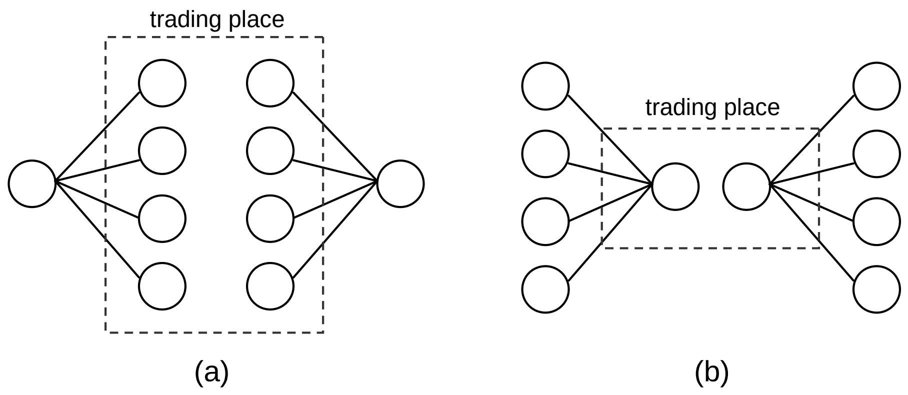


Fig. 4.18. Two different multiagent organization for a trading situation: (a) market-oriented organization for DMP resource control problem, (b) master-slave model for a procurement problem by Satapathy (1999).

In tackling two major constrains - precedence constrains and resource constrains - in DMP resource control problem, the P-TÂTO mechanism resolves the resource constrains in an optimal way in each local market and resolves the precedence constrains throughout a tâtonnement type process in the temporary economy level. The local market evaluation mechanism is formulated as a combinatorial auction mechanism. The major drawback of a combinatorial auction mechanism is the computational complexity of determining the winning bids. We developed an efficient dynamic programming based heuristic algorithm. In the local markets, each buyer's bid (and ultimately utility values on each time-slot bundles) is generated based on estimated deviation costs and precedence cost vectors. By combining these two factors, the overall mechanism can tackle both resource and precedence constrains simultaneously. In preparing the bids, each bidder (task agent) exchanges the minimum level of information, which is the utility profile of the successor task agents, so that the whole mechanism takes full advantages of a decentralized mechanism. In the temporary economy level precedence cost adjustment procedure, we applied the tâtonnement type process on the precedence cost (or penalty) rather than resource prices, which is the basic idea of the original tâtonnement process. The clear direction of improvement of the solution, which is gained from the precedence tâtonnement, results in a high level of solution quality and convergency, which will be examined in Chapter 6. In Chapter 5 we present the virtual market model and the P-TÂTO mechanism more rigorously from the information system's viewpoint.

Chapter 5

Multiagent-based Information Infrastructure

In this chapter, we present the details of the multiagent-based computational model of the *DMP economy* and implementation of the P-Tâto mechanism. The conceptual-level design of the virtual economy model was already explained in Chapter 4. Hence, we discuss directly the detailed model of individual DMP economy agents, their behaviors, and how the behaviors implement the overall P-Tâto mechanism, followed by a review of representative agent architectures in the literature. The architecture of an individual DMP economy agent is a hybrid architecture, which incorporates some aspects of existing agent architectures including *collaborative architectures* (see Section 5.1.5), *layered architectures* (see Section 5.1.4), and *reactive architectures* (see Section 5.1.2). The three major characteristics of the DMP economy agents are its *behavior-oriented*, *state-based*, and *message-driven* nature.

5.1 Architectures for Software Agents

The question *what is an agent?* is one of the most fundamental yet controversial questions in the agent-based computing community. A common way to define the term *agent* is to denote an agent using its necessary properties or *agency*. An agent denotes a software[1] component that has the following properties (Wooldridge and Jennings, 1995; Jennings et al., 1998):

- *autonomy*: Agents should be able to act without the direct intervention of humans or their agent peers;
- *social ability*: agents should be able to interact with humans or other agent peers to carry out specific roles or to achieve some goals;
- *responsiveness*: agents should be able to perceive their environment and respond in a timely manner to changes that occur in the environment; and
- *pro-activeness*: agents should be able to exhibit opportunistic, goal-directed behavior and take initiative when appropriate.

These four characteristics are usually referred to as a *weak* notion of agency. Other researchers add more rigorous *mentalistic* notions, which have been commonly discussed in AI (Artificial Intelligence) literature: *knowledge*, *belief*, *intention* and *obligation* (for further reading, see (Wooldridge and Jennings, 1995)). Due to the diverse definitions of agency, the computational model or architecture of agents are also very much diverse. Nevertheless, a definition of agency and a corresponding architecture must be *situated* in specific problem domains. Namely, different problem domains require different levels of agency and different designs of internal architecture. In the literature, various architectures of software agents have been reported. In this section we discuss some of the representative architectures in the literature before discussing the agent architecture for the DMP economy, which is eventually a combination of the desirable component of the representative architectures, in Section 5.2.

[1]We restrict the scope of agents to software agents, excluding physical agents such as robots.

5.1.1 Logic-based Architectures

Although agent research has an interdisciplinary aspect, the AI community contributes one of major roles in the foundation of agent research. In this sense, it seems natural for some of the earlier agent architectures to follow the tradition of *symbolic AI*[2], and one of these traditions is to use *logical formulae* as the symbolic representation, and accordingly make decisions throughout *logical deduction* (or *theory proving*). Well-known examples of pure logic-based architecture include the CONGOLOG system (Lespérance et al., 1996), and the MetateM and Concurrent MetateM programming languages (Fisher, 1994). Despite their neat logical semantics, logic-based approaches have a critical disadvantage: the inherent computational complexity of theorem proving. Hence, it is very questionable whether such an agent can operate effectively in a time-constrained and dynamic environment such as the DMP resource control problem domain.

5.1.2 Reactive Architectures

Some researchers have proposed a totally different approach, which does not rely on symbolic representation of knowledge/information and symbolic reasoning. They are usually referred to as *behavioral* (since a common theme is that of developing and combining individual behaviors), *situated* (since a common theme is that of agents are actually situated in some environment, rather than being disembodied from it), and finally *reactive* (because such systems are often understood as simple reacting to an environment, without reasoning about it) architecture. One of the earliest reactive architectures is Brooks' *subsumption architecture* (1986). The subsumption architecture involves a hierarchy of activation, inhibition, and responses to the external world. This architecture has a layered structure, in which higher level layers "subsume" the roles of lower level layers when they wish to take control (see Figure 5.1). In addition, multiple behaviors associated with a specific level (or layer) of activities compete to win control over the

[2]Symbolic AI suggests that intelligent behaviors can be generated in a system by giving the system a symbolic representation of its environment and desired behaviors, and by syntactically manipulating this representation (Weiss, 1999).

activities. The subsumption architecture has been implemented in a variety of physical robots with different abilities, including relatively simple robotic agents such as AGV (Automated Guided Vehicle) in agent-based manufacturing systems.

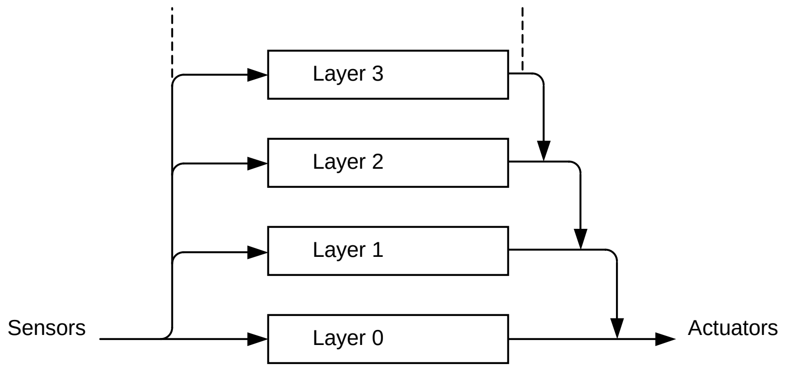


Fig. 5.1. Subsumption architecture (Brooks, 1986)

5.1.3 Belief-Desire-Intension Architectures

Other researchs model a software agent as an *intentional system* (Donnett, 1987), meaning that an agent's behavior can be represented and reasoned by the attribution of *attitudes* such as belief, knowledge, desire, intention, commitment, and choice. The two main research issues in this approach are: (i) logically consistent definitions on interactions between the attitudes, and (ii) ideal combinations of the attitudes required for agent characterization (Wooldridge and Jennings, 1995). So far, various models have been suggested including Levesques' *belief and awareness* (Levesque, 1984), Koniology's *the induction model* (Konolgie, 1986), a few number of *metra-language formalism* (Haas, 1986)(Morgenstern, 1987)(Davies, 1993), Cohen and Leverage's model of *intention* (Cohen and Levesque, 1990), Rad and Georgian's *belief, desire, intention architecture* (Rao and Georgeff, 1991a,b, 1993), Singh's family of logics for some attitudes in a branch-time framework (Singh, 1994), and Worldwide's formalization of multiagent systems (Wooldridge, 1992).

Among all of them, the most popular and influential architecture might be the Rad and Georgian's *belief*, *desire* and *intention* (BDI) model, which is characterized by a mental state with the three components - belief, desire and intention. *Beliefs* describe the states of the world that the agent works on; *desires* represent *options* available to the agent - different possible states of affairs to which the agent may choose to commit; and *intentions* represent states of affairs that the agent has chosen and has committed to. A number of BDI agent systems have been implemented, and the best-known is the *Procedural Reasoning System* (PRS) (Georgeff and Lansky, 1987; SRI International, 1999) and its variants (Lee et al., 1994). Figure 5.2 depicts a practical BDI agent architecture, which is a simplified version of PRS. This architecture consists of (i) current *beliefs* or facts about the world, (ii) a set of *goals*[3], (iii) a set of *plans* describing how *sequences of actions*[4] may be performed to achieve certain goals or to react to particular situations, and (iv) *intentions* containing those plans that have been chosen for execution. *Interpreter* manipulates these components, selecting appropriate plans for execution based on the system's beliefs and goals, creating the corresponding intentions, and then executing them.

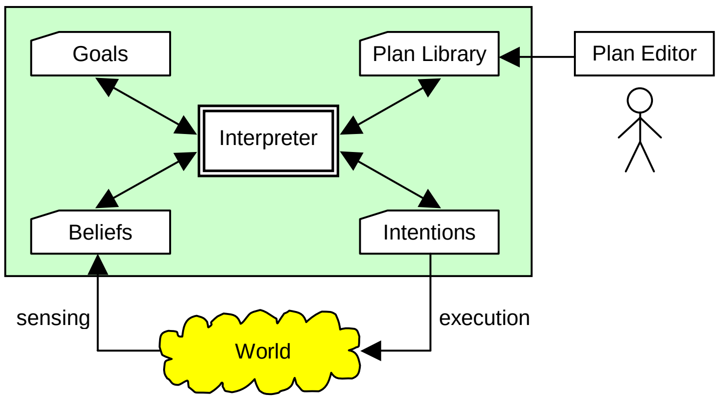


Fig. 5.2. A practical BDI architecture

[3]*Goals* are defined as *mutually consistent desires.* In most practical systems *desires* in BDI formalism can be restricted to *goals.*

[4]This is a fundamental argument about human knowledge in PRS. Namely, it is assumed that much of human knowledge about how to achieve specific goals is *procedural* or sequential in nature.

The BDI model is attractive for several reasons. First, it is intuitive - we all recognize the processes of deciding what to do and then how to do it, and we all have an informal understanding of the notions of belief, desire and intension[5]. Second, it gives us a clear functional decomposition, which indicates what sort of subsystems might be required to build an agent. But the main difficulty is about how to efficiently implement these functions. Also for relatively well-defined collaboration mechanisms (for example, Contract-Net style protocols as discussed in Section 5.1.5) or more dynamic environment requiring immediate reactions, the BDI architecture is a luxurious and/or complicated model.

5.1.4 Layered Architectures

Needs for both reactive response and intensional behavior lead researches to development of layered architectures, which are organized with components or *layers* for different levels of decision making (from direct perception to complicated reasoning). For example, the perception feeds the reasoning subsystem, which governs the actions, including deciding what to perceive next (Huhns and Singh, 1998). We classify this type of architectures into two types in control flow within the layered architecture (Müller et al., 1995):

- *Horizontal layering*: In this architecture, each layer is directly connected to the sensory input and action output. Each layer itself acts like an independent decision maker although it can share common beliefs with other layers (see Figure 5.3(a)).
- *Vertical layering*: In this architecture, each sensory input and action output is dealt with by a specific layer. Vertically layered architectures can be further divided into *one pass* architectures (Figure 5.3(b)) and *two pass* architectures (Figure 5.3(c)).

There is no pure layered architecture, which can be excluded from other approaches. Instead, the design of each layer or interaction among each layer calls for some kind of hybrid architecture by incorporating other approaches.

[5]In this sense, BDI model is categorized as *practical reasoning* architecture.

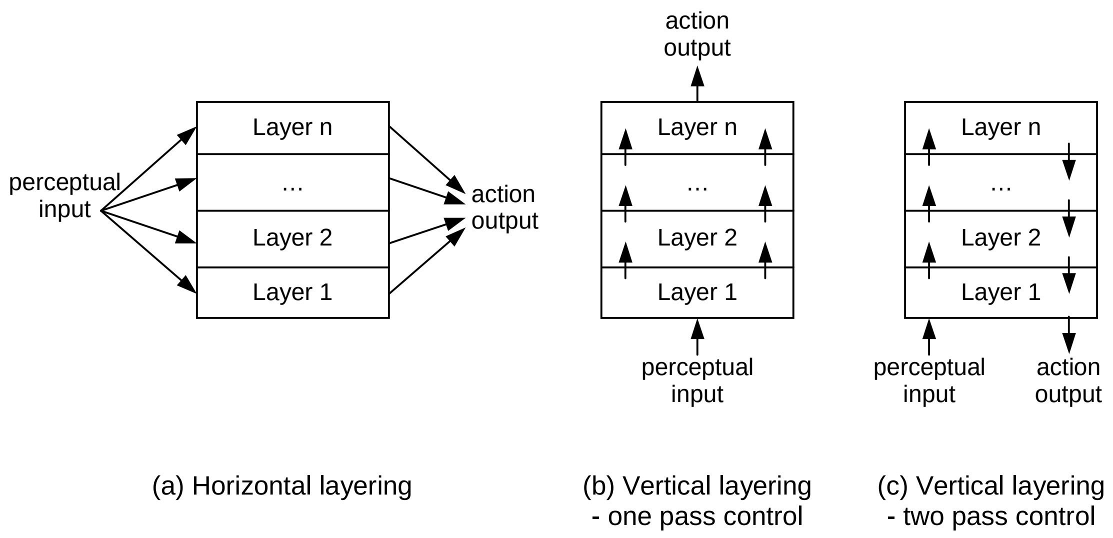


Fig. 5.3. Information and control flows in three types of layered agent architecture (Müller et al., 1995).

The great advantage of horizontally layered architectures is their conceptual simplicity and flexibility: if we need an agent to exhibit n different types of behaviors, then we can implement them in different layers. Hence, if an application domain requires an agent to have multiple roles within a MAS, this architecture can be easily applied.

5.1.5 Collaborative Architectures

Collaborative agents or *social agents* work together to solve specific problems through interaction among the agents; hence, communication between agents is an important element. Although each individual agent still acts autonomously, it is the synergy resulting from their cooperation that makes collaborative agents interesting and useful. Internal architecture of collaborative agents can be any other architecture explained previously. Collaborative attitudes are incorporated in such agent systems to facilitate interaction, communication, task decomposition, distribution, cooperation, and negotiation (Shen and Norrie, 1999). Agents using the Contract Net (Smith, 1980) as the basis for their inter-agent negotiation protocol is a typical example of collaborative agents. Although collaborative agents can have any of the architectures explained previously, their behaviors can be thought of as a component part of an overall multiagent system's governing mechanism. Hence their internal architecture is typically based on relatively simple processes, which are activated by incoming messages and generate outgoing messages from/to other agents.

Consider agents using the Contract Net protocol. The Contract Net provides a solution for so-called *connection problems*: finding an appropriate agent to work on a given task. The basic steps in the Contract Net can be explained as follows:

1. A *manager*, who wants to solve a task, announces the existence of tasks via a (possibly selective) multicast to potential *contractors*, who might be able to solve the tasks of the manager.

2. Potential contractors evaluate the announcement. Some of them submit bids.

3. The manager awards a contract to the most appropriate contractor.

Figure 5.4 shows the internal decision making mechanisms in a manager agent and a potential contractor agent. As shown in the figure, if the collaboration mechanism is clearly defined, the internal decision making process is also well defined by multiple steps including message passing functionalities.

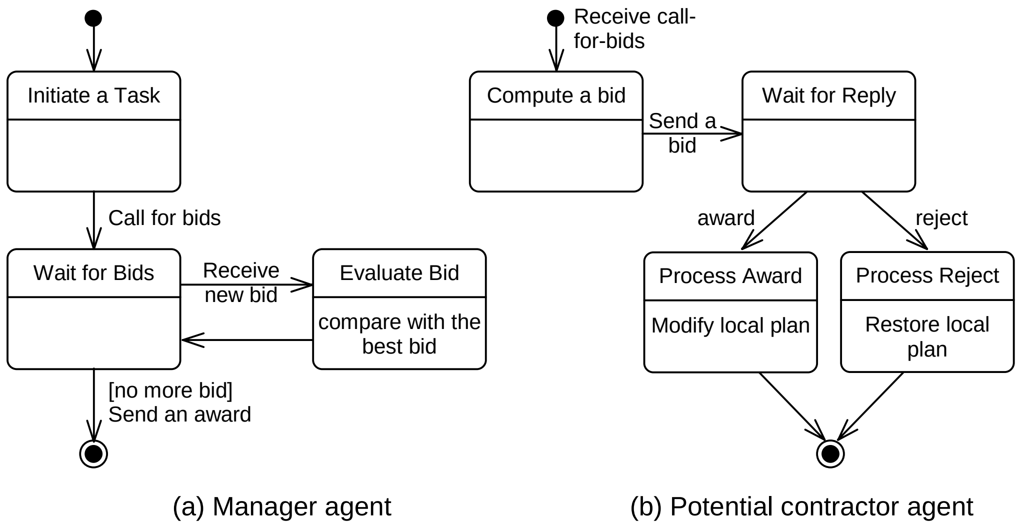


Fig. 5.4. State diagrams for internal decision making processes of a manager agent and a potential contractor agent.

5.2 Agent Model for DMP Economy

The individual agent architecture for the DMP economy adopts some of the desirable aspects of different architectures explained in Section 5.1. The three major characteristics of the individual DMP economy agent model are:

1) **Behavior-Oriented**: As shown in the cases of subsumption architectures or horizontally layered architectures, the overall role of an agent is reactively carried out by independent behaviors. As mentioned, an agent having multiple (quite independent) roles, which is defined under an overall control mechanism or protocol, and/or operates in a dynamic and time-constrained situations, can be effectively implemented by a behavior-oriented architecture. The proposed DMP agent architecture relies more on the idea of horizontally layered architectures. The relationship between agents and behaviors is explained in Section 5.2.1

2) **State-Based**: Each behavior within an agent cannot be a single-flow process. Instead it must be defined by inter-agent interactions or protocols. This means the behavior must explicitly maintain the state within a cycle of the protocol, so that the behavior can be proceeded from the state when the behavior is called next time. So, each behavior is modelled as an augmented finite state machine, which will be explained in Section 5.2.3.

3) **Message-Driven**: Each behavior as an augmented finite state machine has a set of states and transitions. Here, the only external source of events, which causes any transition, is an incoming message[6] from behaviors of other agents. On the other hand, the only way to cause any transition within other behaviors is by sending a message to the agent along with a proper message header.

[6]Here, the message means explicit form of data/information. We use ACL (Agent Communication Language) as the vehicle of the message.

5.2.1 Agents and Platform

We followed FIPA (Foundation for Intelligent Physical Agents) agent management specification (FIPA, 1998) as a platform for the proposed MAS and used a JADE (Bellifemine et al., 1999) to implement our DMP economy. Agents reside on an *agent platform* by registering themselves to the platform (exactly speaking to the agent management system within the agent platform). Figure 5.5 shows a simplified view of the agent management reference model.

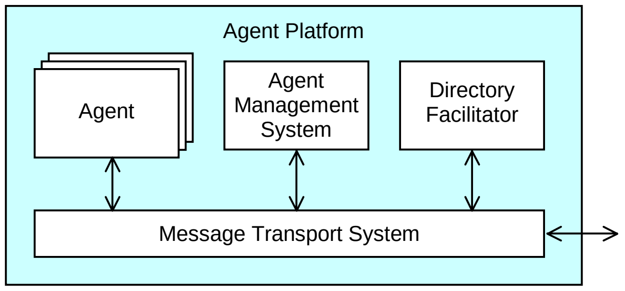


Fig. 5.5. Reference architecture of a FIPA Agent Platform (FIPA, 1998)

- **Agents**: An agent is the basic actor on an agent platform, which has multiple service capability (i.e., behaviors in a DMP environment) including access to external software, human users and communications facilities.
- **Agent Management System**: It is an agent that is in charge of supervisory control of an agent platform, including creation, deletion, suspension, resumption authentication and migration of agents on the agent platform, and provide a “white pages” directory service for all agents on an agent platform.
- **Directory Facilitator**: It is an agent that provides a “yellow pages” directory service for the agents. It stores descriptions of the agents and the services they offer.
- **Message Transport System**: It is the software component that controls all the exchange of messages within the platform, including messages from/to other agent platforms.

The details on FIPA agent management specification can be found in (FIPA, 1998) and (Bellifemine et al., 1999).

5.2.1.1 The `Agent` class

The `Agent` class is a common base class for other function-specific agents, meaning that they inherit (1) features to accomplish basic interactions with the other agents from this `Agent` class, and (2) a basic set of methods that can be called to implement the custom behavior of the agents such as sending/receiving messages. For the DMP resource control system, five agent classes constitute a DMP economy as explained in Chapter 4. Each agent class inherits from the general class `Agent` (see Figure 5.6). Each agent has different behaviors depending on its role in the market protocols.

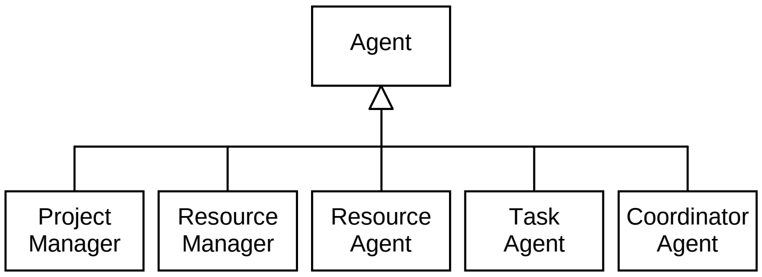


Fig. 5.6. Class diagram of agents

5.2.2 Agent Roles and Behaviors

Figure 5.7 shows the relationship between `Agent` class and `Behavior` class. An agent can *have* multiple behaviors with a strong ownership (or *composite aggregation*), meaning that a behavior must be created within an agent and destroyed when the agent is destroyed. An agent's role in a mechanism is implemented by a behavior. That means that we can easily add new functionality or a role to an agent by adding a new behavior object.

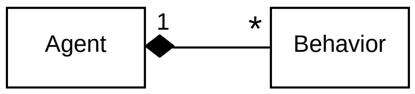


Fig. 5.7. Class diagram of agent and behavior

5.2.3 Behavior Model and Notations

As mentioned, the proposed behavior model is based on states, which is an effective approach for collaborative agents. Formal definition of the proposed behavior model as an augmented finite state machine (or finite automaton) is as follows:

Definition 5.1. (Behavior) A *behavior* $\mathcal{F}$ is a 8-tuple $\langle Q, \tilde{Q}, \Sigma_i, \Sigma_o, q_0, \delta_m, \delta_p, \mu \rangle$, where:

Q is a finite set of *states*.

$\tilde{Q} \subseteq Q$ is a finite set of *action states*, which involve specific actions.

Σ_i is a finite set of input message types.

Σ_o is a finite set of output message types.

$q_0 \in Q$ (the *initial* state).

δ_m is a function from $(Q - \tilde{Q}) \times \Sigma_i$ to Q (*message-driven transition* function).

δ_p is a function from $\tilde{Q}$ to Q (*procedure-driven transition* function).

μ is a function from $\tilde{Q}$ to Σ_o (*messaging* function).

For any element q of *waiting states* $(Q - \tilde{Q})$ and any input message type $\sigma \in \Sigma_i$, we interpret a *transition* $\delta_m(q, \sigma)$ as the new state to which the behavior's current state q moves, if it is in state q and receives a message of type σ. For any element q of action state $(\tilde{Q})$, we interpret a transition $\delta_p(q)$ as the new state to which the behavior's current state q moves if actions defined in q are finished. In the same way, the messaging function $\mu(q)$ is an outgoing message, which is generated as actions defined in q are carried out. $\square$

State transition may involve some action (or internal process) including message handling (for both incoming and outgoing messages). One of effective representation tool for the proposed behavior model is UML (Unified Modeling Language) *state diagram* (Eriksson and Penker, 1998). Figure 5.8 shows typical transition cases using state diagram notation. In case (a), q_1 is to be a waiting state (i.e., $q_1 \in (Q - \tilde{Q})$). If a newly arrived message m_1 is of a specific message type σ_1 the transition is fired, and some actions follow. The actions could include a message-sending action. The state q_2 can be any state, i.e., $q_2 \in Q$. In case (b), q_3 is to be an action state (i.e., $q_3 \in \tilde{Q}$). If the actions are finished, the transition is fired immediately. In this case, as a result of the action in q_3, the outgoing message m_2 is sent. The following state q_4 can be any type of state. Case (c) is another typical case. Once the actions in the action state q_5 are done, the behavior changes the state to q_6 immediately. Although this transition neither handles any messages nor performs any actions, state change itself has meaning in some cases, for example, other behavior sharing beliefs within an agent can utilize this state of the behavior for some kind of coordination or synchronization purposes. Core behaviors for P-TÂTO are explained in Section 5.6.

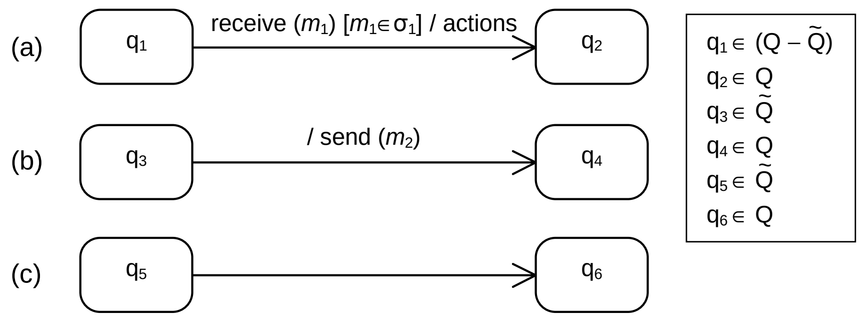


Fig. 5.8. Cases of transition

5.3 Agent Communication

5.3.1 Agent Communication Language: FIPA-ACL

In a MAS, including the DMP resource control system, agents exchange information and knowledge by means of an Agent Communication Language (ACL). ACLs are differentiated from the level of abstraction over other information exchange methods such as RMI (Remote Method Invocation) and CORBA (Common Object Request Broker Architecture) in that they provide the agents with the means of exchanging more complex objects like shared plans, goals and/or strategies. The Knowledge Query and Manipulation Language (KQML), which was a part of the Knowledge Sharing Effort[7], has been the *de facto* standard ACL since its original proposal in 1993. Another ACL developed by the Foundation for Intelligent Physical Agents (FIPA), often called FIPA-ACL (FIPA, 1998), is becoming popular in recent times. However, the main idea and even a big portion of the technical part of FIPA-ACL were inherited from KQML.

An ACL message consists of three layers: (i) the *content layer* contains the actual content of the message, (ii) the *communication layer* encodes message features which describe low-level communication parameters, and (iii) the *message layer* determines the "kind" of interactions one can have with a ACL-speaking agent (Labrou et al., 1999; Finin and Weber, 1993). Figure 5.9 shows an example ACL message, by which the task agent `TA001` inquires about the price of the time slot bundle `F033000T040200` to the resource agent `RA036` using Knowledge Interchange Format (KIF)[8] and an ontology called `DMP_ECONOMY`. The value of the `:content` keyword is the content layer; the values of the `:sender`, `:receiver` and `:reply-with` keywords form the communication layer; and the *performative*[9] name `ask-one` with the `:language` and `:ontology` keywords form the message layer. In FIPA-ACL, performatives are called *communicative acts* (CA).

The CA type in the message layer is the core part of the ACL-based communication protocol because it decides the way of *conversation*[10] between agents. However, the semantics of the CAs may vary depending on applications, meaning that agent designers can decide the *preconditions*, *postconditions* and *completion conditions* for a specific CA

[7]A consortium sponsored by the Advanced Research Projects Agency (ARPA) to develop methodology and software for the sharing and reuse of knowledge.

[8]A computer oriented language for the interchange of knowledge among disparate programs. It is part of the Knowledge Sharing Effort (Genesereth and Fikes, 1992).

[9]In speech act theory, the term *performative* is used to identify the illocutionary force of this special class of utterance (Weiss, 1999). In KQML, however, it means simply any KQML messages (Finin and Weber, 1993).

[10]A series of communications among different agents, typically following a protocol and with some purpose (Weiss, 1999).

```
(ask-one
  :sender TA001
  :content (PRICE F033000T040200 ?price)
  :receiver RA036
  :reply-with price_f033000t040200
  :language KIF
  :ontology DMP_ECONOMY)
```

Fig. 5.9. An example ACL message

by themselves (Labrou and Finin, 1997). Furthermore, the set of CAs itself may be extended if required.

For the DMP resource control economy, we have defined the interactions (in Section 5.6) taking place between agents, which eventually form the market mechanism described in Chapter 4. We map each interaction to an appropriate reserved CA types. For the DMP resource control control economy, only a small number of CA types are used for the core mechanism, including `request`, `agree` and `inform`.

5.3.2 Content Language: FIPA-SL0

Although the CA types define the illocutionary force of the specific messages, the detailed semantics or "contents", are not defined by the ACL. Instead, we need a *content language* (or *inner language*) for detailed description of logics, objects, and actions. FIPA SL (Semantic Language) is the formal language used to define the semantics of the FIPA-ACL. FIPA SL is a general representation formalism that is suitable for use in a number of different agents. FIPA-SL0 is a minimal subset of FIPA SL, and we use FIPA-SL0 as a content language for DMP agent communication. The FIPA-SL0 grammar is given in Appendix B (FIPA, 2000).

5.4 Protocols: Agent Interactions

The P-TÂTO mechanism consists of two different levels of protocols: (1) the temporary economy protocol and (2) the local market evaluation protocol. Within a protocol, interactions between agents must be clearly defined by ACL message passing. The overall sequence of interactions is defined using UML sequence diagrams as shown in Figure 5.10 and Figure 5.11. Also the contents of the messages are briefly explained by types of messages (ACL communicative acts) and contents of the ACL message using FIPA-SL0 as an content language. In this section, the agent interactions, which eventually constitute the P-TÂTO mechanism, are explained focusing on the workflows rather than the detailed meaning of the messages. The detailed ontology of the contents (of ACL messages) is explained in Section 5.5 using ACL and FIPA-SL0.

5.4.1 Temporary Economy Protocol

The temporary economy protocol contains the local market evaluation protocol as a component as shown in Figure 5.10. As explained in Section 4.3.1, a temporary economy can be initiated in many different situations, which are basically significant changes happening in a DMP environment. Whatever the cases are, the RMs are informed of the resource rescheduling needs, and notice the temporary economy-establishment needs to the CO. Temporary economy level interactions (as shown in Figure 5.10) are summarized in Table 5.1 and explained as follows:

- Interaction **a1-a2**: An RM initiates a temporary economy by sending a `request` message **a1** to the CO. This request message asks the CO to establish a temporary economy (`establish-te` action). The CO reviews the temporary economy establishment request, and sends back an `agree` message **a2** to the RM. The CO can refuse the request by sending a `refuse`[11] message **a2′** to the RM. The contents of the `refuse` message are exactly same to the `agree` message.

- Interaction **b1-b2**: The CO establishes a temporary economy by sending a `request` message **b1** to the PMs. This message asks the PM to `select` TAs for the temporary economy. Each PM reviews the request and sends the `agree` message **b2** or `refuse` message **b2′** back to the CO.

[11]The communicative act type `refuse` denotes the action of refusing to perform a given action and explaining the reason for the refusal.

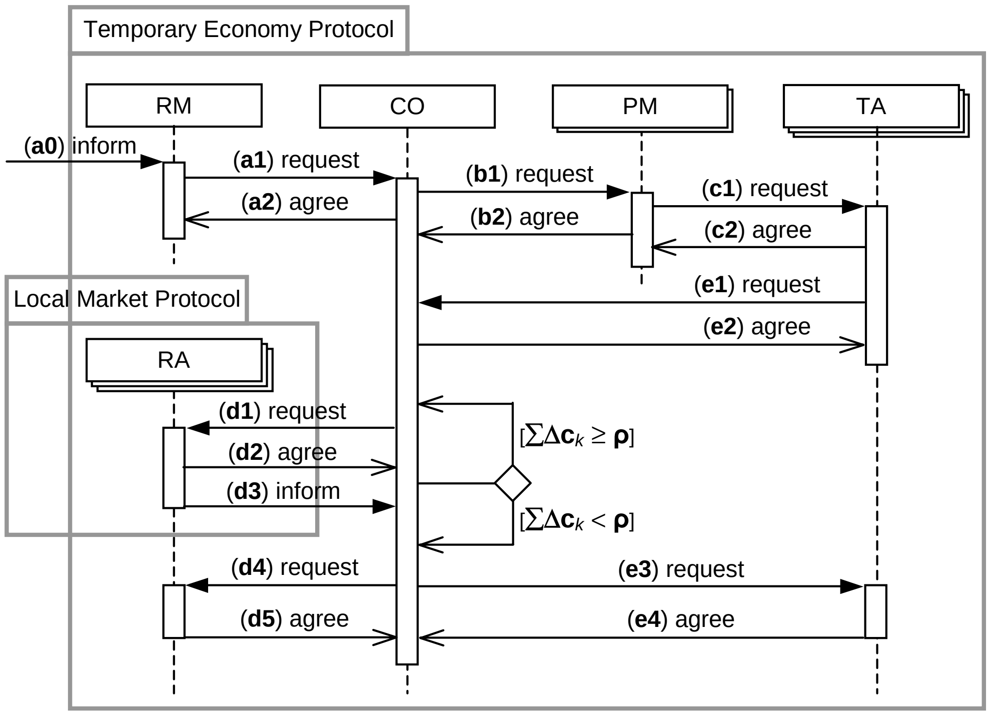


Fig. 5.10. ACL interactions in the temporary economy protocol

- Interaction **c1-c2-e1-e2**: Each PM selects the participants to the temporary economy from its TAs based on the temporary economy information (such as *Control-Window* information), which was enclosed within the `request` message **b1**. The selected TAs get informed via the `request` messages **c1**, which ask the TAs to `attend` the temporary economy. Each of the TAs send an `agree` message **c2** or a `refuse` message **c2′** back to the PM. In addition, each of the TAs sends the CO a `request` message **e1** to ask `register` to the temporary economy. The CO replies with an `agree` message **e2** or `refuse` message **e2′** to the TAs.

- Interaction **d1-d2-d3**: The CO identifies the RAs who must attend the temporary economy based on the TAs' `request` message **e1**. The CO sends each of the RAs a `request` message **d1** to ask them to establish local markets (`establish-lm` action). Each RA sends an `agree` message **d2** or `refuse` message **d2′** back to the CO. After one iterations of local market mechanism, each of the RA reports the aggregate deviation cost vector (*AggDevCostVector*) to the CO via an `inform` message **d3**.

The CO collects all aggregate deviation cost vectors from the RAs, adds up the vectors, and compares the added-up aggregate deviation cost vector $\sum \mathbf{\Delta c}_k$ to the given tolerance vector ρ to determine the clearance of the temporary economy.

- Interaction **d4-d5,e3-e4**: If the vector falls into the predefined tolerance level, the CO sends `request` messages **d1** and **d2** to the RAs and TAs, respectively. These `request` messages ask the RAs and TAs to clear the markets (`clear-lm` action), meaning that they fix the new schedules of the tasks using the current results of local market evaluation mechanism. The RAs and TAs send `agree` messages **d5** and **e4** or `refuse` messages $\mathbf{d5}'$ and $\mathbf{e4}'$ back to the CO.

5.4.2 Local Market Evaluation Protocol

The local market evaluation protocol is an combinatorial auction mechanism as explained in Section 4.3.1. A local market is established by the CO sending an RA a `request`-type ACL message for local market initiation as shown in Figure 5.11. Interactions for a local market evaluation are summarized in Table 5.2 and explained as follows:

- Interaction **f1-c3-c4-f2**: In each local market, the RA sends a `request` message **f1** to the TAs asking them to bid for the resource time slots. A TA (for example, $\texttt{TA}_\texttt{p}$) who received the request prepares the bid. In order to prepare the bid (one component of which is utility for each time-slot bundle), the TA ($\texttt{TA}_\texttt{p}$) sends a `request` message **c3** to its successor TAs (for example, $\texttt{TA}_\texttt{s}$), asking estimated deviation cost information. Upon receipt of the message, the successor TAs ($\texttt{TA}_\texttt{s}$) send an `inform` message **c4** back to $\texttt{TA}_\texttt{p}$. Using this information, $\texttt{TA}_\texttt{p}$ finishes the bid preparation and sends the bid to the RA via an `inform` message **f2**.

- Interaction **f3-c5-c6-f4**: After collecting the bids from TAs for a pre-defined time duration $\texttt{t}^*$, the TR starts calculating the winning bids (the details are explained in Section 4.3.2). The winning bid information (in other words, local optimal schedule with respect to the bid profiles) is sent to the TAs via an `inform` message

Table 5.1. ACL messages in the temporary economy protocol (sequence diagram)

Msg No.	Sender	Receiver	CA type	Content of the message in FIPA-SL0
a1	RM	CO	`request`	`action` *AgentName* (`establish-te` *TE-description*)
a2	CO	RM	`agree`	`establish-te` *TE-description*
a2′	CO	RM	`refuse`	`establish-te` *TE-description*
b1	CO	PM	`request`	`action` *AgentName* (`select` *TE-description*)
b2	PM	CO	`agree`	`select` *TE-description*
b2′	PM	CO	`refuse`	`select` *TE-description*
c1	PM	TA	`request`	`action` *AgentName* (`attend` *TE-description*)
c2	TA	PM	`agree`	`attend` *TE-description*
c2′	TA	PM	`refuse`	`attend` *TE-description*
d1	CO	RA	`request`	`action` *AgentName* (`establish-lm` *LM-description*)
d2	RA	CO	`agree`	`establish-lm` *LM-description*
d2′	RA	CO	`refuse`	`establish-lm` *LM-description*
d3	RA	CO	`inform`	*AggDevCostVector*
d4	CO	RA	`request`	`action` *AgentName* (`clear-lm`)
d5	RA	CO	`agree`	`clear-lm`
d5′	RA	CO	`refuse`	`clear-lm`
e1	TA	CO	`request`	`action` *AgentName* (`register` *TA-description*)
e2	CO	TA	`agree`	`register` *TA-description*
e2′	CO	TA	`refuse`	`register` *TA-description*
e3	CO	TA	`request`	`action` *AgentName* (`clear-lm`)
e4	TA	CO	`agree`	`clear-lm`
e4′	TA	CO	`refuse`	`clear-lm`

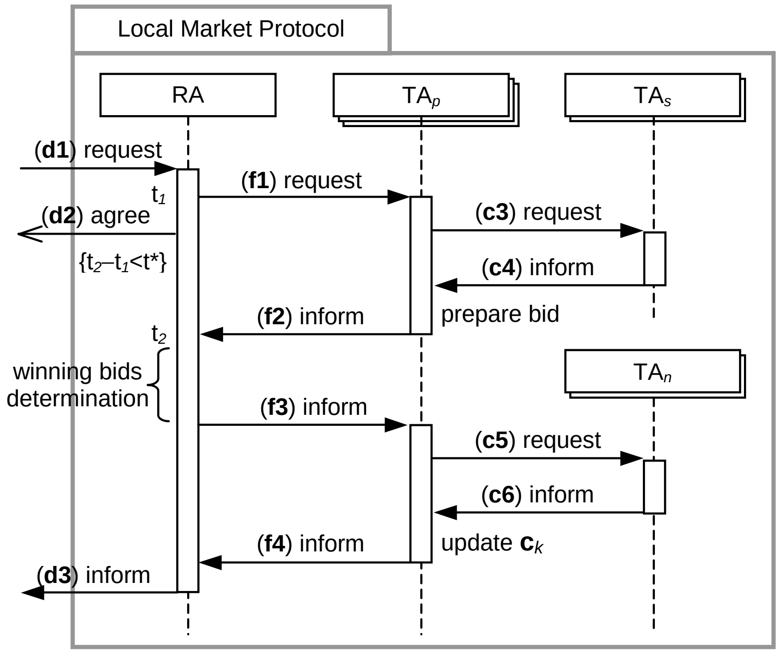


Fig. 5.11. ACL interactions in the local market evaluation protocol (sequence diagram)

Table 5.2. ACL messages in the local market evaluation protocol

Msg No.	Sender	Receiver	CA type	Content of the message in FIPA-SL0
f1	RA	TA	`request`	`action` *AgentName* (`inform bid`)
f2	TA	RA	`inform`	*Bid*
f3	RA	TA	`inform`	*WinningBid*
f4	TA	RA	`inform`	*DevCostVector*
c3	TA_p	TA_s	`request`	`action` *AgentName* (`inform utility-profile`)
c4	TA_s	TA_p	`inform`	*Utility-profile*
c5	TA_p	TA_n	`request`	`action` *AgentName* (`inform task-schedule`)
c6	TA_n	TA_p	`inform`	*TaskSchedule*

f3. The TAs update their *precedence cost vector*. In order to do that, the TAs exchange their new schedule information with their *neighbor* TAs[12] via `request` message **c5** and `inform` message **c6**. The RA adds up the deviation cost vectors of all the TAs and reports the added-up deviation cost vector to the CO via the `inform` message **d3**. The CO determines the clearance of the temporary market based on this information as explained previously.

5.5 DMP Resource Control Ontology

An ontology is a specification of the objects, concepts, and relationships in a specific domain. Regarding the MAS, all agents that share the same ontology for knowledge representation can understand the *content layer* of the ACL messages. Although we already defined some fundamental terminology in Chapter 2 for the formal problem definition, they are not the complete set required for proper operation of P-TÂTO mechanism. In this section, the market-based DMP resource control ontology is defined more precisely.

5.5.1 Grammar

The contents of the ACL messages of the DMP economy consist of two types of components: *actions* and *objects*, as shown in Tables 5.1 and 5.2. The CA `request` and `agree` contain an action within the ACL message, while CA `inform` is an object. We define the representation scheme of the actions and objects using BNF style grammar as follows:

Description of Actions:

```
PtatoAgentAction   =  "(" "establish-te" TE-description ")"
                   |  "(" "establish-lm" LM-description ")"
                   |  "(" "select" TE-description ")"
                   |  "(" "attend" TE-description ")"
```

[12]A pair of *neighbored* TAs are the TAs whose tasks are directly coupled by a precedence constraint.

```
                    |  "(" "clear-lm" ")"
                    |  "(" "register" TA-description ")"
```

Description of Objects:

```
TE-description      =  "(" ":te-description" TE-description-items + ")"
LM-description      =  "(" ":lm-description" LM-description-items + ")"
TA-description      =  "(" ":ta-description" TA-description-items + ")"
TE-description-item =  ControlWindow
                    |  "(" ":slot-size" NumericalConstant ")"
                    |  "(" ":sellers" AgentName + ")"
                    |  "(" ":buyers" AgentName + ")"
LM-description-item =  ControlWindow
                    |  "(" ":slot-size" NumericalConstant ")"
                    |  "(" ":auctioneer" AgentName ")"
                    |  "(" ":bidders" AgentName + ")"
                    |  "(" ":available-slots" Interval + ")"
TA-description-item =  "(" ":agent-id" AgentName ")"
                    |  "(" ":resource-agents" AgentName + ")"
                    |  "(" ":duration" Integer ")"
ControlWindow       =  "(" ":control-window" DateTime DateTime ")"
Bid                 =  "(" ":bid" BidItem + ")"
BidItem             =  "(" Bundle Utility ")"
WinningBid          =  "(" ":winning-bid" BidItem + ")"
Bundle              =  Interval
Interval            =  "(" Integer Integer ")"
TaskSchedule        =  "(" ":task-schedule" Interval ")"
UtilityProfile      =  "(" ":utility-profile" Utility + ")"
DevCostVector       =  "(" ":dev-cost-vector" DevCost + ")"
AggDevCostVector    =  "(" ":agg-dev-cost-vector" DevCost + ")"
DevCost             =  Float
```

```
Utility             =  Float
```

5.5.2 Ontology of Actions

Although the syntectic representation of actions were defined in the previous section, we need to define the meaning and usage of the action contents further. The following tables describe the ontology of the actions including `establish-te`, `establish-lm`, `select`, `attend`, `clear-lm`, and `register`.

Table 5.3. Ontology of action `establish-te`

Description	`establish-te` action involves a new temporary economy establishment. The expected rational effect of `establish-te` action request is activation of the actor's `TemporaryEconomyCoordinatorBehavior` behavior.
Content	`TE-description` (see Section 5.5.1)
Requester	Resource Manager (RM)
Actor	Coordinator Agent (CO)
Protocol	`temporary-economy`

Example	`(request`
	`    :sender rm1-agent@lb233a.ie.psu.edu:1090`
	`    :receiver co-agent@lb233f.ie.psu.edu:1090`
	`    :content`
	`      (action co-agent@lb233a.ie.psu.edu:1090`
	`        (establish-te`
	`          :te-description`
	`            (:control-window 2400 2480)`
	`            (:slot-size 1)))`
	`    :language SL0`
	`    :protocol temporary-economy`
	`    :ontology dmp-resource-control)`

Table 5.4. Ontology of action `establish-lm`

Description	`establish-lm` action involves a new local market evaluation. The expected rational effect of `establish-lm` action request is activation of the actor's `LocalMarketAuctioneerBehavior` behavior.
Content	`LM-description` (see Section 5.5.1)
Requester	Coordinator Agent (CO)
Actor	Resource Agent (RA)
Protocol	`temporary-economy`

Example	`(request`
	    `:sender co-agent@lb233f.ie.psu.edu:1090`
	    `:receiver ra1-agent@lb233a.ie.psu.edu:1090`
	    `:content`
	      `(action ra1-agent@lb233a.ie.psu.edu:1090`
	        `(establish-lm`
	          `:lm-description`
	            `(:control-window 2400 2480)`
	            `(:slot-size 1)`
	            `(:auctioneer ra1-agent@lb233a.ie.psu.edu:1090)`
	            `(:bidders ta1-agent@lb233c.ie.psu.edu:1090`
	                        `ta2-agent@lb233d.ie.psu.edu:1090`
	                        `ta3-agent@lb233e.ie.psu.edu:1090)))`
	    `:language SL0`
	    `:protocol temporary-economy`
	    `:ontology dmp-resource-control)`

Table 5.5. Ontology of action `select`

Description	`select` action involves a new temporary economy establishment by selecting adequate TAs who are supposed to be affected by resource rescheduling. The expected rational effect of the `select` action request is to make the actor send request messages to the selected TAs asking them to register for the temporary economy.
Content	`TE-description` (see Section 5.5.1)
Requester	Coordinator Agent (CO)
Actor	Project Manager (PM)
Protocol	`temporary-economy`

Example	`(request`
	`        :sender co-agent@lb233f.ie.psu.edu:1090`
	`        :receiver pm1-agent@lb233b.ie.psu.edu:1090`
	`        :content`
	`           (action pm1-agent@lb233b.ie.psu.edu:1090`
	`              (select`
	`                 :te-description`
	`                    (:control-window 2400 2480)`
	`                    (:slot-size 1)))`
	`        :language SL0`
	`        :protocol temporary-economy`
	`        :ontology dmp-resource-control)`

Table 5.6. Ontology of action `attend`

Description	`attend` action involves a new temporary economy establishment by asking adequate TAs to register for the temporary economy. The expected rational effect of `attend` action request is registration of the actor to the temporary economy as a buyer of resource time-slots.
Content	`TE-description` (see Section 5.5.1)
Requester	Project Manager (PM)
Actor	Task Agent (TA)
Protocol	`temporary-economy`

Example	`(request`
	`        :sender pm1-agent@lb233b.ie.psu.edu:1090`
	`        :receiver ta1-agent@lb233c.ie.psu.edu:1090`
	`                  ta2-agent@lb233d.ie.psu.edu:1090`
	`                  ta3-agent@lb233e.ie.psu.edu:1090`
	`        :content`
	`          (action co-agent@lb233a.ie.psu.edu:1090`
	`            (attend`
	`              :te-description`
	`                (:control-window 2400 2480)`
	`                (:slot-size 1)))`
	`        :language SL0`
	`        :protocol temporary-economy`
	`        :ontology dmp-resource-control)`

Table 5.7. Ontology of action `register`

Description	`register` action involves a new temporary economy establishment. The expected rational effects of `register` action request are (1) to allow the requester to participate in the temporary economy as buyers of the resource time-slots, and (2) to let the actor informed of the possible sellers (RAs) based on the message contents (`TA-description`).
Content	`TA-description` (see Section 5.5.1)
Requester	Task Agent (TA)
Actor	Coordinator Agent (CO)
Protocol	`temporary-economy`

Example	`(request`
	`    :sender ta1-agent@lb233c.ie.psu.edu:1090`
	`    :receiver co-agent@lb233f.ie.psu.edu:1090`
	`    :content`
	`     (action co-agent@lb233f.ie.psu.edu:1090`
	`      (register`
	`       :ta-description`
	`        (:agent-id ta1-agent@lb233c.ie.psu.edu:1090)`
	`        (:resource-agent ra1-agent@lb233a.ie.psu.edu:1090)`
	`        (:duration 8)))`
	`    :language SL0`
	`    :protocol temporary-economy`
	`    :ontology dmp-resource-control)`

Table 5.8. Ontology of action `clear-lm`

Description	`clear-lm` action involves a clearance of a local market. The expected rational effect of `clear-lm` action request is finalization of the resource and task schedule using the current schedule updated during the recent local market evaluation.
Content	None
Requester	Coordinator Agent (CO)
Actor	Resource Agent (RA) and Task Agent (TA)
Protocol	`temporary-economy`

Example	`(request`
	`    :sender co-agent@lb233f.ie.psu.edu:1090`
	`    :receiver ra1-agent@lb233a.ie.psu.edu:1090`
	`    :content`
	`      (action ra1-agent@lb233a.ie.psu.edu:1090`
	`        (clear-lm))`
	`    :language SL0`
	`    :protocol temporary-economy`
	`    :ontology dmp-resource-control)`

5.6 Behaviors for the P-TÂTO Mechanism

As mentioned in Section 5.2, an agent can have multiple behaviors and a set of interrelated behaviors held by multiple collaborative agents constitutes a decentralized mechanism like P-TÂTO. In this section, we present the details of the core behaviors defined in P-TÂTO mechanism. As an augmented finite automaton, the behaviors can be clearly defined by state diagrams as explained in Section 5.2.3.

5.6.1 `TemporaryEconomyInitiatorBehavior`

The behavior `TemporaryEconomyInitiatorBehavior` (see Figure 5.12) is held by Resource Manager (RM) agents. The states q_1, q_2 and q_3 are all waiting state (i.e., $\{q_1, q_2, q_3\} \subset \tilde{Q}$). State q_1 ("waiting change reports") is the initial state of the behavior. If a message of a specific type (i.e., σ_1, which is a significant change `inform` message) is received from a reliable sender, the behavior sends a `request` message (m_2) to the CO asking temporary economy establishment and changes the state to q_2 ("waiting a response"). If an `agree` message is received from the CO, the state is changed to q_3 ("waiting market clearance") and waiting the market-clearance `inform` message ($m_7 \in \sigma_3$) from the CO later on. However, if the request is `refuse`d (via $m_5 \in \sigma_3$) or not replied for a given maximum waiting time, the state is changed back to q_1. While the behavior is in the states of q_2 and q_3, another new message of type σ_1 cannot be processed, thereby being replied via a `refuse` message.

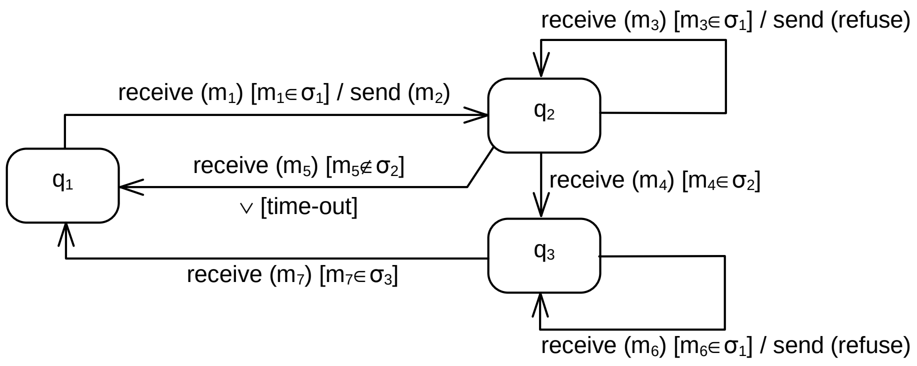

receive (m3) [m3∈σ1] / send (refuse)
receive (m1) [m1∈σ1] / send (m2)
q2
q1
receive (m5) [m5∉σ2]
∨ [time-out]
receive (m4) [m4∈σ2]
q3
receive (m7) [m7∈σ3]
receive (m6) [m6∈σ1] / send (refuse)

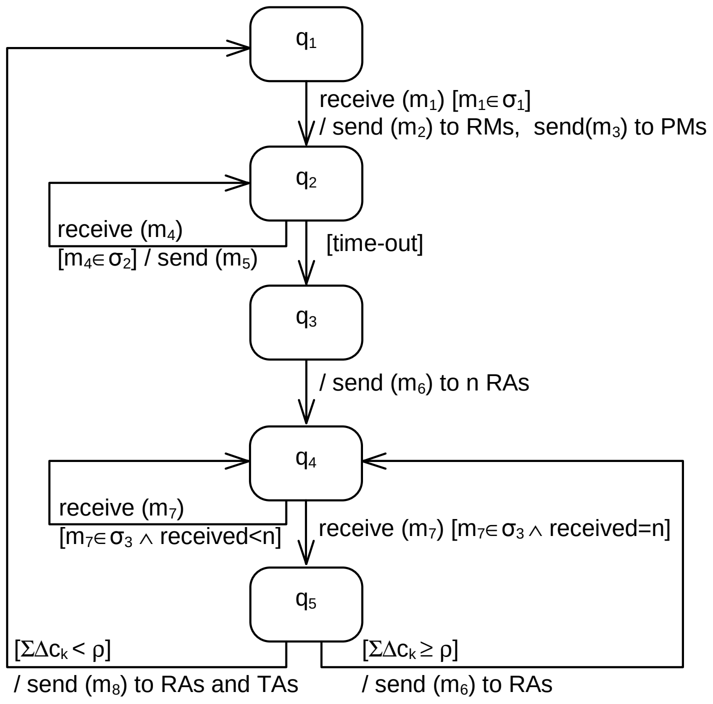
Fig. 5.12. `TemporaryEconomyInitiatorBehavior` (state diagram)

5.6.2 `TemporaryEconomyCoordinatorBehavior`

This behavior is a coordinator agent (CO)'s behavior, which coordinates the overall workflow of P-TÂTO mechanism. In the initial state q_1 ("waiting temporary economy initiation"), if the behavior receives temporary economy establishment `request` message (m_1), the behavior sends an `agree` message (m_2) back to the sender (an RM) and sends a `request` message (m_3) to the PMs asking them to make adequate TAs to attend the temporary economy. Then, the behavior changes its state to q_2 ("waiting registration request") and starts receiving registration `request` messages (m_4) from TAs. When a new registration request message received, an `agree` message (m_5) is sent back to the TA. After a given period of waiting time, the behavior changes the state to q_3 ("preparing temporary economy"). In the action state, the behavior finalizes the list of RAs and TAs, who participate in the temporary economy, then sends local market evaluation request messages (m_6) to the RAs and changes the state to q_4 ("waiting local market evaluation"). When all the RAs finish the local market evaluation and send the precedence cost change information, the state is changed to another action state q_5 in which the behavior decides if the market must be cleared or not. Depending on the decision, the state is changed back to the initial state q_1 or to q_4 for another iteration of local market evaluation.

5.6.3 `LocalMarketAuctioneerBehavior`

This behavior is held by a resource agent (RA). As shown in Figure 5.14 this behavior has 4 states, among which q_3 is an action state ($q_3 \in \tilde{Q}$) and others are all waiting states. This behavior starts from q_1 ("waiting local market-evaluation request"). If a local market evaluation `request` message ($m_1 \in \sigma_1$) is received from the CO, request-for-bid messages (m_2) are sent to TAs, who are listed in the *LM-description* enclosed in the message m_1, and the state is changed to q_2 ("waiting bids") immediately. In state q_2, the behavior collects bids from the TAs until all the TAs bid or time elapses a predefined waiting time. In both cases, the state is changed to q_3 ("calculating winning bids"), in which the time-consuming winning bids calculation is carried out. Upon completion of the process, winning-bids information is sent to the TAs via message m_4, and the state



q_1
receive (m_1) [$m_1 \in \sigma_1$]
/ send (m_2) to RMs, send(m_3) to PMs
q_2
receive (m_4)
[$m_4 \in \sigma_2$] / send (m_5)
[time-out]
q_3
/ send (m_6) to n RAs
q_4
receive (m_7)
[$m_7 \in \sigma_3 \wedge$ received<n]
receive (m_7) [$m_7 \in \sigma_3 \wedge$ received=n]
q_5
[$\Sigma\Delta c_k < \rho$]
/ send (m_8) to RAs and TAs
[$\Sigma\Delta c_k \geq \rho$]
/ send (m_6) to RAs

Fig. 5.13. `TemporaryEconomyCoordinatorBehavior` (state diagram) is changed to q_4 ("waiting precedence cost vector information") immediately. In q_4 the behavior collects the updated precedence cost vectors from TAs, until all the TAs reply to the request or time is over. The behavior then adds up the vectors and sends it to the CO (via message m_6), and changes the state back to q_1, the initial state.

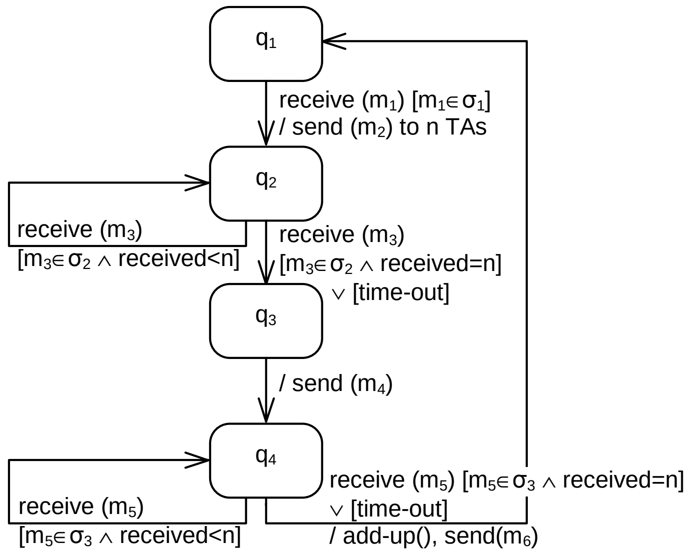


Fig. 5.14. `LocalMarketAuctioneerBehavior` (state diagram)

5.6.4 `LocalMarketBidderBehavior`

This behavior, which is held by a task agent (TA), starts from q_1 ("waiting request-for-bid") as shown in Figure 5.15. If received a request-for-bid message message ($m_1 \in \sigma_1$) from an RA, this behavior inquires utility profile information from its successor TAs (TA$_s$'s) via message m_2, and changes the state to q_2 ("waiting bid profile information") immediately. In state q_2, the behavior collects the successor TAs' utility profiles until all the TA$_s$'s reply or time elapses a predefined waiting time. In both cases, the state is changed to q_3 ("preparing bid"), in which the bid is prepared based on the current precedence cost vector and the successor TAs' utility profiles. Upon completion of the process, the behavior sends the bid to the RA via message m_4, changes the state to q_4 ("waiting winning bids information"), and waits until the winning bid information is available from the RA (via message m_5). When the winning bid information is received, the behavior sends second inquiry message (m_6) to the neighbor TAs (TA$_n$'s) asking up-to-date task schedule information (*TaskSchedule*). The behavior uses the collected information to update the precedence cost vector (see Section 4.4.5), sends the precedence cost vector to the RA, and changes the state back to the initial state q_1.

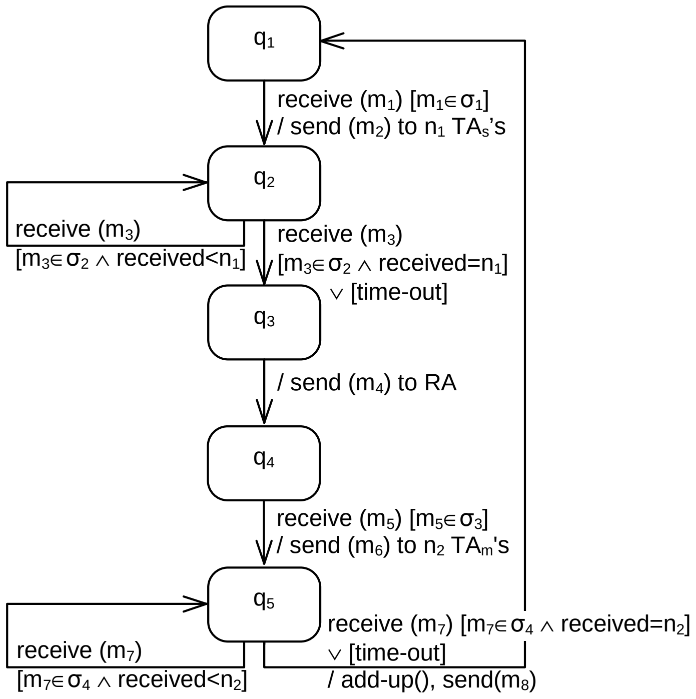

q_1
receive (m_1) [$m_1 \in \sigma_1$]
/ send (m_2) to n_1 TA$_s$'s
q_2
receive (m_3)
[$m_3 \in \sigma_2 \wedge$ received<n_1]
receive (m_3)
[$m_3 \in \sigma_2 \wedge$ received=n_1]
$\vee$ [time-out]
q_3
/ send (m_4) to RA
q_4
receive (m_5) [$m_5 \in \sigma_3$]
/ send (m_6) to n_2 TA$_m$'s
q_5
receive (m_7)
[$m_7 \in \sigma_4 \wedge$ received<n_2]
receive (m_7) [$m_7 \in \sigma_4 \wedge$ received=n_2]
$\vee$ [time-out]
/ add-up(), send(m_8)

Fig. 5.15. `LocalMarketBidderBehavior` (state diagram)

5.6.5 Other Behaviors

Other than the core behaviors explained above, some simple behaviors are required for completeness of the mechanism. An example is `LocalMarketClearingBehavior`, which is to clear a local market (just by finalizing the current resource/task schedule) based upon a market clearing request from the CO. This behavior is held by both of TAs and RAs, but with different implementations of internal actions. Another example is `InquiryResponseBehavior`, which is to respond to some information inquiries, such as the messages m_2 and m_6 in `LocalMarketBidderBehavior` explained in Section 5.6.4. Such simple behaviors are typically in the form of the state diagram in Figure 5.16.

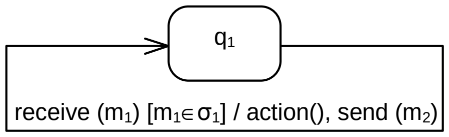


Fig. 5.16. `Simple response behavior` (state diagram)

5.7 Implementation

The programming language of choice is Java because of its powerful object oriented programming features in distributed heterogeneous computing environments, including object serialization, multi-threading, reflection API and remote method invocation (RMI). There are many publicly available open-source Java packages for agent developers to use; however, due to the diversity of designs in reasoning system in various agent application areas, these Java packages generally support for developing *communicating agents*, meaning that the *intelligence* should be coded by developers themselves. We use JADE (Java Agent DEvelopment Framework), which follows FIPA agent management specification and FIPA-ACL compliant. It was developed using Java and easily integrated with the JESS (Java Expert System Shell)(Friedman-Hill, 2000), so that a rule-based reasoning system can be integrated in a relatively easy way (Bellifemine et al., 1999). Some other well-known open-source Java packages are as follows:

- Java Agent Template, Lite (JATLite) is a package of Java programs, developed at Stanford University, that allow users to create communicating agents quickly. Each agent runs as an applet launched from a Web browser. All agents communicate through a message router facilitator (Jeon et al., 2000).
- Java-based Agent Framework for Multiagent Systems (JAFMAS) is a set of Java classes that supports implementing communicating agents, developed at

the University of Cincinnati. It supports directed (point-to-point) communication as well as subject-based, broadcast communications (Chauhan and Baker, 1998).

- Jackal is another Java package that allows Java applications to communicate via KQML (Cost et al., 1998), developed at University of Maryland, Baltimore County. It is currently used in the CIIMPLEX project (Peng et al., 1998), a project that involves planning and scheduling for manufacturing.

- Zeus is a Java toolkit for the rapid development of collaborative agent applications using GUI-based development environment, developed at British Telecommunication Laboratories (BT labs). It is FIPA-ACL compliant and has been used to implement several major applications (Nwana et al., 1998).

Figure 5.17 shows the major components of the `dmp` package and relationships with the components in `jade.core` package.

5.8 Summary

In this chapter we presented detailed design of the individual DMP agent model and implementation of the market mechanism within the agent model. Many different agent architectures have been introduced in the agent research literature; however, there is no perfect architecture, which is suitable for all situations. Instead, different problem domains require different levels of agency and different design of internal architectures. We reviewed several representative architecture model to identify the desirable architecture of DMP agents. Basically DMP agents are categorized into collaborative agent (explained in Section 5.1.5) with reactive behaviors. Three major characteristics of DMP agents are *behavior-oriented*, *state-based*, and *message-driven*. Agent communication and management of DMP multiagent system follows the FIPA agent communication and management specification. The behavior model is clearly specified as an augmented finite automaton, which has two distinctive state types: *waiting state* and *action state*. The possible transitions among states in different situations were identified, and corresponding UML state diagram notations were proposed.

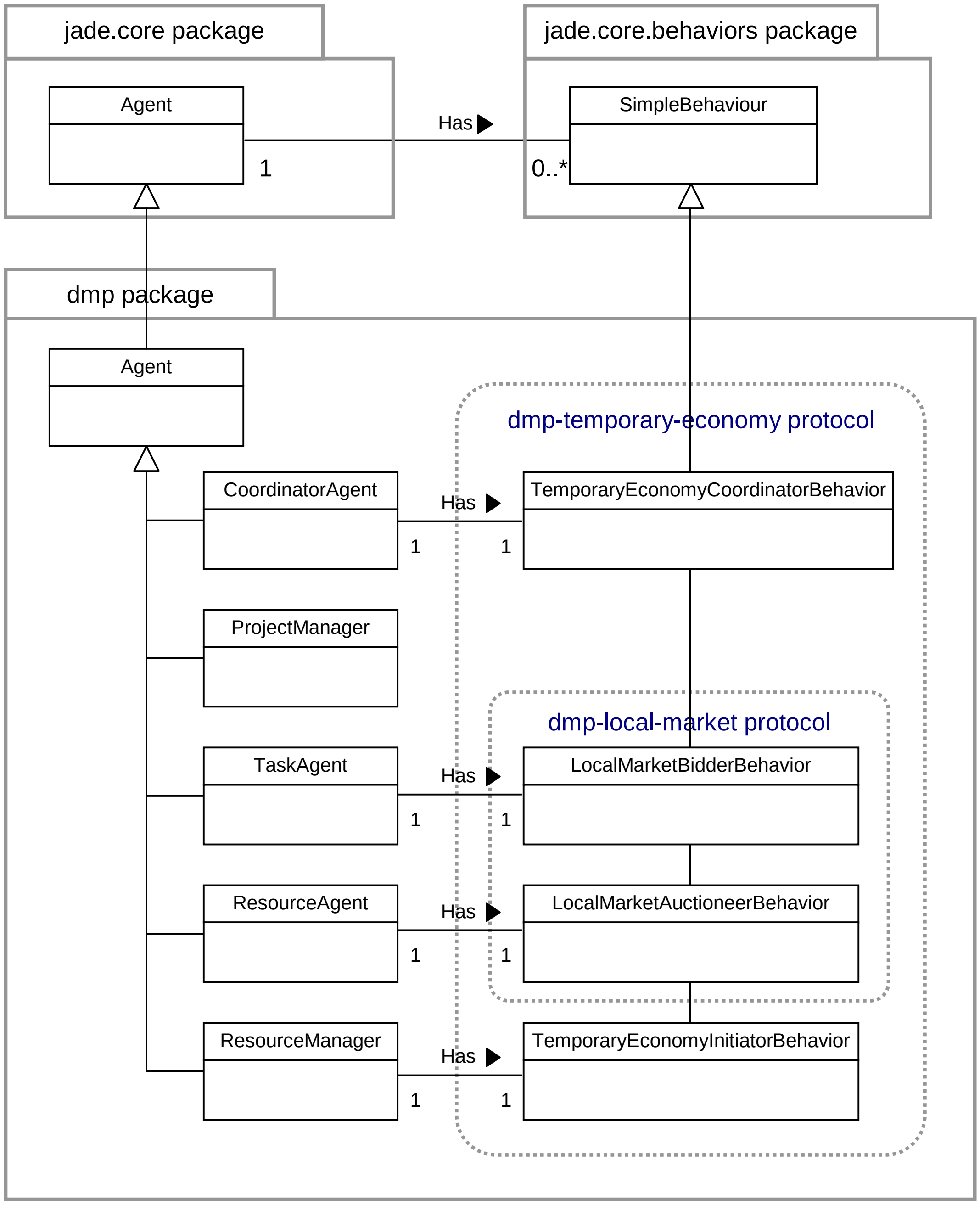

jade.core package
Agent
1
Has
0..*
jade.core.behaviors package
SimpleBehaviour
dmp package
Agent
dmp-temporary-economy protocol
CoordinatorAgent
Has
1
1
TemporaryEconomyCoordinatorBehavior
ProjectManager
dmp-local-market protocol
TaskAgent
Has
1
1
LocalMarketBidderBehavior
ResourceAgent
Has
1
1
LocalMarketAuctioneerBehavior
ResourceManager
Has
1
1
TemporaryEconomyInitiatorBehavior

Fig. 5.17. Components of the `dmp` package

In addition to the agent model, the P-TÂTO mechanism was clearly defined from the multiagent information-systems' viewpoint. A distributed mechanism can be embodied within multiple agents in the form of behaviors. The mechanism was thoroughly defined using UML sequence diagrams. Also, using the FIPA-ACL (as an agent communication language), FIPA-SL0 (as a content language), and DMP resource control ontology, the detailed interactions among agents were explained. Finally, the core behaviors were clearly defined using state diagrams.

Chapter 6

Empirical Analysis

Experimental analysis and verification of the proposed approach is one of the most important parts of this research. However, it is very difficult to test and verify the proposed approach in real DMP environments due to their huge scale and variety of organization and project characteristics. Hence, intensive testing and analysis is conducted in an experimental setting using a wide range of DMP instances generated by a project instance generator called ProGen. We developed a simulation software, which emulate the P-TÂTO mechanism to conduct the experimental analysis. In this chapter, we present the experimental analysis results followed by the experimental setup and DMP instance generation issues. The experimental analysis results demonstrate a high level of solution quality and computational efficiency.

6.1 Experimental Setup

6.1.1 Simulation Programs: PtatoSim

We develop and use a simulation program, called *PtatoSim*, for an extensive experimental analysis of the P-TÂTO mechanism. This program uses DMP instances (`*.dat`) generated by *ProGen* (explained in the following subsection) as an input, carries out the simulation, and generates simulation outputs: log file (`*.out`), final schedule (`*.sch`), and LINDO input model (`*.ltx`) as shown Figure 6.1. The file `*.bas` is the input file for ProGen, which defines the different configuration parameters for project instances to be generated. The output of ProGen (`*.dat` files) are randomly (yet controlled by the input parameters) generated project scheduling problem instances. Each `*.dat` file fully defines a project scheduling problem in a ProGen-genuine format. The simulation log file (`*.out`) contains detailed simulation log including each agents utility profiles, bidding information, winning bid determination process and results, and precedence cost vector for every iteration of P-TÂTO mechanism. The final schedule file (*.sch) contains a resultant schedule of each resource, which can be used for drawing Gantt charts automatically by PtatoSim. Finally, the `*.ltx` file contains an IP formulation of the DMP instance in a LINDO input model format, which is based on the centralized IP formulation explained in Section 2.2. This LINDO input model is used for generating optimal solutions to be compared to P-TÂTO solutions (see Section 6.3). Figure 6.2 shows an example of LINDO input generated from a small DMP instance.

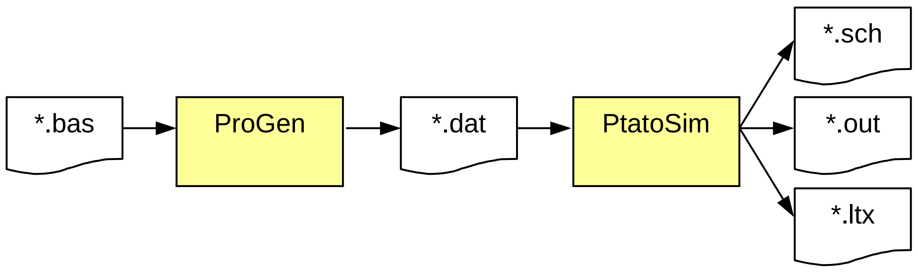
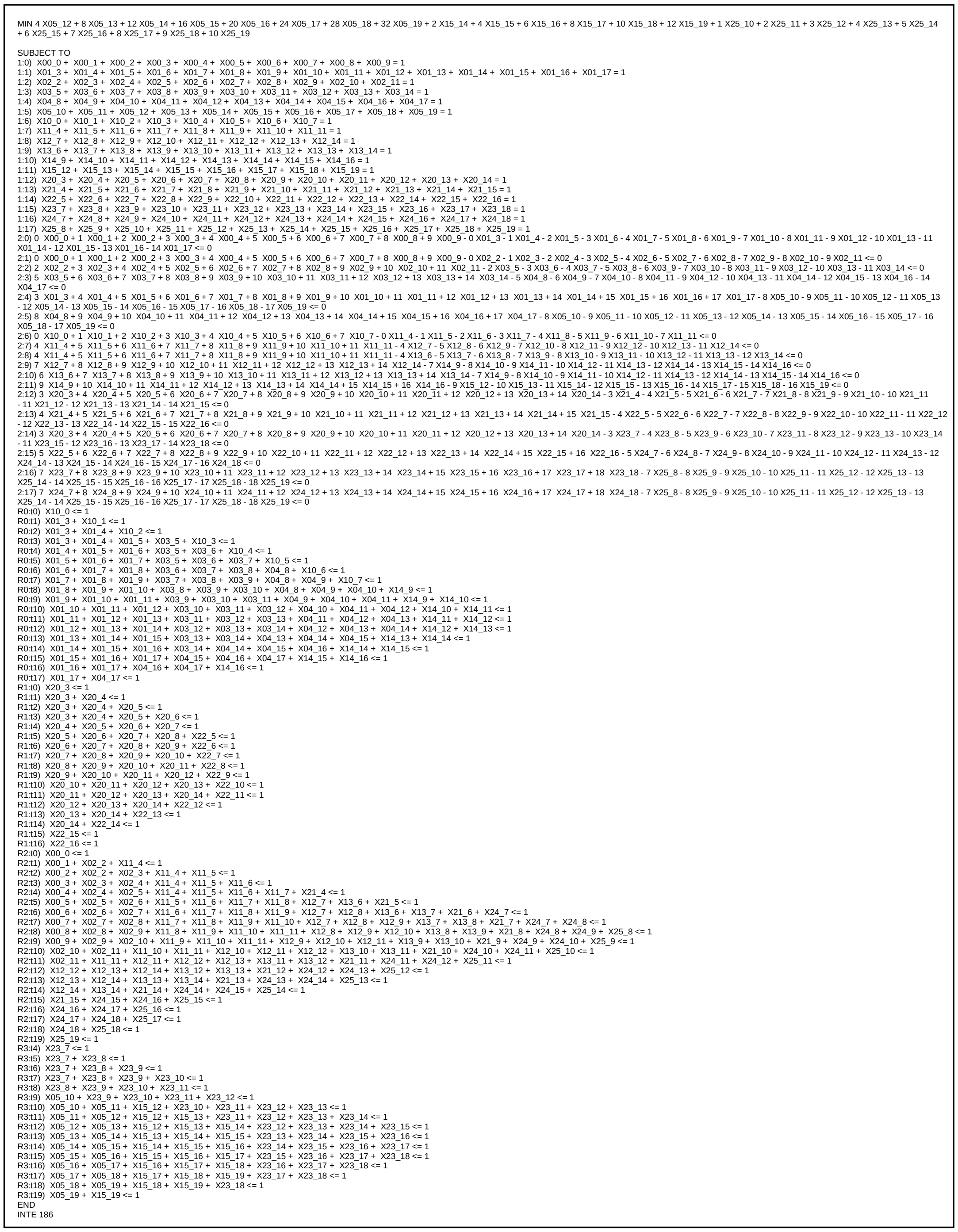


Fig. 6.1. Overview of PtatoSim



```
MIN 4 X05_12 + 8 X05_13 + 12 X05_14 + 16 X05_15 + 20 X05_16 + 24 X05_17 + 28 X05_18 + 32 X05_19 + 2 X15_14 + 4 X15_15 + 6 X15_16 + 8 X15_17 + 10 X15_18 + 12 X15_19 + 1 X25_10 + 2 X25_11 + 3 X25_12 + 4 X25_13 + 5 X25_14
+ 6 X25_15 + 7 X25_16 + 8 X25_17 + 9 X25_18 + 10 X25_19

SUBJECT TO
1:0)  X00_0 +  X00_1 +  X00_2 +  X00_3 +  X00_4 +  X00_5 +  X00_6 +  X00_7 +  X00_8 +  X00_9 = 1
1:1)  X01_3 +  X01_4 +  X01_5 +  X01_6 +  X01_7 +  X01_8 +  X01_9 +  X01_10 +  X01_11 +  X01_12 +  X01_13 +  X01_14 +  X01_15 +  X01_16 +  X01_17 = 1
1:2)  X02_2 +  X02_3 +  X02_4 +  X02_5 +  X02_6 +  X02_7 +  X02_8 +  X02_9 +  X02_10 +  X02_11 = 1
1:3)  X03_5 +  X03_6 +  X03_7 +  X03_8 +  X03_9 +  X03_10 +  X03_11 +  X03_12 +  X03_13 +  X03_14 = 1
1:4)  X04_8 +  X04_9 +  X04_10 +  X04_11 +  X04_12 +  X04_13 +  X04_14 +  X04_15 +  X04_16 +  X04_17 = 1
1:5)  X05_10 +  X05_11 +  X05_12 +  X05_13 +  X05_14 +  X05_15 +  X05_16 +  X05_17 +  X05_18 +  X05_19 = 1
1:6)  X10_0 +  X10_1 +  X10_2 +  X10_3 +  X10_4 +  X10_5 +  X10_6 +  X10_7 = 1
1:7)  X11_4 +  X11_5 +  X11_6 +  X11_7 +  X11_8 +  X11_9 +  X11_10 +  X11_11 = 1
1:8)  X12_7 +  X12_8 +  X12_9 +  X12_10 +  X12_11 +  X12_12 +  X12_13 +  X12_14 = 1
1:9)  X13_6 +  X13_7 +  X13_8 +  X13_9 +  X13_10 +  X13_11 +  X13_12 +  X13_13 +  X13_14 = 1
1:10)  X14_9 +  X14_10 +  X14_11 +  X14_12 +  X14_13 +  X14_14 +  X14_15 +  X14_16 = 1
1:11)  X15_12 +  X15_13 +  X15_14 +  X15_15 +  X15_16 +  X15_17 +  X15_18 +  X15_19 = 1
1:12)  X20_3 +  X20_4 +  X20_5 +  X20_6 +  X20_7 +  X20_8 +  X20_9 +  X20_10 +  X20_11 +  X20_12 +  X20_13 +  X20_14 = 1
1:13)  X21_4 +  X21_5 +  X21_6 +  X21_7 +  X21_8 +  X21_9 +  X21_10 +  X21_11 +  X21_12 +  X21_13 +  X21_14 +  X21_15 = 1
1:14)  X22_5 +  X22_6 +  X22_7 +  X22_8 +  X22_9 +  X22_10 +  X22_11 +  X22_12 +  X22_13 +  X22_14 +  X22_15 +  X22_16 = 1
1:15)  X23_7 +  X23_8 +  X23_9 +  X23_10 +  X23_11 +  X23_12 +  X23_13 +  X23_14 +  X23_15 +  X23_16 +  X23_17 +  X23_18 = 1
1:16)  X24_7 +  X24_8 +  X24_9 +  X24_10 +  X24_11 +  X24_12 +  X24_13 +  X24_14 +  X24_15 +  X24_16 +  X24_17 +  X24_18 = 1
1:17)  X25_8 +  X25_9 +  X25_10 +  X25_11 +  X25_12 +  X25_13 +  X25_14 +  X25_15 +  X25_16 +  X25_17 +  X25_18 +  X25_19 = 1
2:0) 0  X00_0 + 1  X00_1 + 2  X00_2 + 3  X00_3 + 4  X00_4 + 5  X00_5 + 6  X00_6 + 7  X00_7 + 8  X00_8 + 9  X00_9 - 0 X01_3 - 1 X01_4 - 2 X01_5 - 3 X01_6 - 4 X01_7 - 5 X01_8 - 6 X01_9 - 7 X01_10 - 8 X01_11 - 9 X01_12 - 10 X01_13 - 11
X01_14 - 12 X01_15 - 13 X01_16 - 14 X01_17 <= 0
2:1) 0  X00_0 + 1  X00_1 + 2  X00_2 + 3  X00_3 + 4  X00_4 + 5  X00_5 + 6  X00_6 + 7  X00_7 + 8  X00_8 + 9  X00_9 - 0 X02_2 - 1 X02_3 - 2 X02_4 - 3 X02_5 - 4 X02_6 - 5 X02_7 - 6 X02_8 - 7 X02_9 - 8 X02_10 - 9 X02_11 <= 0
2:2) 2  X02_2 + 3  X02_3 + 4  X02_4 + 5  X02_5 + 6  X02_6 + 7  X02_7 + 8  X02_8 + 9  X02_9 + 10  X02_10 + 11  X02_11 - 2 X03_5 - 3 X03_6 - 4 X03_7 - 5 X03_8 - 6 X03_9 - 7 X03_10 - 8 X03_11 - 9 X03_12 - 10 X03_13 - 11 X03_14 <= 0
2:3) 5  X03_5 + 6  X03_6 + 7  X03_7 + 8  X03_8 + 9  X03_9 + 10  X03_10 + 11  X03_11 + 12  X03_12 + 13  X03_13 + 14  X03_14 - 5 X04_8 - 6 X04_9 - 7 X04_10 - 8 X04_11 - 9 X04_12 - 10 X04_13 - 11 X04_14 - 12 X04_15 - 13 X04_16 - 14
X04_17 <= 0
2:4) 3  X01_3 + 4  X01_4 + 5  X01_5 + 6  X01_6 + 7  X01_7 + 8  X01_8 + 9  X01_9 + 10  X01_10 + 11  X01_11 + 12  X01_12 + 13  X01_13 + 14  X01_14 + 15  X01_15 + 16  X01_16 + 17  X01_17 - 8 X05_10 - 9 X05_11 - 10 X05_12 - 11 X05_13
- 12 X05_14 - 13 X05_15 - 14 X05_16 - 15 X05_17 - 16 X05_18 - 17 X05_19 <= 0
2:5) 8  X04_8 + 9  X04_9 + 10  X04_10 + 11  X04_11 + 12  X04_12 + 13  X04_13 + 14  X04_14 + 15  X04_15 + 16  X04_16 + 17  X04_17 - 8 X05_10 - 9 X05_11 - 10 X05_12 - 11 X05_13 - 12 X05_14 - 13 X05_15 - 14 X05_16 - 15 X05_17 - 16
X05_18 - 17 X05_19 <= 0
2:6) 0  X10_0 + 1  X10_1 + 2  X10_2 + 3  X10_3 + 4  X10_4 + 5  X10_5 + 6  X10_6 + 7  X10_7 - 0 X11_4 - 1 X11_5 - 2 X11_6 - 3 X11_7 - 4 X11_8 - 5 X11_9 - 6 X11_10 - 7 X11_11 <= 0
2:7) 4  X11_4 + 5  X11_5 + 6  X11_6 + 7  X11_7 + 8  X11_8 + 9  X11_9 + 10  X11_10 + 11  X11_11 - 4 X12_7 - 5 X12_8 - 6 X12_9 - 7 X12_10 - 8 X12_11 - 9 X12_12 - 10 X12_13 - 11 X12_14 <= 0
2:8) 4  X11_4 + 5  X11_5 + 6  X11_6 + 7  X11_7 + 8  X11_8 + 9  X11_9 + 10  X11_10 + 11  X11_11 - 4 X13_6 - 5 X13_7 - 6 X13_8 - 7 X13_9 - 8 X13_10 - 9 X13_11 - 10 X13_12 - 11 X13_13 - 12 X13_14 <= 0
2:9) 7  X12_7 + 8  X12_8 + 9  X12_9 + 10  X12_10 + 11  X12_11 + 12  X12_12 + 13  X12_13 + 14  X12_14 - 7 X14_9 - 8 X14_10 - 9 X14_11 - 10 X14_12 - 11 X14_13 - 12 X14_14 - 13 X14_15 - 14 X14_16 <= 0
2:10) 6  X13_6 + 7  X13_7 + 8  X13_8 + 9  X13_9 + 10  X13_10 + 11  X13_11 + 12  X13_12 + 13  X13_13 + 14  X13_14 - 7 X14_9 - 8 X14_10 - 9 X14_11 - 10 X14_12 - 11 X14_13 - 12 X14_14 - 13 X14_15 - 14 X14_16 <= 0
2:11) 9  X14_9 + 10  X14_10 + 11  X14_11 + 12  X14_12 + 13  X14_13 + 14  X14_14 + 15  X14_15 + 16  X14_16 - 9 X15_12 - 10 X15_13 - 11 X15_14 - 12 X15_15 - 13 X15_16 - 14 X15_17 - 15 X15_18 - 16 X15_19 <= 0
2:12) 3  X20_3 + 4  X20_4 + 5  X20_5 + 6  X20_6 + 7  X20_7 + 8  X20_8 + 9  X20_9 + 10  X20_10 + 11  X20_11 + 12  X20_12 + 13  X20_13 + 14  X20_14 - 3 X21_4 - 4 X21_5 - 5 X21_6 - 6 X21_7 - 7 X21_8 - 8 X21_9 - 9 X21_10 - 10 X21_11
- 11 X21_12 - 12 X21_13 - 13 X21_14 - 14 X21_15 <= 0
2:13) 4  X21_4 + 5  X21_5 + 6  X21_6 + 7  X21_7 + 8  X21_8 + 9  X21_9 + 10  X21_10 + 11  X21_11 + 12  X21_12 + 13  X21_13 + 14  X21_14 + 15  X21_15 - 4 X22_5 - 5 X22_6 - 6 X22_7 - 7 X22_8 - 8 X22_9 - 9 X22_10 - 10 X22_11 - 11 X22_12
- 12 X22_13 - 13 X22_14 - 14 X22_15 - 15 X22_16 <= 0
2:14) 3  X20_3 + 4  X20_4 + 5  X20_5 + 6  X20_6 + 7  X20_7 + 8  X20_8 + 9  X20_9 + 10  X20_10 + 11  X20_11 + 12  X20_12 + 13  X20_13 + 14  X20_14 - 3 X23_7 - 4 X23_8 - 5 X23_9 - 6 X23_10 - 7 X23_11 - 8 X23_12 - 9 X23_13 - 10 X23_14
- 11 X23_15 - 12 X23_16 - 13 X23_17 - 14 X23_18 <= 0
2:15) 5  X22_5 + 6  X22_6 + 7  X22_7 + 8  X22_8 + 9  X22_9 + 10  X22_10 + 11  X22_11 + 12  X22_12 + 13  X22_13 + 14  X22_14 + 15  X22_15 + 16  X22_16 - 5 X24_7 - 6 X24_8 - 7 X24_9 - 8 X24_10 - 9 X24_11 - 10 X24_12 - 11 X24_13 - 12
X24_14 - 13 X24_15 - 14 X24_16 - 15 X24_17 - 16 X24_18 <= 0
2:16) 7  X23_7 + 8  X23_8 + 9  X23_9 + 10  X23_10 + 11  X23_11 + 12  X23_12 + 13  X23_13 + 14  X23_14 + 15  X23_15 + 16  X23_16 + 17  X23_17 + 18  X23_18 - 7 X25_8 - 8 X25_9 - 9 X25_10 - 10 X25_11 - 11 X25_12 - 12 X25_13 - 13
X25_14 - 14 X25_15 - 15 X25_16 - 16 X25_17 - 17 X25_18 - 18 X25_19 <= 0
2:17) 7  X24_7 + 8  X24_8 + 9  X24_9 + 10  X24_10 + 11  X24_11 + 12  X24_12 + 13  X24_13 + 14  X24_14 + 15  X24_15 + 16  X24_16 + 17  X24_17 + 18  X24_18 - 7 X25_8 - 8 X25_9 - 9 X25_10 - 10 X25_11 - 11 X25_12 - 12 X25_13 - 13
X25_14 - 14 X25_15 - 15 X25_16 - 16 X25_17 - 17 X25_18 - 18 X25_19 <= 0
R0:t0)  X10_0 <= 1
R0:t1)  X01_3 +  X10_1 <= 1
R0:t2)  X01_3 +  X01_4 +  X10_2 <= 1
R0:t3)  X01_3 +  X01_4 +  X01_5 +  X03_5 +  X10_3 <= 1
R0:t4)  X01_4 +  X01_5 +  X01_6 +  X03_5 +  X03_6 +  X10_4 <= 1
R0:t5)  X01_5 +  X01_6 +  X01_7 +  X03_5 +  X03_6 +  X03_7 +  X10_5 <= 1
R0:t6)  X01_6 +  X01_7 +  X01_8 +  X03_6 +  X03_7 +  X03_8 +  X04_8 +  X10_6 <= 1
R0:t7)  X01_7 +  X01_8 +  X01_9 +  X03_7 +  X03_8 +  X03_9 +  X04_8 +  X04_9 +  X10_7 <= 1
R0:t8)  X01_8 +  X01_9 +  X01_10 +  X03_8 +  X03_9 +  X03_10 +  X04_8 +  X04_9 +  X04_10 +  X14_9 <= 1
R0:t9)  X01_9 +  X01_10 +  X01_11 +  X03_9 +  X03_10 +  X03_11 +  X04_9 +  X04_10 +  X04_11 +  X14_9 +  X14_10 <= 1
R0:t10)  X01_10 +  X01_11 +  X01_12 +  X03_10 +  X03_11 +  X03_12 +  X04_10 +  X04_11 +  X04_12 +  X14_10 +  X14_11 <= 1
R0:t11)  X01_11 +  X01_12 +  X01_13 +  X03_11 +  X03_12 +  X03_13 +  X04_11 +  X04_12 +  X04_13 +  X14_11 +  X14_12 <= 1
R0:t12)  X01_12 +  X01_13 +  X01_14 +  X03_12 +  X03_13 +  X03_14 +  X04_12 +  X04_13 +  X04_14 +  X14_12 +  X14_13 <= 1
R0:t13)  X01_13 +  X01_14 +  X01_15 +  X03_13 +  X03_14 +  X04_13 +  X04_14 +  X04_15 +  X14_13 +  X14_14 <= 1
R0:t14)  X01_14 +  X01_15 +  X01_16 +  X03_14 +  X04_14 +  X04_15 +  X04_16 +  X14_14 +  X14_15 <= 1
R0:t15)  X01_15 +  X01_16 +  X01_17 +  X04_15 +  X04_16 +  X04_17 +  X14_15 +  X14_16 <= 1
R0:t16)  X01_16 +  X01_17 +  X04_16 +  X04_17 +  X14_16 <= 1
R0:t17)  X01_17 +  X04_17 <= 1
R1:t0)  X20_3 <= 1
R1:t1)  X20_3 +  X20_4 <= 1
R1:t2)  X20_3 +  X20_4 +  X20_5 <= 1
R1:t3)  X20_3 +  X20_4 +  X20_5 +  X20_6 <= 1
R1:t4)  X20_4 +  X20_5 +  X20_6 +  X20_7 <= 1
R1:t5)  X20_5 +  X20_6 +  X20_7 +  X20_8 +  X22_5 <= 1
R1:t6)  X20_6 +  X20_7 +  X20_8 +  X20_9 +  X22_6 <= 1
R1:t7)  X20_7 +  X20_8 +  X20_9 +  X20_10 +  X22_7 <= 1
R1:t8)  X20_8 +  X20_9 +  X20_10 +  X20_11 +  X22_8 <= 1
R1:t9)  X20_9 +  X20_10 +  X20_11 +  X20_12 +  X22_9 <= 1
R1:t10)  X20_10 +  X20_11 +  X20_12 +  X20_13 +  X22_10 <= 1
R1:t11)  X20_11 +  X20_12 +  X20_13 +  X20_14 +  X22_11 <= 1
R1:t12)  X20_12 +  X20_13 +  X20_14 +  X22_12 <= 1
R1:t13)  X20_13 +  X20_14 +  X22_13 <= 1
R1:t14)  X20_14 +  X22_14 <= 1
R1:t15)  X22_15 <= 1
R1:t16)  X22_16 <= 1
R2:t0)  X00_0 <= 1
R2:t1)  X00_1 +  X02_2 +  X11_4 <= 1
R2:t2)  X00_2 +  X02_2 +  X02_3 +  X11_4 +  X11_5 <= 1
R2:t3)  X00_3 +  X02_3 +  X02_4 +  X11_4 +  X11_5 +  X11_6 <= 1
R2:t4)  X00_4 +  X02_4 +  X02_5 +  X11_4 +  X11_5 +  X11_6 +  X11_7 +  X21_4 <= 1
R2:t5)  X00_5 +  X02_5 +  X02_6 +  X11_5 +  X11_6 +  X11_7 +  X11_8 +  X12_7 +  X13_6 +  X21_5 <= 1
R2:t6)  X00_6 +  X02_6 +  X02_7 +  X11_6 +  X11_7 +  X11_8 +  X11_9 +  X12_7 +  X12_8 +  X13_6 +  X13_7 +  X21_6 +  X24_7 <= 1
R2:t7)  X00_7 +  X02_7 +  X02_8 +  X11_7 +  X11_8 +  X11_9 +  X11_10 +  X12_7 +  X12_8 +  X12_9 +  X13_7 +  X13_8 +  X21_7 +  X24_7 +  X24_8 <= 1
R2:t8)  X00_8 +  X02_8 +  X02_9 +  X11_8 +  X11_9 +  X11_10 +  X11_11 +  X12_8 +  X12_9 +  X12_10 +  X13_8 +  X13_9 +  X21_8 +  X24_8 +  X24_9 +  X25_8 <= 1
R2:t9)  X00_9 +  X02_9 +  X02_10 +  X11_9 +  X11_10 +  X11_11 +  X12_9 +  X12_10 +  X12_11 +  X13_9 +  X13_10 +  X21_9 +  X24_9 +  X24_10 +  X25_9 <= 1
R2:t10)  X02_10 +  X02_11 +  X11_10 +  X11_11 +  X12_10 +  X12_11 +  X12_12 +  X13_10 +  X13_11 +  X21_10 +  X24_10 +  X24_11 +  X25_10 <= 1
R2:t11)  X02_11 +  X11_11 +  X12_11 +  X12_12 +  X12_13 +  X13_11 +  X13_12 +  X21_11 +  X24_11 +  X24_12 +  X25_11 <= 1
R2:t12)  X12_12 +  X12_13 +  X12_14 +  X13_12 +  X13_13 +  X21_12 +  X24_12 +  X24_13 +  X25_12 <= 1
R2:t13)  X12_13 +  X12_14 +  X13_13 +  X13_14 +  X21_13 +  X24_13 +  X24_14 +  X25_13 <= 1
R2:t14)  X12_14 +  X13_14 +  X21_14 +  X24_14 +  X24_15 +  X25_14 <= 1
R2:t15)  X21_15 +  X24_15 +  X24_16 +  X25_15 <= 1
R2:t16)  X24_16 +  X24_17 +  X25_16 <= 1
R2:t17)  X24_17 +  X24_18 +  X25_17 <= 1
R2:t18)  X24_18 +  X25_18 <= 1
R2:t19)  X25_19 <= 1
R3:t4)  X23_7 <= 1
R3:t5)  X23_7 +  X23_8 <= 1
R3:t6)  X23_7 +  X23_8 +  X23_9 <= 1
R3:t7)  X23_7 +  X23_8 +  X23_9 +  X23_10 <= 1
R3:t8)  X23_8 +  X23_9 +  X23_10 +  X23_11 <= 1
R3:t9)  X05_10 +  X23_9 +  X23_10 +  X23_11 +  X23_12 <= 1
R3:t10)  X05_10 +  X05_11 +  X15_12 +  X23_10 +  X23_11 +  X23_12 +  X23_13 <= 1
R3:t11)  X05_11 +  X05_12 +  X15_12 +  X15_13 +  X23_11 +  X23_12 +  X23_13 +  X23_14 <= 1
R3:t12)  X05_12 +  X05_13 +  X15_12 +  X15_13 +  X15_14 +  X23_12 +  X23_13 +  X23_14 +  X23_15 <= 1
R3:t13)  X05_13 +  X05_14 +  X15_13 +  X15_14 +  X15_15 +  X23_13 +  X23_14 +  X23_15 +  X23_16 <= 1
R3:t14)  X05_14 +  X05_15 +  X15_14 +  X15_15 +  X15_16 +  X23_14 +  X23_15 +  X23_16 +  X23_17 <= 1
R3:t15)  X05_15 +  X05_16 +  X15_15 +  X15_16 +  X15_17 +  X23_15 +  X23_16 +  X23_17 +  X23_18 <= 1
R3:t16)  X05_16 +  X05_17 +  X15_16 +  X15_17 +  X15_18 +  X23_16 +  X23_17 +  X23_18 <= 1
R3:t17)  X05_17 +  X05_18 +  X15_17 +  X15_18 +  X15_19 +  X23_17 +  X23_18 <= 1
R3:t18)  X05_18 +  X05_19 +  X15_18 +  X15_19 +  X23_18 <= 1
R3:t19)  X05_19 +  X15_19 <= 1
END
INTE 186
```

Fig. 6.2. An example of LINDO input file, which is automatically generated by PtatoSim (DMP instance: OTEST3-1.DAT, which has 3 projects, 4 resources and 6 jobs per project with time horizon = 25).

6.1.2 Simulation Data Sets

Some of the most frequently referred project scheduling problem sets in literature are Patterson's data (Patterson, 1984) and PSPLIB (Kolisch and Sprecher, 1996). Both of them are basically dedicated to single project scheduling problems, while our DMP problem domain is a distributed multiple projects environment. PSPLIB, which is more frequently referred to recently, was generated by a general purpose project scheduling problem-instance generator called *ProGen* (Kolisch et al., 1992), which can generate multi-project instances. Therefore, we use ProGen to generate a variety of different problem instances for simulating and testing many different aspects of P-TÂTO mechanism. Figure 6.3 shows snapshots of the ProGen input file, execution, and output file.

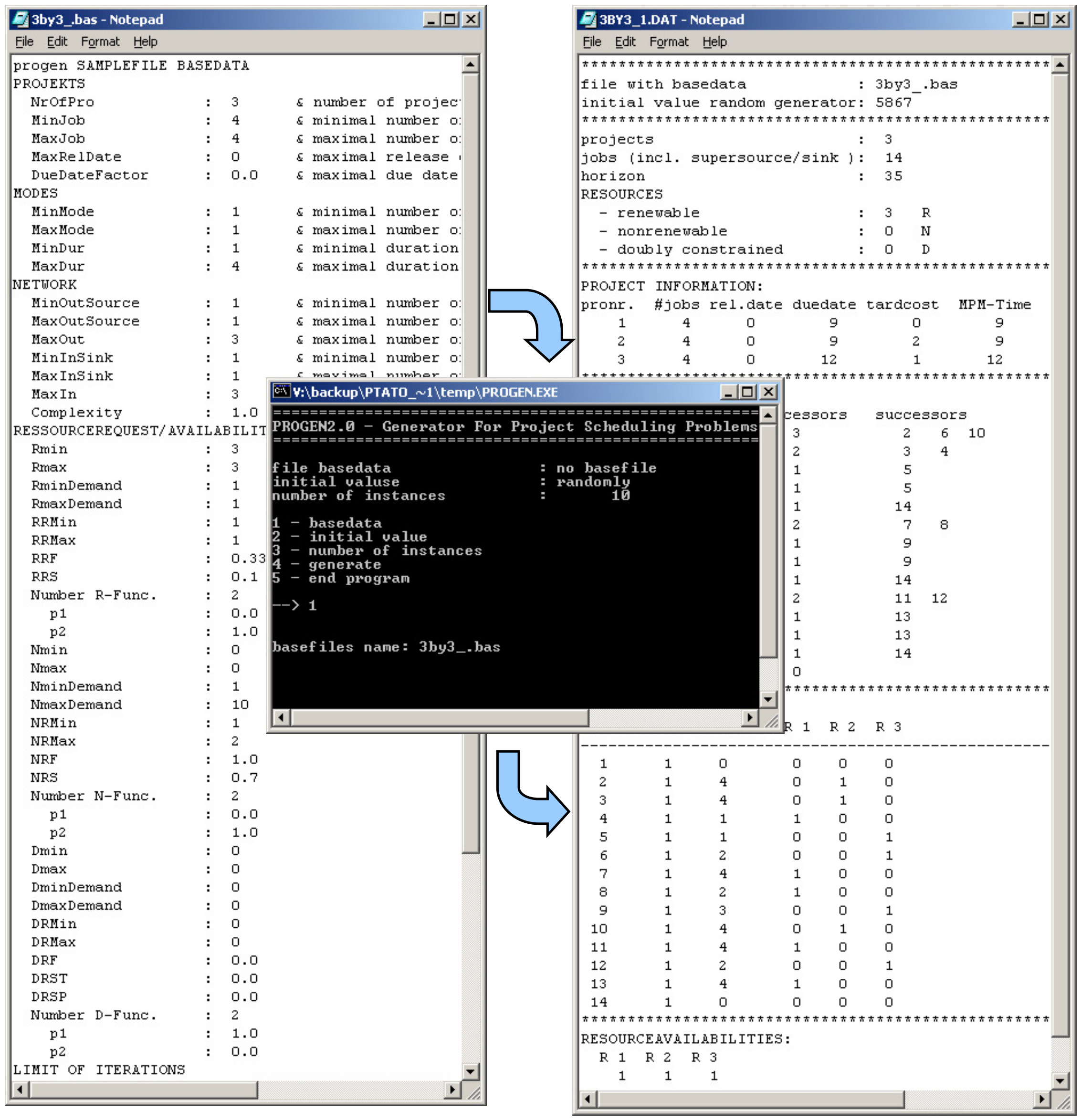
6.1.3 ProGen Parameters

About fifty parameters can be defined in the input file by users to generate project scheduling-problem instances (which have different configurations) using ProGen. In this case, which is single mode, renewable resource only, and multiple project scheduling, only part of the parameters can be altered. Table 6.1 summaries the important parameters to be controlled for generating DMP problem instances. The detailed and comprehensive description on the parameters can be found in (Kolisch et al., 1992) and (Kolisch and Sprecher, 1996).

Among the parameters list in Table 6.1, the following three parameters need to be explained in a little bit more:

- **DueDateFactor**, denoted by δ_{fac}, is used for generating the target dates (G_{i1}) of each project i. $G_{i1} = \text{trunc}(EF_i + \delta_{fac}(\bar{T} - EF_i))$, where EF_i is the earliest finish time[1] of project P_i and $\bar{T}$ is the time horizon, which is the sum of duration times for all jobs through out the projects.

[1] This is automatically calculated without any consideration of resource constraints based on the critical path analysis.



```
progen SAMPLEFILE BASEDATA
PROJEKTS
  NrOfPro             :  3      & number of projec
  MinJob              :  4      & minimal number o
  MaxJob              :  4      & maximal number o
  MaxRelDate          :  0      & maximal release
  DueDateFactor       :  0.0    & maximal due date
MODES
  MinMode             :  1      & minimal number o
  MaxMode             :  1      & maximal number o
  MinDur              :  1      & minimal duration
  MaxDur              :  4      & maximal duration
NETWORK
  MinOutSource        :  1      & minimal number o
  MaxOutSource        :  1      & maximal number o
  MaxOut              :  3      & maximal number o
  MinInSink           :  1      & minimal number o
  MaxInSink           :  1      & maximal number o
  MaxIn               :  3
  Complexity          :  1.0
RESSOURCEREQUEST/AVAILABILIT
  Rmin                :  3
  Rmax                :  3
  RminDemand          :  1
  RmaxDemand          :  1
  RRMin               :  1
  RRMax               :  1
  RRF                 :  0.33
  RRS                 :  0.1
  Number R-Func.      :  2
    p1                :  0.0
    p2                :  1.0
  Nmin                :  0
  Nmax                :  0
  NminDemand          :  1
  NmaxDemand          :  10
  NRMin               :  1
  NRMax               :  2
  NRF                 :  1.0
  NRS                 :  0.7
  Number N-Func.      :  2
    p1                :  0.0
    p2                :  1.0
  Dmin                :  0
  Dmax                :  0
  DminDemand          :  0
  DmaxDemand          :  0
  DRMin               :  0
  DRMax               :  0
  DRF                 :  0.0
  DRST                :  0.0
  DRSP                :  0.0
  Number D-Func.      :  2
    p1                :  1.0
    p2                :  0.0
LIMIT OF ITERATIONS
```

```
===================================================
PROGEN2.0 - Generator For Project Scheduling Problems
===================================================

file basedata                   : no basefile
initial valuse                  : randomly
number of instances             :        10

1 - basedata
2 - initial value
3 - number of instances
4 - generate
5 - end program

--> 1


basefiles name: 3by3_.bas
```

```
****************************************************
file with basedata            : 3by3_.bas
initial value random generator: 5867
****************************************************
projects                      :  3
jobs (incl. supersource/sink ):  14
horizon                       :  35
RESOURCES
  - renewable                 :  3   R
  - nonrenewable              :  0   N
  - doubly constrained        :  0   D
****************************************************
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  MPM-Time
    1      4      0        9        0        9
    2      4      0        9        2        9
    3      4      0       12        1       12
****************************************************

                      cessors   successors
                      3            2   6   10
                      2            3   4
                      1            5
                      1            5
                      1           14
                      2            7   8
                      1            9
                      1            9
                      1           14
                      2           11  12
                      1           13
                      1           13
                      1           14
                      0
                      *****************************

                      R 1  R 2  R 3
----------------------------------------------------
  1      1      0       0    0    0
  2      1      4       0    1    0
  3      1      4       0    1    0
  4      1      1       1    0    0
  5      1      1       0    0    1
  6      1      2       0    0    1
  7      1      4       1    0    0
  8      1      2       1    0    0
  9      1      3       0    0    1
 10      1      4       0    1    0
 11      1      4       1    0    0
 12      1      2       0    0    1
 13      1      4       1    0    0
 14      1      0       0    0    0
****************************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  R 3
    1    1    1
```

Fig. 6.3. Snapshots of ProGen input file, execution window, and output file (The sample project has 3 projects, 3 resources and 4 jobs per projects).

Table 6.1. Controllable ProGen Parameters for DMP problems with example values.

Categories	Parameters	Description	Example
Projects	NrOfPro	number of projects	3
	MinJob	minimal number of jobs per projects	10
	MaxJob	maximal number of jobs per projects	10
	DueDateFactor	maximal due date	0.0
Modes	MinDur	minimal duration of tasks	1
	MaxDur	maximal duration of tasks	10
Network	MinOutSource	minimal number of start activities per project	1
	MaxOutSource	maximal number of start activities per project	1
	MaxOut	maximal number of successor per activity	3
	MinInSink	minimal number of end activities per project	1
	MaxInSink	maximal number of end activities per project	1
	MaxIn	maximal number of predecessors per activity	3
	Complexity	complexity of network	1.2
Resources	Rmin	minimal number of renewable resources	4
	Rmax	maximal number of renewable resources	4
	RF	resource factor	0.25

- **Complexity**, denoted by C, is the average number of arcs per node[2] (in activity-in-node representation). So, this complexity measure can be understood in the way that for a fixed number of jobs a higher complexity results in a greater interconnectedness of the network.

- **Resource Factor**, denoted by RF, is the average portion of resources requested per job. If $RF = 1$, then each job requests all resources. $RF = 0$ indicates that no job requests any resources, thus we obtain the unconstrained case. In our case, we fix $Rmin = Rmix$ and $RF = \frac{1}{Rmin}$ because a job occupies a single resource type (i.e., single mode and single resource unit).

Figure 6.4 shows a project scheduling instance, which was generated by ProGen using the input parameters listed in Table 6.1. An actual ProGen-generated project instance file (`*.dat`) includes two additional (dummy) nodes, called *super-source* and *super-sink*, constituting a single big project network. These super-nodes are eliminated to get separate multiple project networks by PtatoSim internally.

6.2 Performance of Local Market Evaluation Mechanism

The optimality and efficiency of the local market evaluation mechanism are critical factors for those of the temporary economy clearing mechanism (namely, whole P-TÂTO mechanism). Especially the optimality and efficiency of the winning bid determination step (explained in Section 4.5) is the dominant factors for those of the local market evaluation mechanism. In order to evaluate the optimality and efficiency of the local market evaluation heuristic algorithm, we investigate 10 randomly generated DMP instances (data set 35J81 $\backsim$ 35J810), each of which has 3 projects, 5 resources and 8 tasks per project. P-TÂTO converged in 11, 3, 10, 3, 7, 20, 12, 8, 4 and 4 iterations for these DMP instances respectively. Because every resource agent solves its own local market combinatorial auction in every iteration, we can gather a total of 410 ($= 82 \times 5$) local

[2]Here, the super-node and super-sink are also taken into account.

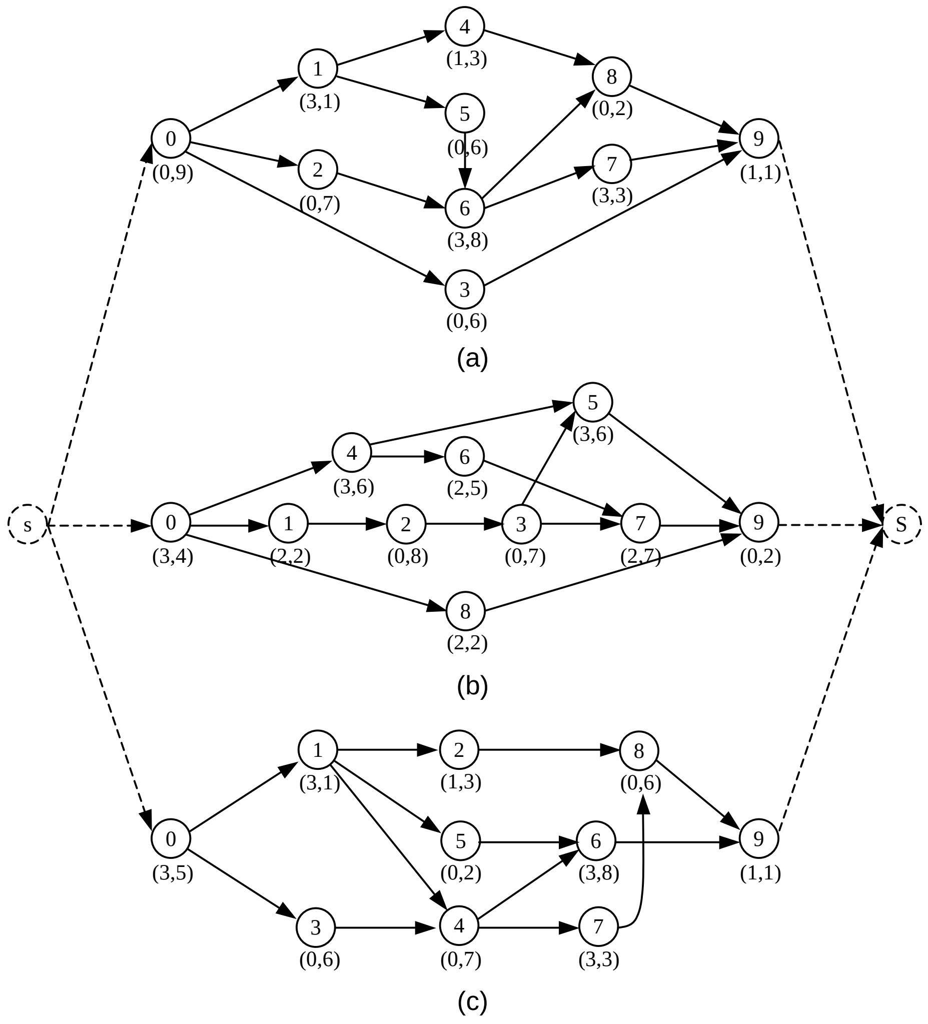


Fig. 6.4. Sample DMP network generated by ProGen (See previous table for the parameters) (a) project-1 (due: 28, weight: 2); (b) project-2 (due: 29, weight: 5); (c) project-3 (due: 30, weight: 7). The two numbers below each node denote resource identifiers and duration times, respectively. The nodes S and s are super-sink and super-source respectively.

market evaluation instances, whose numbers of tasks ranges from 2 to 8[3]. They were compared to the optimal local schedules, which were found by total enumeration[4]. The results are summarized in Table 6.2 and Figure 6.5. As shown in Figure 6.5(a), the search space (the number of examined sequences) increases almost linearly as the number of tasks in a resource increases, while the total search space (which is a set of all possible task sequences) increase in a factorial way. As a result, *search efficiency* - the number of examined sequences divided by the number of all possible sequences - converges to 0 quickly as the number of tasks increases (as shown in Figure 6.5(c)). However, the optimality of the solution is still very high, ranging from 99.8% to 100% of the optimal solution, which generated by total enumeration (as shown in Figure 6.5(c)).

Table 6.2. Experimental results of local market allocation algorithm.

No. of tasks per resource (N)	2	3	4	5	6	7	8
Total number of possible sequences ($N!$)	2	6	24	120	720	5040	40320
Average number of examined sequences (n)	2	3.3	4.2	5.8	9.3	8.4	10.3
Average search efficiency ($n/N!$) (full search = 1.0)	1.0	0.56	0.175	0.046	0.013	0.002	3E-4
Average optimality (optimal = 1.0)	1.0	1.0	1.0	1.0	0.999	0.998	0.999

6.3 Performance of Temporary Economy Mechanism

In order to test the solution optimality of temporary economy, we tested 10 randomly generated small-size DMP instances. Each of these DMP instances has 3 projects, 4 resources and 6 jobs per project[5]. The target dates are set to the earliest completion

[3]Because ProGen generates project instances in a random way, the numbers of tasks assigned per resource are not always same for each resource.

[4]In order to study the optimality of the local market evaluation algorithm using PtatoSim, the user can choose if either the heuristic algorithm or the total enumeration is used for the local market evaluation.

[5]For bigger problems, for example more than 6 jobs per project, the number of variables in the IP formulation easily exceeds 800, which is the maximum number of integer variables of the standard LINDO solver.

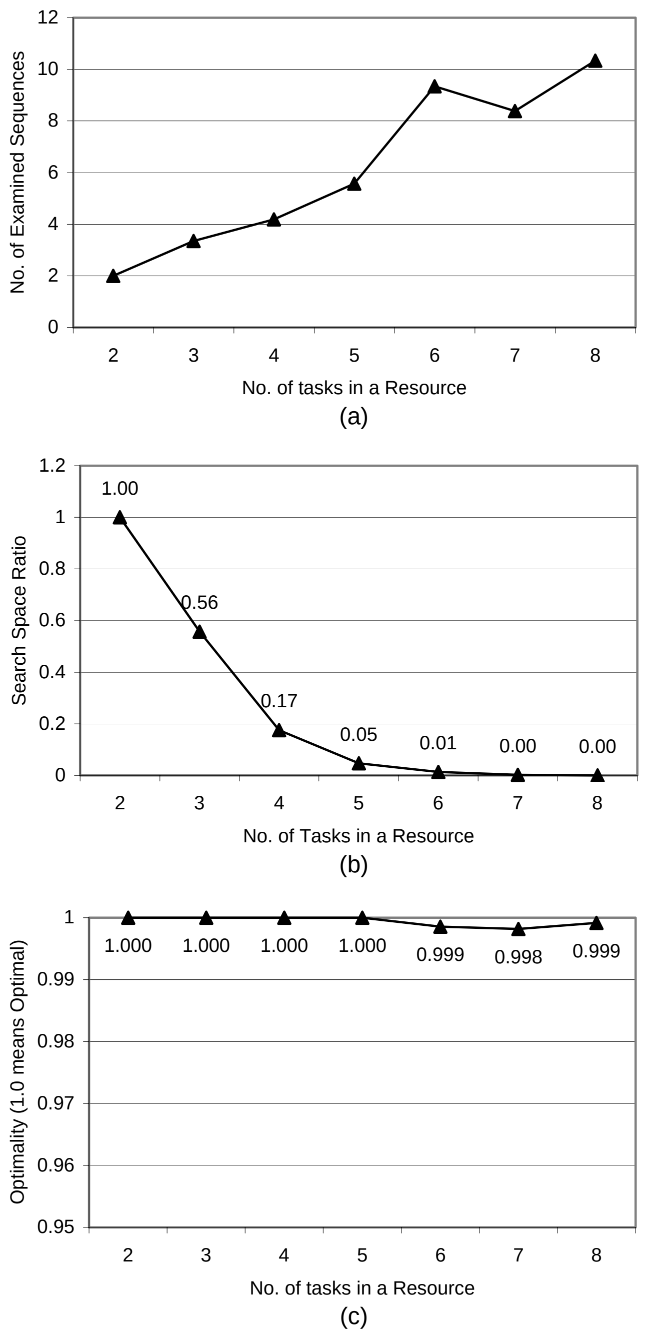


Fig. 6.5. Experimental performance of local market optimal allocation algorithm (Data set: 35J81 ∽ 35J810, total 410 instances used): (a) average numbers of examined sequences; (b) average search efficiency; (c) optimality.

date based on the project-wise critical path analysis ($\delta_{fac} = 0$), meaning that all DMP instances are supposed to generate positive weighted deviation cost values due to the resource constraints. This setting is too tight of a condition for real practice, because they cannot finish the projects earlier than the given target dates by any means. However, in the experimental situation, we can see the solution quality of different solution approaches clearly based on this setting. The DMP instances were also solved by LINDO, based on the IP formulation explained in Chapter 2. The LINDO input models are automatically generated by *PtatoSim* using *Export to LINDO* menu. Figure 6.6 shows the weighted deviation costs of the sample data sets. The weights (or unit tardiness costs) are assigned to each project, ranging from 1.0 to 5.0, which means the the results (of weighted sum of deviation costs) are 1 to 5 times bigger than the real delay (in dates or hours, for example). As shown in the figure, 6 out of 10 instances generate optimal solution with the average excessive weighted deviation cost = 11%.

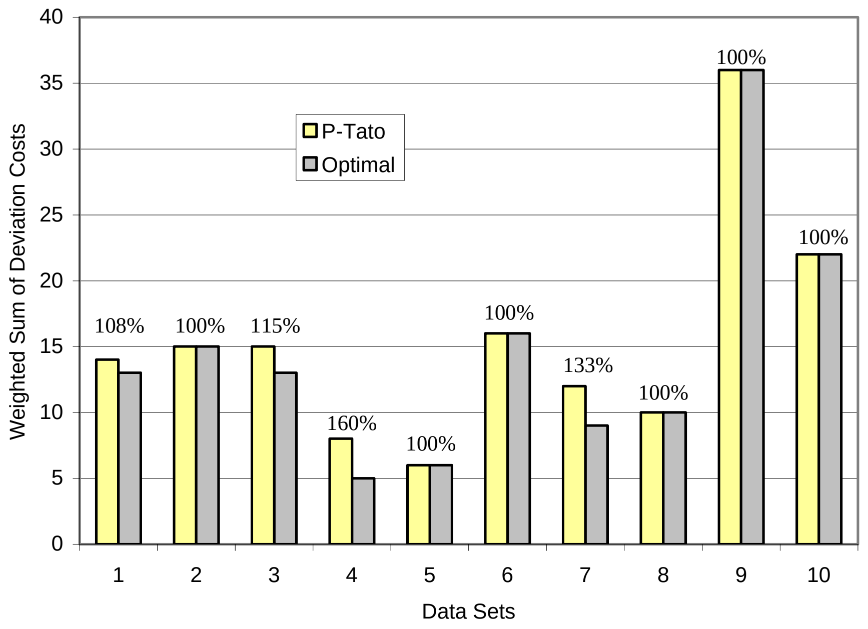


Fig. 6.6. Optimality of P-TÂTO for small examples (Data set: OTEST3). Each of ten DMP instances has 3 projects, 4 resources and 6 activities per project.

Figure 6.7 illustrates the computational time comparison between PtatoSim and LINDO for the same DMP instances of Figure 6.6. The average computation time is about 12% of the LINDO solution. The computation time for both approaches are seriously dependent on the length of the time horizon (see Section 6.4.1). Especially for the IP formulation, the number of integer variables increases rapidly with the time horizon. We run the PtatoSim first using enough time horizon for the algorithm, and based on the schedule generated by the PtatoSim, we set the minimum number of time slots in the horizon for the IP formulation. Hence, the computational time of the IP formulation in the experiments might be longer than the results if we compare them in a same condition. In addition, the gap between computational efficiencies of P-TÂTO and the IP formulation will get bigger when the P-TÂTO mechanism is implemented in a multiagent system, where the most-time-consuming local market evaluation is to be carried out by each resource agent simultaneously in distributed local computers.

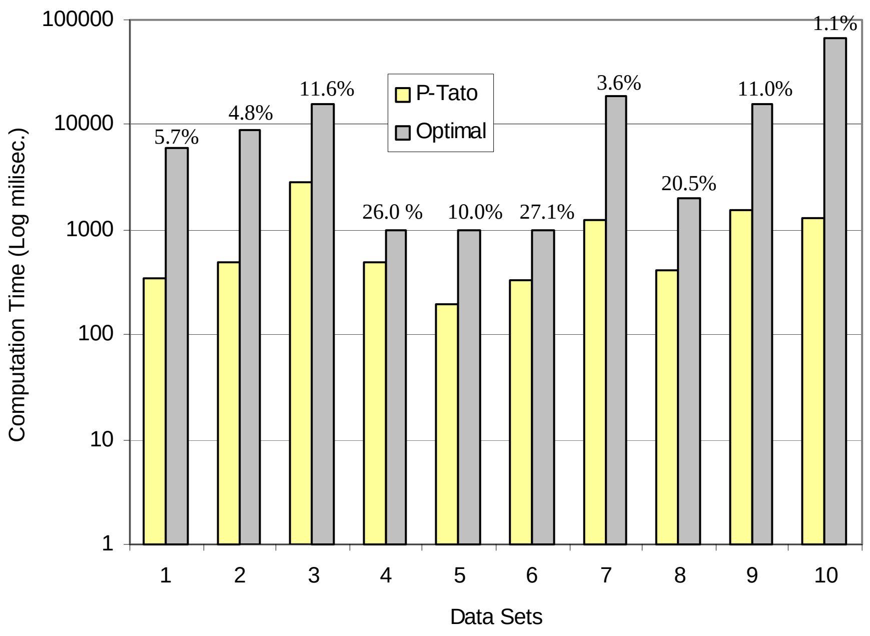


Fig. 6.7. Computation time comparison P-TÂTO vs. IP formulation (using PtatoSim and LINDO respectively).

6.4 Effects of DMP Configuration: Scalability

Scalability is defined as the ability of a system or algorithm to continue to function *well* as either the system or the assigned task is changed in size or volume. Hence, the measure of scalability must be closely related to the concept of efficiency, which is the ability to do a task within certain constraints. In order to study the scalability of the P-TÂTO mechanism, it must be examined for changes in computational performance as some of problem parameters - including number of projects, number of resources, number of tasks, and scheduling horizon - increases. We examine the overall performance of the P-TÂTO as the DMP configuration varies accordingly. We investigate many DMP instances of difference configuration using the following metrics:

- **Convergence** : The convergence is measured by the number of iterations until the P-TÂTO mechanism is completed or stops. The condition for stopping is controlled in PtatoSim by two user-defined parameters: (1) *maximum number of iteration* (I_{max}), (2) *precedence cost change*[6] *tolerance* (ρ). We set the maximum number of iteration 400, and set $\rho = 0$ in most cases.[7]

- **Computational efficiency** : We measure the computation time in PtatoSim. We are not only interested in the total time until the P-TÂTO mechanism converges but also the local computation time for each project and resource, because the whole P-TÂTO mechanism will be carried out in a distributed computing environment in practice.

- **Dynamic behavior** : As a complimentary information for convergence of the P-TÂTO mechanism, we also check changes of intermediate output values including deviation costs of the projects in each iterations. This additional information is useful to understand the speed of convergence.

[6] *Precedence cost change* is defined as the sum of absolute values of *precedence cost adjustment* (Δc_{kl}, where k is a task index and l is a slot index.) in a given iteration.

[7] If deviation cost change at the I_{max}-th iteration is still bigger than ρ, PtatoSim returns the best feasible solution during the iterations. If there is no feasible solution until then, the solution at the last iteration is to be returned although it is not a feasible solution.

In the remainder of this section, we investigate the effects of some characteristic factors of the DMP configuration on the above performance measures of P-TÂTO. Investigated characteristic factors of a DMP configuration include:

- **Scheduling horizon**: The scheduling horizon defines the number of time slots, which are the commodities to be bought and sold. So, this parameter affects the performance of the local market evaluation mechanism, and ultimately the overall P-TÂTO mechanism.
- **Complexity of project networks**: The more precedence relations between tasks in a project means more tasks are coupled in generating feasible schedules. We use the average number of arcs from a node as the measure of the complexity of project networks.
- **Number of Projects, Resources and Tasks**: These three factors define the overall size and the resource availability of a DMP configuration. We examine the effects of the number of projects, resource availability, and the number of tasks per project.

6.4.1 Effect of Scheduling Horizon

The time horizon of a project scheduling problem significantly affects the performance of the IP solution procedure. How about P-TÂTO mechanism? In order to evaluate the effect of the scheduling horizon on the performance of P-TÂTO mechanism, we investigate 3 randomly generated sets of DMP instances thoroughly. Each DMP instances in these 3 sets has 3 projects, 4 resources and 10 tasks per each project. Each set has 5 DMP instances, and the corresponding three instances across the sets are exactly the same instances except for the scheduling time horizon. The levels of time horizon are 50, 80, and 100.

The experimental results are summarized in Table 6.3. As shown in the table, the overall performance (e.g., convergence, solution quality, etc.) are not different from each other. Even the dynamic behaviors during the P-TÂTO mechanism exactly the same as shown in Figure 6.8. The computational load increases as the time horizon increases (see

Figure 6.9), because the complexity of the local market evaluation algorithm increases according to the length of scheduling horizon. However, the computational complexity seems to increase almost linearly, rather than exponentially, as the horizon increases for the tested horizon range (50 - 100) as shown in the figure.

Table 6.3. Experimental results of testing the effect of varying scheduling horizons (†: infeasible allocation happened in Resource-1).

time horizon	number of iterations		latest completion	aggregate deviation cost	computation time total (sec)	computation time per iteration
50	1	140	36	76	81	0.58
	2	22	38	18	6	0.27
	3†	-	40	-	-	-
	4	16	42	47	4	0.25
	5	9	34	56	3	0.33
80	1	140	36	76	360	2.57
	2	22	38	18	36	1.64
	3	23	40	16	30	1.30
	4	16	42	47	41	2.56
	5	9	34	56	21	2.33
100	1	140	36	76	765	5.46
	2	22	38	18	71	3.23
	3	23	40	16	66	2.87
	4	16	42	47	88	5.50
	5	9	34	56	40	4.44

6.4.2 Effect of Project Network Complexity

P-TÂTO is the first convergent market-based mechanism applied to (multiple) "project" resource scheduling environment. That means the complexity of the project network can be a source of significant computational burden, which has never been reported in the literature. In order to analyze the effect of project network complexity on the computational efficiency, we need project networks having different levels of network complexity with other characteristics remaining the same. We define the network

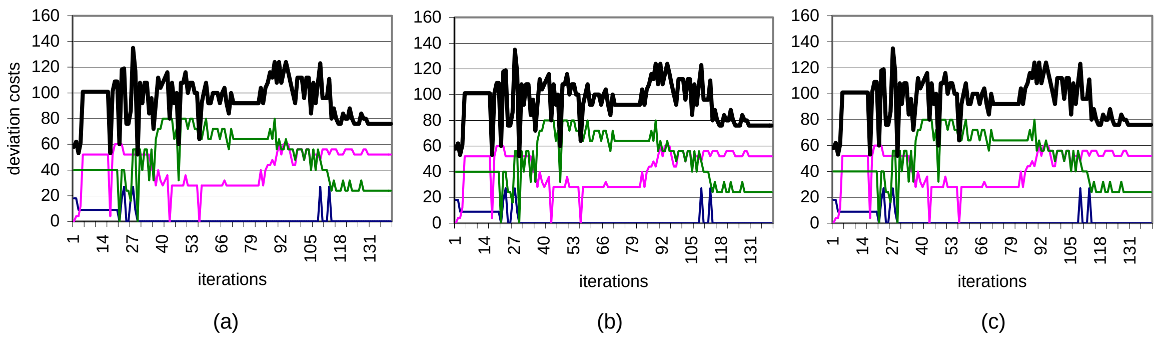


Fig. 6.8. Deviation cost plot of a same DMP instances varying time horizon (The 1st instances of the data set: C12H50, C12H80, C12H100; the thick lines are aggregate deviation costs, while the other lines are deviation costs of individual projects). (a) horizon = 50, (b) horizon = 80, (c) horizon = 100.

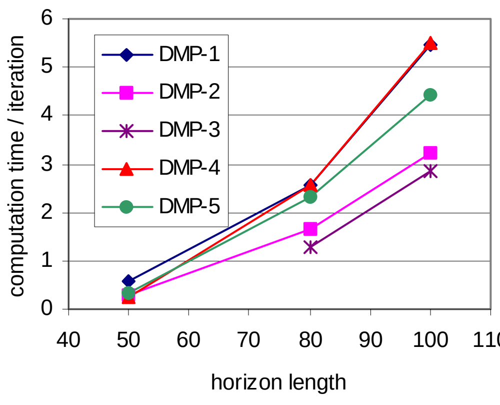


Fig. 6.9. Effect of the scheduling horizon on computation time (second).

complexity, which is a measure of coupling among tasks within a project based on the precedence relationship, as follows.

Definition 6.1. (Project Network Complexity) The *project network complexity* of a project is the average number of outgoing arcs of each node excluding the first and the last nodes (i.e., super-source and super-sink nodes as explained in Section 6.1.3), where the project network uses *activity-in-node* representation scheme. □

We generate DMP instances with 3 projects, 3 resources and 15 jobs per projects using ProGen. The initial project networks are shown in Figure 6.10 and consist of 28 arcs per project excluding the arcs from the starting nodes. Hence, the project network complexity is 2.15. We decrease the complexity of the project networks in multiple steps by removing arcs randomly as shown in Figure 6.11. The complexity levels of the project networks are 1.85, 1.54, 1.23 and 1.08. For exact comparison, we use the same length of scheduling horizon for all instances. We investigate the dynamic behavior of each project's deviation costs over iterations and computation complexity (using convergence and computation time).

- Dynamic behavior of each DMP instance varies according to the difference levels of complexity as shown in Figure 6.12. We can hardly find any trend on dynamic behavior for the different levels of network complexity.
- The computation time and the number of iterations do not increase according to the increase of complexity as shown in Figure 6.13, and in some regions ($1.08 \leq$ project network complexity ≤ 1.54), the measures even decrease as the network complexity increases, meaning that the computational complexity of P-TÂTO mechanism is also indifferent to (or positively affected by) the complexity of project networks.

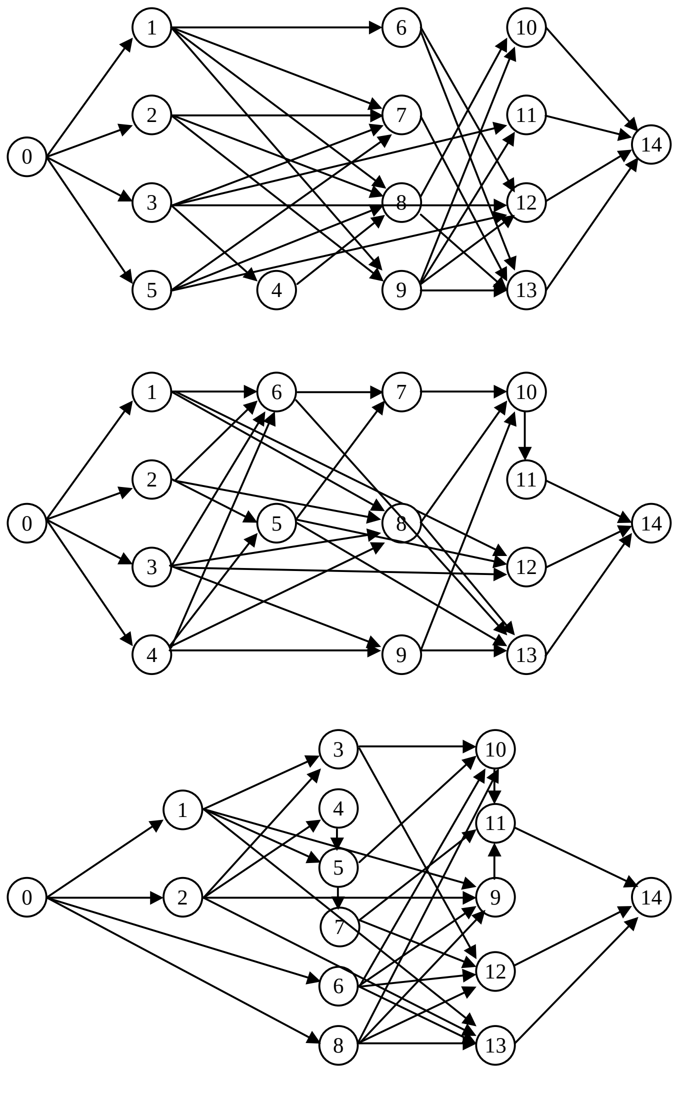


Fig. 6.10. The initial project networks with complexity = 2.15 (Data Set: P3R3J10-28)

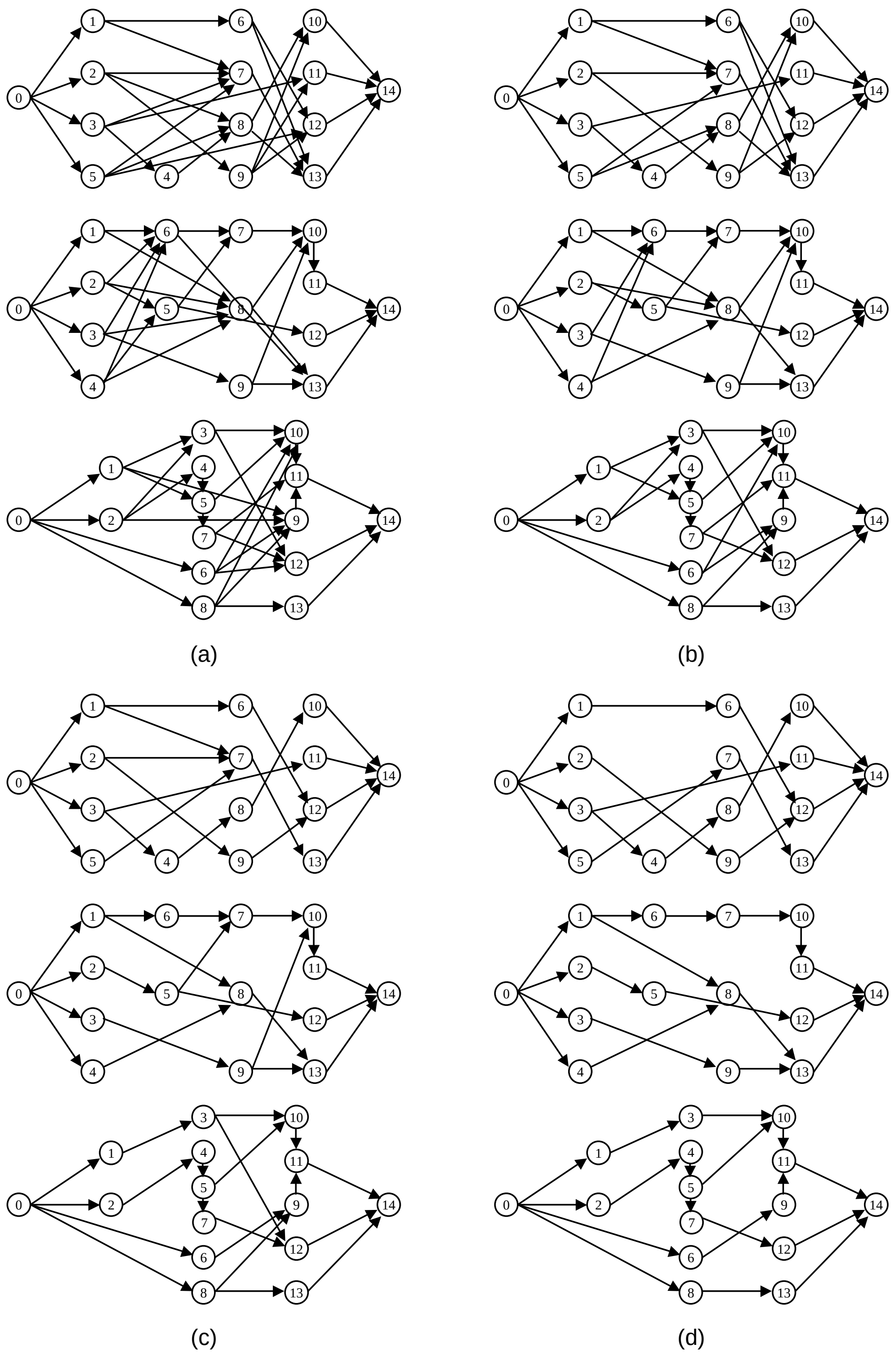
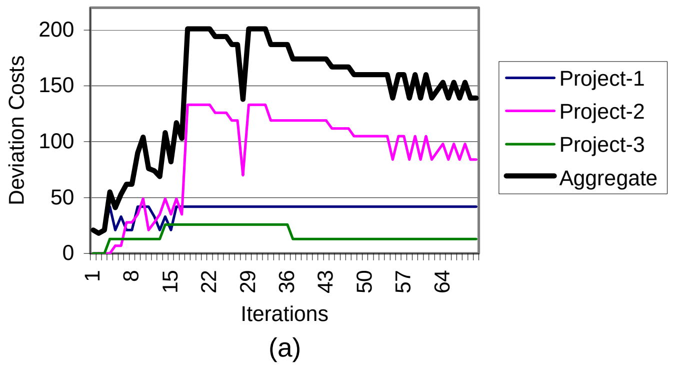
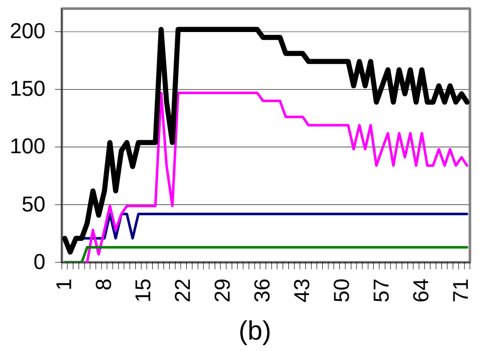
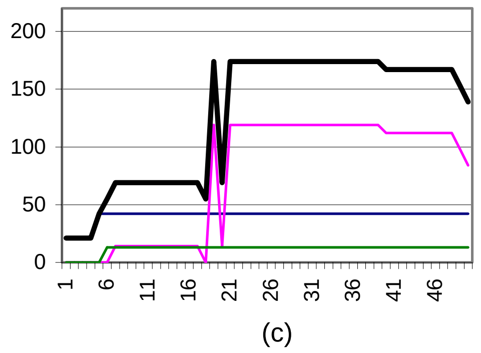
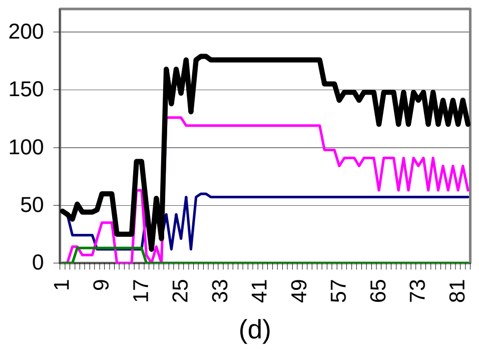


Fig. 6.11. DMP instances with different levels of project network complexity: (a) 1.85, (b) 1.54, (c) 1.23, (d) 1.08.









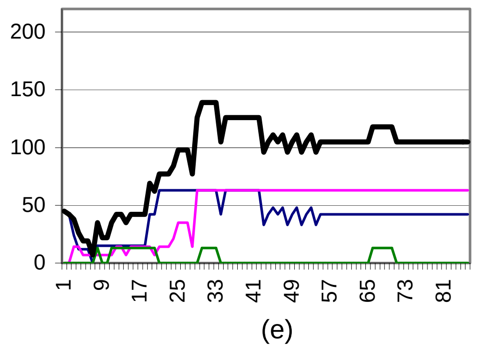


Fig. 6.12. Dynamic behaviors of the DMP instance according to the different levels of network complexity (complexity levels: (a) 2.15 (b) 1.85, (c) 1.54, (d) 1.23, (e) 1.08).

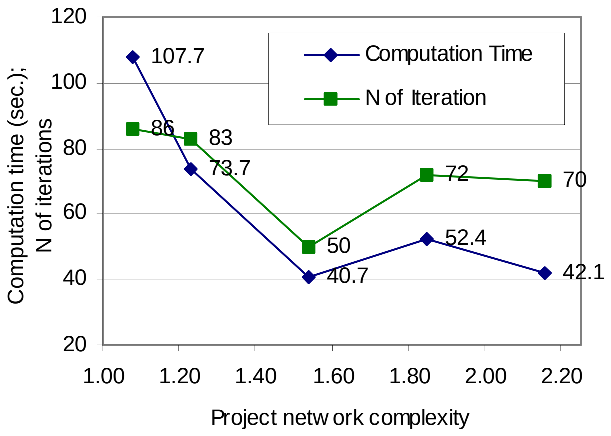


Fig. 6.13. Computational measures (computation time and number of iterations) of the DMP instances with different levels of project network complexity.

6.4.3 Effect of Number of Projects, Resources, and Tasks

The number of projects, resources and tasks determines the size of the DMP configuration. In the P-TÂTO mechanism, two main conflict resolution interactions take place (a) between task agents and resource agents in each local market and (b) among task agents within a project. Hence, the relative availability of resources with respect to the number of tasks is more influential than the number of resources itself. In the same sense, the number of tasks per project is a more direct factor to computational efficiency compare to the total number of tasks. We define the *resource availability* as follows:

Definition 6.2. (Resource Availability) The *resource availability* of a DMP instance is the ratio of the number of available resources over the total number of tasks which require the resources. □

We first examine the effect of the number of projects by fixing the resource availability and the number of tasks per project, then investigate the effect of resource availability and the number of tasks per project.

6.4.3.1 Effect of Number of Projects

In order to examine the effect of the number of projects on the computational efficiency of P-TÂTO mechanism, we randomly generate 160 DMP instances, each with 2 to 5 projects as shown in Table 6.4. The 16 data sets are grouped according to 4 different levels of resource availability and the number of tasks per project. Figure 6.14 shows the simulation results. As shown in the figure, the effect of the number of projects is not significant as far as resource availability, and the number of tasks per project remains the same.

Table 6.4. Data sets and simulation results on the effect of number of projects (scheduling horizon = 80, network complexity = 1.2).

No. of projects	tasks / project	No. of resources	resource availability	No. of DMP instances	data set name	average iterations
2	6	2	6	10	P2J6R2	10.3
	8	2	8	10	P2J8R2	13.4
	10	2	10	10	P2J10R2	23.6
	12	2	12	10	P2J12R2	39.4
3	6	3	6	10	P3J6R3	9.9
	8	3	8	10	P3J8R3	16.8
	10	3	10	10	P3J10R3	27.2
	12	3	12	10	P3J12R3	30.9
4	6	4	6	10	P4J6R4	11
	8	4	8	10	P4J8R4	24.2
	10	4	10	10	P4J10R4	38.2
	12	4	12	10	P4J12R4	49
5	6	5	6	10	P5J6R5	12
	8	5	8	10	P5J8R5	17.6
	10	5	10	10	P5J10R5	25.2
	12	5	12	10	P5J12R5	48.5
			total instances	160		

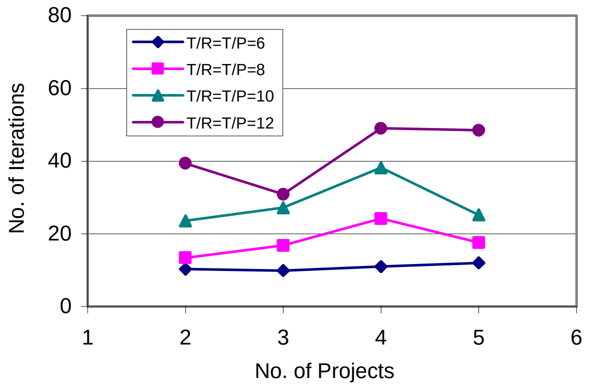


Fig. 6.14. Effect of the number of projects on computational load.

6.4.3.2 Effect of Resource Availability and the Number of Tasks Per Project

In order to examine both resource availability and the number of tasks per project, we simulate 350 DMP instances as summarized in Table 6.5. As shown in the table, the data sets have different levels of resource availability (levels are 2, 3, 4, 6, 8, 9, 12, and 18) and number of tasks per project (levels are 4, 6, 8, 9, 12, and 18). The simulation results show an overall trend of increasing number of iterations according to both of the *inverse resource availability* and the number of tasks per project as shown in Figure 6.15. Here the inverse resource availability is the average number of tasks to be assigned per resource. Note that the scales of the domain axes (both of resource availability and the number of tasks per project) are not linear.

Figure 6.16 shows the effect of the resource availability and the number of tasks per project separately. As shown in the figure, according to the increase in the number of tasks per project, the convergence speed increases almost linearly (see Figure 6.16(a)). A similar result is shown for the effect of resource availability (Figure 6.16(b)). That means P-TÂTO mechanism is scalable or applicable to a practical size of the problems, because the computational time per each iteration varies from a few seconds to 1 minute in extreme cases. Also this time must be divided by the number of local markets (i.e., number of resources), which can be evaluated simultaneously on separate computers.

Table 6.5. Data sets and simulation results on the effect of resources availability and the number of tasks per project (scheduling horizon = 80, network complexity = 1.2).

No. of projects	tasks / project	No. of resources	resource availability	No. of DMP instances	data set name	average iterations
2	8	4	4	10	P2J8R4	4.2
	9	6	3	10	P2J9R6	3.8
	12	6	4	10	P2J12R6	5.2
	12	8	3	10	P2J12R8	4.7
	18	6	6	10	P2J18R6	21.0
	18	9	4	10	P2J18R9	7.8
	18	12	3	10	P2J18R12	8.9
4	4	2	8	10	P4J4R2	6.2
	6	4	6	10	P4J6R4	9.9
	6	6	4	10	P4J6R6	6.1
	6	8	3	10	P4J6R8	4.0
	12	4	12	10	P4J12R4	66.6
	12	6	8	10	P4J12R6	30.2
	12	8	6	10	P4J12R8	18.9
	18	4	18	10	P4J18R4	123.7
	18	6	12	10	P4J18R6	78.6
	18	8	9	10	P6J18R8	39.2
6	4	2	12	10	P6J4R2	14.0
	4	4	6	10	P6J4R4	6.3
	4	6	4	10	P6J4R6	3.9
	4	8	3	10	P6J4R8	3.9
	6	2	18	10	P6J6R2	33.0
	8	4	12	10	P6J8R4	30.4
	8	6	8	10	P6J8R6	21.9
	8	8	6	10	P6J8R8	13.3
	12	4	18	10	P6J12R4	70.8
	12	6	12	10	P6J12R6	46.6
	12	8	9	10	P6J12R8	43.2
8	6	4	12	10	P8J6R4	22.3
	6	6	8	10	P8J6R6	18.3
	6	8	6	10	P8J6R8	12.3
	9	4	18	10	P8J9R4	49.1
	9	6	12	10	P8J9R6	36.7
	9	8	9	10	P8J9R8	22.9
9	4	2	18	10	P9J4R2	21.0
			total instances	350		

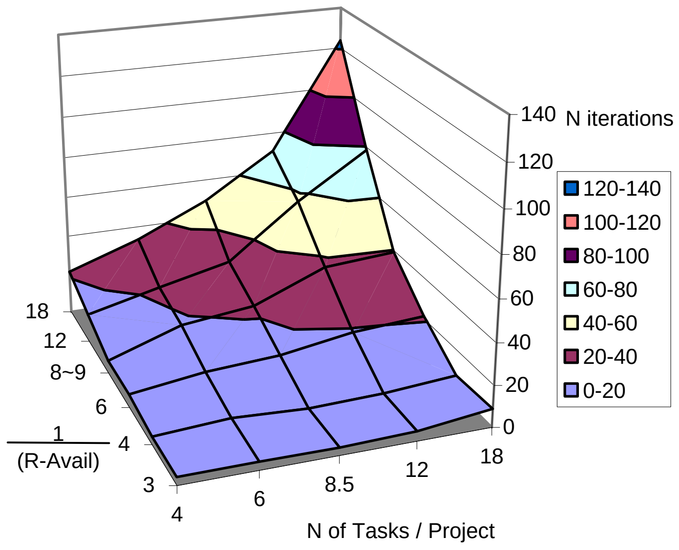


Fig. 6.15. Effect of resource availability and number of tasks per project on computational load. (Note: The scales of the domain axes are not linear. Hence the steepness in the range of 12 to 18 is inflated.)

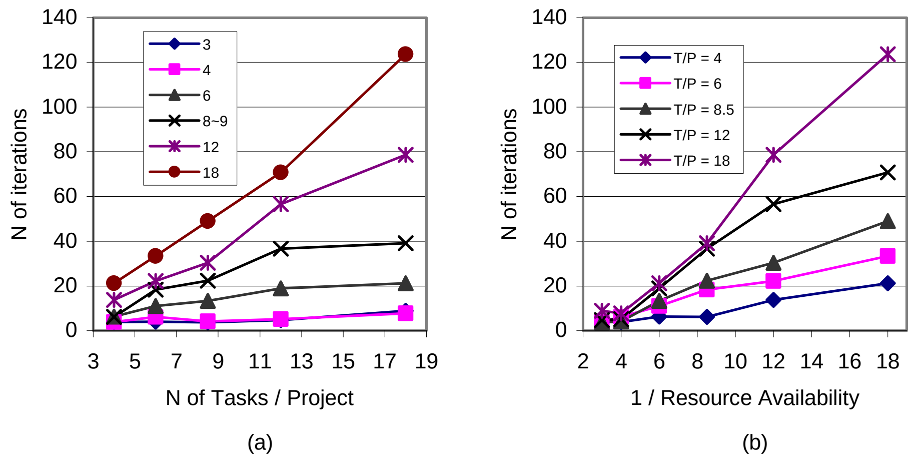


Fig. 6.16. Effect of resource availability and number of tasks per project on computational load: (a) effect of the number of tasks per projects, (b) effect of resource availability.

6.5 Controllability

Each project can have a unique deviation cost function in the proposed problem solving model. Consider a linear deviation cost function, which has a unit tardiness cost as the slops of the function. This unit tardiness cost can apparently be interpreted as the level of urgency or importance (simply a *weight*) of the project. We can give different weights to different projects, and the weights can be changed over time depending on the DMP situation. Whatever the cases are, different levels of the weight must affect the resultant deviation costs of the projects after the temporary economy clearance using the P-TÂTO mechanism. In order to examine the correlations clearly, we transform the weight (w_i's) and resultant deviation cost (d_i's) of each project to generate a *normalize weight* ($\tilde{w_i}$'s) and a *normalized deviation cost* ($\tilde{d_i}$'s) where i denotes the project index as follows:

$$\tilde{w_i} = (w_i - min_i(w_i))/(max_i(w_i) - min_i(w_i)) \tag{6.1}$$

$$\tilde{d_i} = 1 - (d_i - min_i(d_i))/(max_i(d_i) - min_i(d_i)) \tag{6.2}$$

The controllability measure can be defined using these values as follows:

Definition 6.3. (Controllability) The *controllability* is the correlation coefficient between normalized weight distribution and the resultant deviation cost distribution (generated by P-TÂTO mechanism). □

In order to verify the *controllability* of P-TÂTO mechanism, we conducted experimental analysis using 20 randomly generated DMP instances, each of which has 4 projects, 4 resources, 10 jobs per project, network complexity = 1.5, and due date factor = 0.0. For three sample DMP instances from the data set, we plot the deviation costs of individual projects over iterations. As shown in Figure 6.17, in the early stage there is no clear differentiation in resultant deviation costs according to the given weights distribution, but soon the deviation costs are ordered clearly according to the given weight distribution.

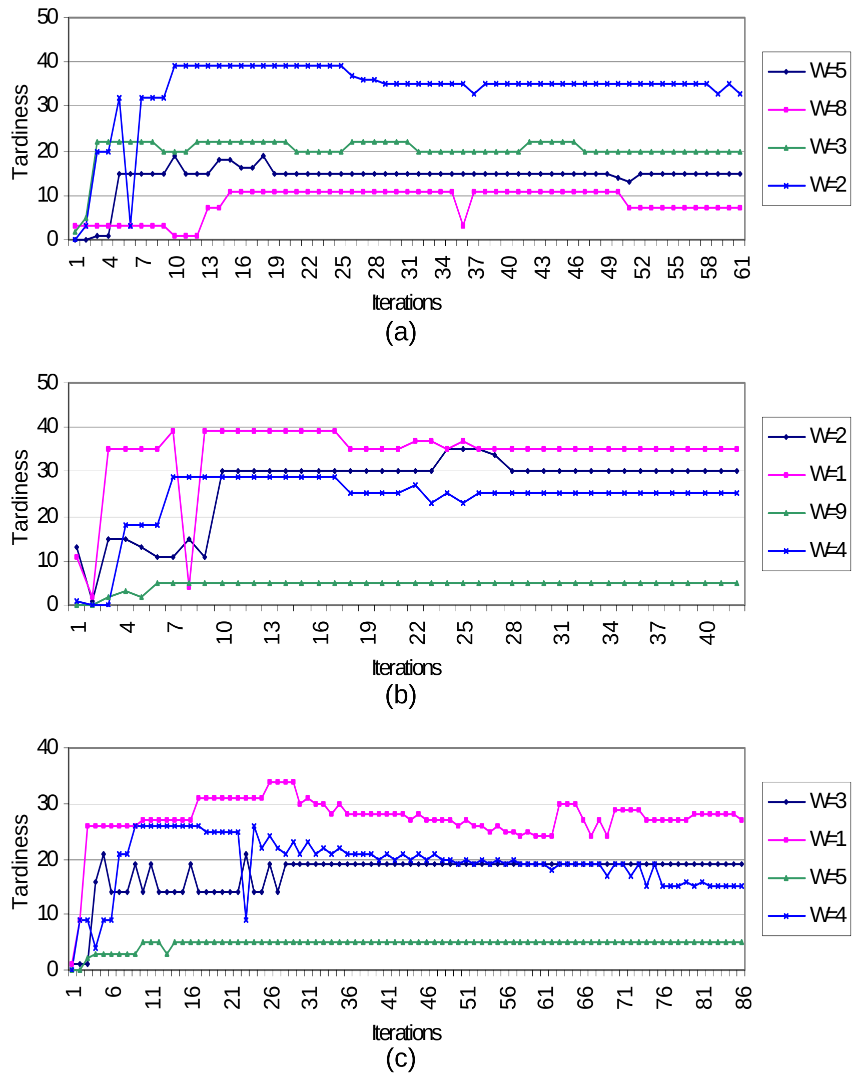


Fig. 6.17. Time series of tardiness with different tardiness costs (W): (a) CONT-02, (b) CONT-04, (c) CONT-05, each of which has 4 projects, 4 resources, 10 jobs per project, network complexity 1.5, due date factor 0.0.

Figure 6.18 shows the normalized correlation between the normalized project weight ($\tilde{w}_i$'s) distribution and resultant deviation cost ($\tilde{d}_i$'s) distribution. As shown in figure, most of the instances show the strong correlation between the tardiness cost weights and resultant tardiness values, and the calculated controllability is 0.89, saying that there is strong correlation between the given weights and the resultant weights. The dynamic behavior and the correlation coefficient show that we can relate the importance of the project and the resultant cost deviation.

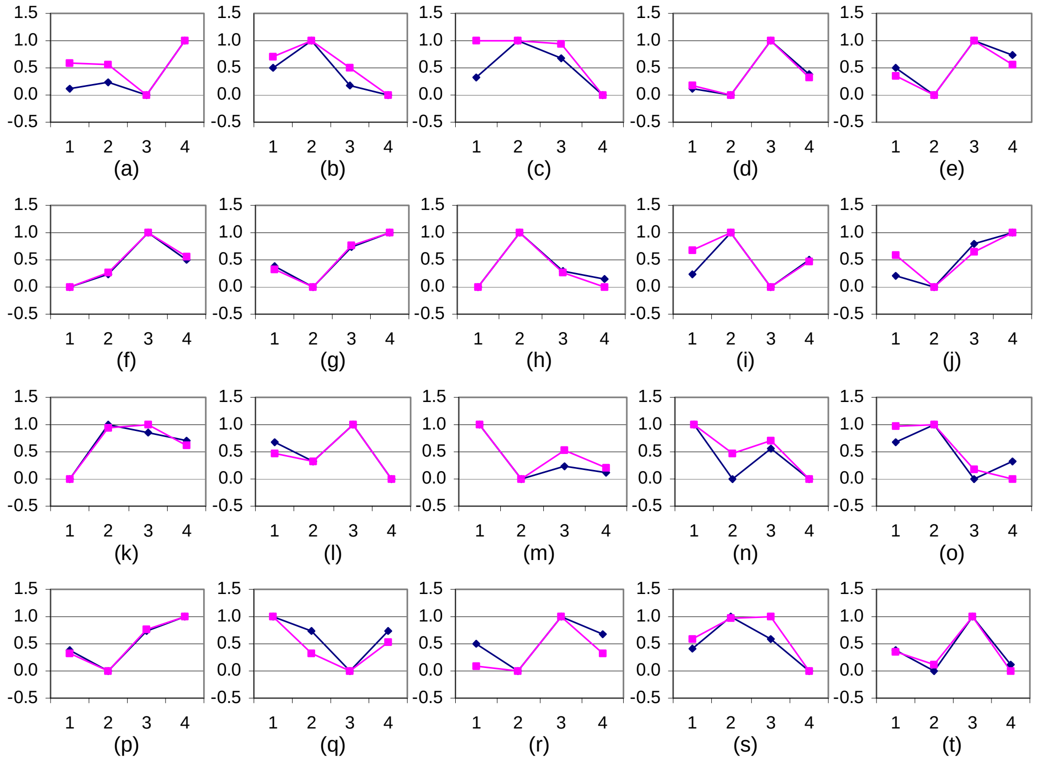


Fig. 6.18. Correlation between tardiness cost weight and resultant tardiness of projects (♦: normalized tardiness cost weights; ■: normalized resultant tardiness values of projects).

6.6 Convergence

In the precedence cost adjustment rule (see Section 4.4.5), $\bar{c}_I$ can be interpreted as a strength of the force to enable precedence-feasible schedules, while $\bar{c}_D$ as a strength of the force to compress the schedule. Hence the ratio α between these two parameters ($= \bar{c}_I/\bar{c}_D$) can affect the convergence performance of P-TÂTO significantly. We examined 60 DMP instances in three different size classes (as summarized in Table 6.6), by applying different α levels and measuring the number of iterations until the mechanism stops (as a measure of convergence speed).

As expected, the number of iterations until convergence varies greatly depending on the different levels of α, while we could not find any significant difference in solution quality for different levels of α. As shown in Figure 6.19 (a), $\alpha = 0.4$ to 0.8 reveals good convergence speeds. Obviously, for $\alpha \geq 1.0$ we found ill-convergence for all three classes because the precedence costs will keep increasing over iterations. For $\alpha = 1.0$, 1.2 and 1.4, 18%, 77% and 82% of the DMP instances failed to converge without any feasible solutions generated in 500 iterations as shown in Figure 6.20. The average number of iterations for $\alpha = 0.2$ to 0.8 are 12.6, 71.5, and 119.0 for the three classes, respectively.

Table 6.6. Experimental design (number of DMP instances) for testing the effect of α values on the convergence of P-TÂTO mechanism (CONV1: $4\times4\times6$, CONV2: $4\times4\times12$, CONV3: $4 \times 4 \times 18$, where (N of projects)$\times$(N of resources)$\times$(N of jobs per project); $\bar{c}_I = 5$).

α levels	data set-1 (CLASS-1)	data set-2 (CLASS-2)	data set-3 (CLASS-3)
0.2	20	20	20
0.4	20	20	20
0.6	20	20	20
0.8	20	20	20
1.0	20	20	20
1.2	20	20	20
1.4	20	20	20

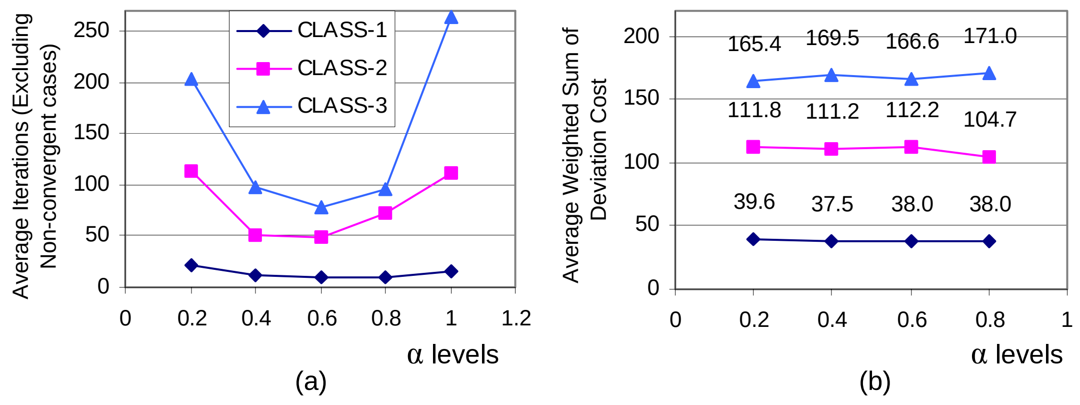


Fig. 6.19. Effect of α values: (a) on the convergence performance, (b) on the solution quality.

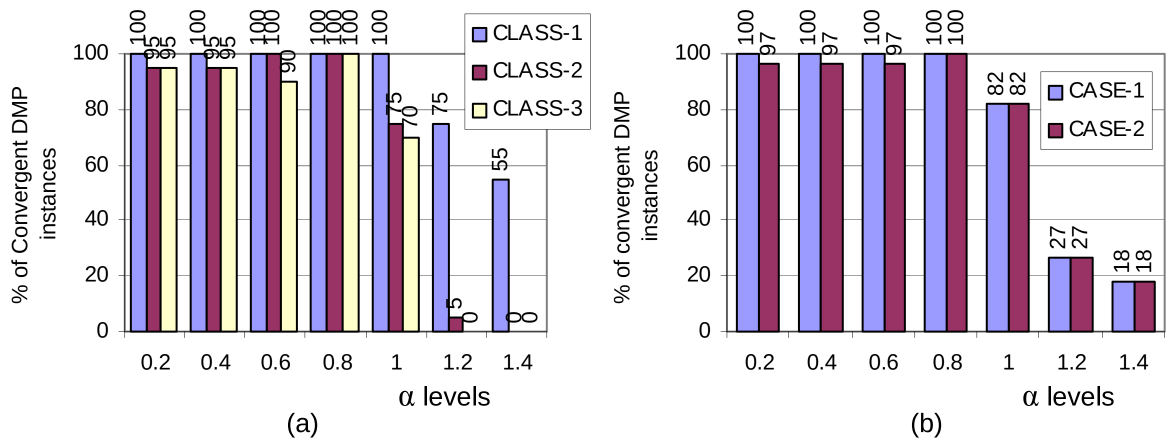


Fig. 6.20. Effect of α values on convergence (in 500 iterations) : (a) percentages of convergent DMP instances, (b) average percentages of convergent DMP instances (CASE-1: including feasible/non-convergent cases; CASE-2: excluding feasible/non-convergent cases).

Table 6.7. Simulation result (the number of iterations): effect of α on convergence ($*$: not convergent in 500 iterations, but obtained the best feasible solution at this iteration; †: neither converged nor obtained a feasible solution in 500 iterations).

	$\alpha = 0.2$			$\alpha = 0.4$			$\alpha = 0.6$			$\alpha = 0.8$			$\alpha = 1.0$			$\alpha = 1.2$			$\alpha = 1.4$		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
1	33	36	438	15	19	56	11	16	106	9	25	95	8	38	226	9	†	†	†	†	†
2	5	208	89	3	86	78	4	145	55	4	129	82	5	†	†	5	†	†	6	†	†
3	25	87	155	17	52	67	9	70	69	9	167	96	28	†	387	†	†	†	†	†	†
4	19	102	133	13	33	72	15	31	52	14	51	75	25	195	†	27	†	†	20	†	†
5	20	67	231	8	49	48	8	51	30	7	72	59	8	52	190	10	†	†	8	†	†
6	21	335	369	10	78	90	9	55	86	8	19	152	9	72	†	12	†	†	†	†	†
7	30	70	316	23	14	167	22	15	136	19	12	160	36	25	†	†	†	†	†	†	†
8	2	159*	195	2	45*	75*	2	43	116	2	36	80	4	95	476	4	†	†	4	†	†
9	39	190	165	21	164	88	14	124	69	10	230	186	13	†	143	28	†	†	†	†	†
10	4	67	121	4	29	81	6	37	104	6	19	132	7	51	†	7	126	†	7	†	†
11	6	112	216	6	49	93	6	23	27	6	26	44	7	62	166	6	†	†	8	†	†
12	46	128	162	21	37	136	14	53	175*	19	113	73	40	60	†	†	†	†	†	†	†
13	17	82	105	10	39	33	8	35	126	7	30	58	7	60	448	†	†	†	†	†	†
14	32	190	138	14	88	64	7	65	46	7	157	95	5	269	287	10	†	†	6	†	†
15	7	94	295*	9	29	79	11	26	98	16	41	133	63	103	145	†	†	†	†	†	†
16	4	107	276	3	59	146	3	39	70*	3	65	62	3	235	0	7	†	†	7	†	†
17	40	43	114	15	21	214	10	39	64	9	74	117	13	331	250	26	†	†	†	†	†
18	38	89	215	18	62	101	13	53	58	15	56	52	17	†	148	21	†	†	23	†	†
19	7	83	242	6	32	84	5	53	36	5	95	99	9	†	112	13	†	†	29	†	†
20	19	68	184	10	28	146	8	17	125	9	20	74	5	37	463	5	†	†	5	†	†

Table 6.8. Simulation result (weighted deviation cost): effect of α on convergence ($-$: neither converged nor obtained a feasible solution in 500 iterations).

	$\alpha = 0.2$			$\alpha = 0.4$			$\alpha = 0.6$			$\alpha = 0.8$			$\alpha = 1.0$			$\alpha = 1.2$			$\alpha = 1.4$		
	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
1	21	48	186	21	48	190	21	48	165	21	48	187	21	38	181	21	-	-	-	-	-
2	17	181	139	8	195	132	8	163	131	8	170	132	8	-	-	8	-	-	8	-	-
3	30	108	222	30	108	246	30	108	246	30	101	246	30	-	177	-	-	-	-	-	-
4	59	115	166	59	115	166	59	131	166	59	125	167	52	141	-	52	-	-	59	-	-
5	90	136	137	90	136	187	90	136	181	90	136	169	90	140	113	90	-	-	90	-	-
6	71	70	196	38	67	190	49	67	196	49	70	214	38	78	-	38	-	-	-	-	-
7	71	72	201	71	82	173	71	82	173	71	56	191	55	53	-	-	-	-	-	-	-
8	3	109	209	3	109	219	3	92	206	3	110	209	3	71	209	3	-	-	3	-	-
9	50	151	184	50	168	115	50	165	115	50	172	222	50	-	143	50	-	-	-	-	-
10	23	47	210	23	42	204	23	103	263	23	37	227	23	56	-	23	56	-	23	-	-
11	45	116	88	45	118	102	45	122	116	45	116	121	45	115	99	45	-	-	45	-	-
12	33	151	179	33	165	108	33	185	158	33	151	125	43	39	-	-	-	-	-	-	-
13	41	115	152	41	115	155	41	94	110	41	50	133	41	77	100	-	-	-	-	-	-
14	28	148	163	28	148	170	28	148	181	28	131	187	28	107	122	28	-	-	28	-	-
15	63	89	176	63	94	231	63	89	228	63	98	180	63	93	157	-	-	-	-	-	-
16	6	155	126	6	128	150	6	136	164	6	136	105	6	128		6	-	-	6	-	-
17	57	164	167	57	124	165	57	112	159	57	124	155	70	110	77	70	-	-	-	-	-
18	52	83	144	52	83	162	52	76	133	52	76	176	52	-	126	52	-	-	52	-	-
19	22	138	109	22	138	126	22	138	112	22	138	126	31	-	95	31	-	-	31	-	-
20	9	40	153	9	41	198	9	49	129	9	49	147	9	28	111	9	-	-	9	-	-

6.7 Summary

In this chapter, the results of the experimental analysis are presented. In order to simulate the P-Tâto mechanism, we developed and utilized a P-Tâto simulator program - PtatoSim. The sample DMP instances for the simulation study were generated by a general purpose project-instance generator - ProGen. Using the ProGen we generated a variety of project instances for verifying specific aspects of P-Tâto mechanism. The simulation results support a high level of solution quality and computational efficiency of the mechanism. The analysis results are summarized as follows:

- The local market evaluation mechanism shows a high level of computational efficiency and near-optimal solution quality. As the number of tasks increases in a local market, the computational load in the local market evaluation mechanism increases almost linearly, while the optimality of the solution remains at a very high level (see Section 6.2).

- At the temporary economy level, the average computation time of P-ÂTO is less than 12% of the result of LINDO, while the average deviation costs are closed to the optimal values (deviating less than 11% for a extreme case). More importantly, LINDO reaches the limit of solvable problem size quickly according to the increase of the scheduling horizon, but not in P-TÂTO (see Section 6.3).

- The overall performance (e.g., convergence, solution quality, etc.) are not much different for different levels of scheduling time horizons in P-TÂTO. The dynamic behavior of the intermediate deviation costs are exactly the same, even though the computational load seems to increase according to the increase in scheduling horizon, but almost linearly (see Section 6.4.1).

- The effect of a DMP size (the number of projects, resource availability, and the number of tasks per project) is not a significant factor in computational efficiency of the P-TÂTO mechanism. Although the number of iterations (i.e., convergence speed) increases with the *resource availability*$^{-1}$ and the *number of tasks per project*, the increasing rate is almost linear (see Section 6.4.3).

- The dynamic behavior (e.g., converging pattern), the computation time and the number of iterations are quite indifferent to the complexity of project networks. Even more precedence constraints seem helpful for better convergence speed. This property is very desirable, because the proposed mechanism deals with project environments, which is differentiated from other types of scheduling problems by the project activity network structure (see Section 6.4.2).

- Regardless of the initial solution, the resultant DMP schedule clearly reflects the order and magnitude of the given project weight distribution. That means we can relate each project's importance/urgency and the resultant deviation costs throughout the P-TÂTO mechanism (see Section 6.5).

- The number of iterations until convergence varies greatly depending on the α level, without any significant variations in the solution quality. $\alpha = 0.4$ to 0.8 shows a good convergence speed for different sizes of DMP instances (see Section 6.6).

Chapter 7

Conclusions and Future Research

In distributed multiple projects (DMP) environments, effective resource control is an important issue because operational management effort for everyday face-to-face resource rescheduling (resource control) is significant. In addition, the results of face-to-face negotiations for resource rescheduling fall far short from the enterprise-level optimality with respect to the relative importance or urgency of each project. That means we need an automated approach to apply the *project weight distribution* at the low-level everyday resource control process in a consistent manner. However, this problem is not easy to solve. The difficulty of the problem arises from the dynamic, distributed, tightly-coupled, and decentralized nature of DMP resource control problem as explained in Chapter 1. We solve this problem through a market-based control approach operating in a multiagent information infrastructure. First, we clearly define the DMP resource control problem in Chapter 2 and propose a virtual economy model and a detailed design of market mechanism called P-TÂTO, along with multiagent information system to embody the mechanism and the virtual economy model in Chapters 4 and 5. The simulation results shown in Chapter 6 demonstrate that the proposed approach works effectively for the DMP resource control problem. In the following sections, we review the major contributions from this research, and suggest future research that is needed.

7.1 Contributions

The results of the proposed economy model and market mechanism were already summarized in Section 6.7. Instead, we discuss the major contributions from this research work as follows:

1. This research is the first to deal with the distributed multiple projects (DMP) resource control problem. This research sets a direction for DMP problem solving. We identified the importance and difficulty of DMP resource control problem, and formally specified the DMP resource control problem in Chapters 1 and 2. Also we suggested and examined the important performance measures of DMP resource control solution approaches. All of them contribute toward the future research on the DMP resource control problem.

2. We successfully combine a market-based approach, operations research methodology, and multiagent information systems for the DMP resource control problem. The market-based control mechanism and the multiagent-based information infrastructure effectively support the distributed and decentralized nature of the DMP environment. In addition, by applying time-slot based scheduling formulation from the operations research literature, we transform the short-term scheduling problem into general resource allocation problem so that the market-based resource allocation approach can be directly applied.

3. The proposed virtual economy model is a novel approach that effectively incorporates the dynamic, coupled and distributed nature of the DMP resource control problem. The three-layered virtual market model (i.e., DMP economy, temporary economy, and local market) is effective to tackle resource constraints and precedence constraints separately. Distributed local markets solve the resource conflicts within each resource schedule, while in the the temporary economy level precedence conflicts among the tasks within each project are resolved using the precedence tâtonnement process.

4. The local market evaluation mechanism, which is a core part of the P-TÂTO mechanism, is formulated as a combinatorial auction. The winning bid determination problem in a combinatorial auction is a $\mathcal{NP}$-complete problem. We developed an effective local market winning bid determination algorithm, based on the proposed problem structure and dynamic programming (DP) based heuristics. Simulation results show the computational demand increases linearly according to the increase in the number of tasks to be allocated to a resource.

5. The individual DMP agent model proposed in this research is an effective combination of previous models. A behavior-based, state-based and message-driven architecture was formalized using an augmented finite automaton, which represents the behavior of a DMP agent. Also it was shown that the P-Tâto can be effectively embodied by several core behaviors of each agent.

6. Using the general purpose project generator, we tested many different aspects of the virtual economy model and control mechanism. Our simulation analysis procedure using the DMP instances can give a good guidelines for solution performance analysis of DMP resource control solving models.

7.2 Future Research

Although this research suggested a clear problem definition for the DMP resource control problem, and a thorough problem solving model including virtual economy model, market mechanism and information infrastructure, some extensions are available for more general problem situations. Followings are three major possible extensions of the proposed solution method.

7.2.1 Multi-Mode Scheduling - Double Auction Mechanism

In this research we restrict the resource scheduling problem to the single-mode case, meaning that each task can be processed by only one resource type. If we allowed multiple modes, the local market evaluation mechanism must be modified to a double auction mechanism, where multiple sellers and multiple buyers bid for the exchange of a designated commodity. Two possible double auction mechanisms can be considered (Wurman et al., 1998): (1) *continuous double auction* (CDA), which matches buyers and sellers immediately on detection of compatible bids; and (2) *call market* (or *clearinghouse*), which is a periodic version of the double auction, which clears the market at the expiration of a specified bidding interval.

7.2.2 Full Self-Interestedness - Incentive Compatible Mechanism Design

If the DMP environment is extended to the situation of totally self-interested decision makers, the local market evaluation mechanism requires modification to satisfy two feasibility conditions of the auction mechanism: *individual rationality* and *incentive compatible.* Individual rationality means that each agent in a market benefits by participating in the allocation mechanism. In other words, every rational agents will attend the bidding if they are rational. The payment rule must be carefully designed for individual rationality requirement. A mechanism is called *incentive compatible* if truth revelation is each bidder's dominant strategy which should be adopted. For example, if the DMP resource control system allows human access to the utility function, the auction mechanism needs to have a device to make the bidders honest. Otherwise, every bidder will bid with the maximum utility value on desirable time-slot bundles in every auction, causing chaos. Hence, incentive compatibility is a crucial property of the mechanisms for coordinating distributed self-interested agents.

Vickrey (Vickrey, 1961) suggested an incentive compatible auction mechanism that asks the participants to bid on a good and awards the good to the highest bidder at the *second* highest price[1]. The original form of the Vickrey auction has been generalized by several researchers including MacKie-Mason and Varian (1994), Krishna and Perry (1998), Cramton and Ausubel (1999); Ausubel (1999). With some exceptions, most of them were generalized in the sense that they allow multiple units for a bidder and complementarity among the goods. GVAs (Generalized Vickrey Auctions) provide a good starting point to design an incentive compatible version of P-TÂTO mechanism, but it is not directly applicable. One simple reason is that the task agents do not

[1]So, it is often called *second-price sealed bid auction.*

actually care about the winning price because the bids are submitted with an abstract utility value rather than real money.

For both of the individual rationality and incentive compatible mechanism designs, one possible approach is to introduce the concept of a *budget*. The basic idea is that every project starts with a specific amount of *virtual* money or budget, which can be used for buying resource time slots. However, this kind of simple *limited budget* model requires complicated decision making procedure for software agents, who are in charge of most decision making in the proposed system. In other words, a project can be bankrupt before finishing the project if the project manager fails to manage the budget effectively.

7.2.3 Loosely-Coupled Organization - Coalition Formation

We assumed that task agents or project agents are assigned to pre-defined resource agents, and these assignments between the resource agents and task agents are fixed. However, if the DMP resource control problem is extended to more loosely-coupled *virtual extended enterprises*, where the organization of the DMP itself can dynamically changed based on changing market needs. In such a situation, the task agents have to search for resource agents, or vice versa, or the temporary economy cannot be easily be broken down into local markets. In this case, finding the appropriate resources for each task could be thought of as coalition formation (Bhargava et al., 2002). This is not considered in this research, but it is a possible area in which future research may be useful because the virtual extended enterprise will be common in the future. In addition, we can find another possible extension to coalition formation within the local market evaluation process. Suppose in the proposed formulation that the task agents are motivated to form a coalition because the combined bid for $B_{i,k}$ has a better chance to get the required time-slot bundles, $B_{i,j}$ and $B_{j+1,k}$. In such a case the DMP resource control problem can be extended to address coalition formation. This offers an important benefit in the DMP situation if an efficient coalition formation search mechanism can be found. The overall convergence of the mechanism and communication burden will be reduced, since during the coalition search process infeasible sets of bids will be ruled out.

Appendix A

An Example Problem of P-TÂTO Mechanism

In order to verify our approach, we apply the P-TÂTO protocol to an example from (Kutanoglu and Wu, 1997) as shown in Table A.1. This example is a simple random job shop scheduling problem. We converted the notations to those of project environment and added the latest project completion date, which define the project failure. This date is used for calculating latest finish time $\overline{w}_{i,j}$ of each task. Some assumptions for this example are:

- Deterministic task duration time as shown in the Table A.1.
- Linear deviation cost function. Project P_i's deviation cost function $\theta_i(t) = max\{(t-\tilde{g}_i), 0\}$, meaning that the cost is just as mush as the project tardiness (see Figure 4.7(a)).
- Fixed change rate of precedence costs. Namely, $\bar{c} = 1$ in the precedence cost adjustment rule in Section 4.4.5.

Table A.1. An example projects (from 3X3 Random job shop example (Kutanoglu and Wu, 1997), converted to a project environment): Each project has three tasks and they are in a row.

Project	Weight	Target Date	Failure Date	Resource (Duration Time)
1	4	10	21	1(3), 2(1), 3(6)
2	6	10	21	3(3), 1(7), 2(1)
3	2	12	21	1(2), 3(4), 2(4)

Using the critical path analysis[1], we get the commitment windows [$\underline{w}_{i,j}$,$\overline{w}_{i,j}$] of each tasks as shown in Table A.2. The commitment window information allows for saving

[1]The commitment windows are calculated using minimal task duration in general. In this example the minimal task duration is the given task duration itself because we assume the deterministic duration time.

communication and computation effort. However, we do not use this information from the second iteration for simplicity.

Table A.2. (Example-1) after critical path analysis.

Project (P_i)	Task ($T_{i,j}$)	Resource (R_k)	Duration ($\delta_{i,j}$)	$\underline{w}_{i,j}$	$\overline{w}_{i,j}$
P_1	$T_{1,1}$	R_1	3	1	13
	$T_{1,2}$	R_2	1	4	14
	$T_{1,3}$	R_3	6	5	20
P_2	$T_{2,1}$	R_3	3	1	12
	$T_{2,2}$	R_1	7	4	19
	$T_{2,3}$	R_2	1	11	20
P_3	$T_{3,1}$	R_1	2	1	12
	$T_{3,2}$	R_3	4	3	16
	$T_{3,3}$	R_2	4	7	20

The initial precedence cost vectors $\mathbf{c}_{i,j}$ and estimated deviation cost vectors $\hat{\theta}_{i,j}$ of individual tasks are as shown in Table A.3 and A.4. All elements in $\mathbf{c}_{i,j}$ are set to zero because no precedence violation was detected so far. The elements of $\hat{\theta}_{i,j}$ are calculated by Equation (4.11). Because there is no initial allocation (or schedule), no "truncated" deviation cost (see Figure 4.9(c)) shows up in the table. Each task agent's bid (bundle-utility pairs) for resources are generated based on the Equation 4.2, summarized in Table A.5. In the table the column numbers mean the completion time slot index of the bundles. Because the precedence cost vectors are all zero, the bidding values of the bundles are exactly same to the elements of the $-\hat{\theta}_{i,j}$'s in Table A.4. For the convenience, we add 10 to each utility value to make it positive.

Based on the bids, each resource agent solves the local market clearing problem using the algorithm explained in Section 4.5.2. The results are as shown in Table A.6. In the table, W_j is the welfare function, which is basically the weighted sum of the utility values of winning bids. As shown in Table A.6, two projects, P_1 and P_2, already got an feasible project schedule, meaning that no precedence cost will be adjusted for these two projects. The precedence cost $\mathbf{c}_{3,k}$'s are calculated as follows using the rule in Section 4.4.5.

Table A.3. Precedence cost vectors after critical path analysis.

P_i	$\mathbf{c}_{i,j}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P_1	$\mathbf{c}_{1,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-
	$\mathbf{c}_{1,2}$	-	-	-	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-
	$\mathbf{c}_{1,3}$	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P_2	$\mathbf{c}_{2,1}$	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-
	$\mathbf{c}_{2,2}$	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
	$\mathbf{c}_{2,3}$	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0
P_3	$\mathbf{c}_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-
	$\mathbf{c}_{3,2}$	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-
	$\mathbf{c}_{3,3}$	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.4. Estimated deviation cost vectors after critical path analysis.

P_i	$\hat{\theta}_{i,j}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P_1	$\hat{\theta}_{1,1}$	-	-	0	1	2	3	4	5	6	7	8	9	10	-	-	-	-	-	-	-
	$\hat{\theta}_{1,2}$	-	-	-	0	1	2	3	4	5	6	7	8	9	10	-	-	-	-	-	-
	$\hat{\theta}_{1,3}$	-	-	-	-	-	-	-	-	-	0	1	2	3	4	5	6	7	8	9	10
P_2	$\hat{\theta}_{2,1}$	-	-	1	2	3	4	5	6	7	8	9	10	-	-	-	-	-	-	-	-
	$\hat{\theta}_{2,2}$	-	-	-	-	-	-	-	-	-	1	2	3	4	5	6	7	8	9	10	-
	$\hat{\theta}_{2,3}$	-	-	-	-	-	-	-	-	-	-	1	2	3	4	5	6	7	8	9	10
P_3	$\hat{\theta}_{3,1}$	-	0	0	0	1	2	3	4	5	6	7	8	9	10	-	-	-	-	-	-
	$\hat{\theta}_{3,2}$	-	-	-	-	-	0	0	0	1	2	3	4	5	6	7	8	-	-	-	-
	$\hat{\theta}_{3,3}$	-	-	-	-	-	-	-	-	-	0	0	0	1	2	3	4	5	6	7	8

Table A.5. Each task agent's bids for resources in the first iteration: Each cell holds the utility of task $T_{i,k}$ for the bundle $B_{l-\delta_{i,k}+1,l}$ of resource R_j, where $\delta_{i,k}$ is the duration time for $T_{i,k}$.

P_i	$T_{i,k}$	R_j	$l =$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P_1	$T_{1,1}$	R_1	-	-	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-
	$T_{1,2}$	R_2	-	-	-	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-
	$T_{1,3}$	R_3	-	-	-	-	-	-	-	-	-	10	9	8	7	6	5	4	3	2	1	0
P_2	$T_{2,1}$	R_3	-	-	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-	-
	$T_{2,2}$	R_1	-	-	-	-	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0	-
	$T_{2,3}$	R_2	-	-	-	-	-	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0
P_3	$T_{3,1}$	R_1	-	10	10	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-
	$T_{3,2}$	R_3	-	-	-	-	-	10	10	10	9	8	7	6	5	4	3	2	-	-	-	-
	$T_{3,3}$	R_2	-	-	-	-	-	-	-	-	-	10	10	10	9	8	7	6	5	4	3	2

Table A.6. The allocation in the first iteration: Each cell holds the project index i for the project P_i. W_j denotes the welfare value of the resource R_j.

R_j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	W_j
R_1	1	1	1	2	2	2	2	2	2	2	3	3									98
R_2				1							2	3	3	3	3						108
R_3	2	2	2		1	1	1	1	1	1	3	3	3	3							104

$$\begin{aligned}
\mathbf{c}_{3,1} &= (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)^T \\
\mathbf{c}_{3,2} &= (1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0)^T \\
\mathbf{c}_{3,3} &= (1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0)^T
\end{aligned} \tag{A.1}$$

From the second iteration, changes happen only in P_3, because (1) P_1 and P_2 already get feasible solutions, meaning that no precedence cost will change and (2) from the resources viewpoints any possible bids from P_3 cannot change the current allocation of $T_{1,k}$'s and $T_{2,k}$'s due to the weight distribution. Hence, just updating tables regarding P_3 is enough for the later iterations. Table A.7 to A.10 summarize the results of following iterations. Here we do not take into account the commitment windows for simplicity. In the tables, boldfaced slots denote the optimal allocations of each local market. As shown in Table A.10, project groups get a feasible solution and P-TÂTO procedure stops. The generated schedule is optimal with respect to weighted sum of deviation costs.

Table A.7. Second iteration summary of P_3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathbf{c}_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,2}$	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,3}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
$c_{3,1}(B)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,2}(B)$	0	2	3	4	4	4	4	4	4	4	4	4	3	2	1	0	0	0	0	0
$c_{3,3}(B)$	0	2	3	4	4	4	4	4	4	4	4	4	4	4	3	2	1	0	0	0
$\hat{\theta}_{3,1}$	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\hat{\theta}_{3,2}$	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12
$\hat{\theta}_{3,3}$	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8
$u_{3,1}(B)$	10	10	10	10	9	8	7	6	5	4	**3**	**2**	1	0	-1	-2	-3	-4	-5	-6
$u_{3,2}(B)$	-	8	7	6	6	6	6	6	5	4	**3**	**2**	**2**	**2**	2	2	1	0	-1	-2
$u_{3,3}(B)$	-	8	7	6	6	6	6	6	6	6	6	**6**	**5**	**4**	**4**	4	4	4	3	2

Table A.8. Third iteration summary of P_3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathbf{c}_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,2}$	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,3}$	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0
$c_{3,1}(B)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,2}(B)$	-	4	6	8	8	8	8	8	8	8	8	8	6	4	2	0	0	0	0	0
$c_{3,3}(B)$	-	4	6	8	8	8	8	8	8	8	8	8	8	8	6	4	2	0	0	0
$\hat{\theta}_{3,1}$	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\hat{\theta}_{3,2}$	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12
$\hat{\theta}_{3,3}$	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8
$u_{3,1}(B)$	10	10	10	10	9	8	7	6	5	4	**3**	**2**	1	0	-1	-2	-3	-4	-5	-6
$u_{3,2}(B)$	-	6	4	2	2	2	2	2	1	0	-1	-2	**-1**	**0**	**1**	**2**	1	0	-1	-2
$u_{3,3}(B)$	-	6	4	2	2	2	2	2	2	2	2	2	1	0	**1**	**2**	**3**	**4**	3	2

Table A.9. Fourth iteration summary of P_3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathbf{c}_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,2}$	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,3}$	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	1	0	0	0	0
$c_{3,1}(B)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,2}(B)$	-	4	6	8	8	8	8	8	8	8	8	8	6	4	2	0	0	0	0	0
$c_{3,3}(B)$	-	6	9	12	12	12	12	12	12	12	12	12	12	12	10	8	5	2	1	0
$\hat{\theta}_{3,1}$	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\hat{\theta}_{3,2}$	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12
$\hat{\theta}_{3,3}$	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8
$u_{3,1}(B)$	10	10	10	10	9	8	7	6	5	4	**3**	**2**	1	0	-1	-2	-3	-4	-5	-6
$u_{3,2}(B)$	-	6	4	2	2	2	2	2	1	0	-1	-2	**-1**	**0**	**1**	**2**	1	0	-1	-2
$u_{3,3}(B)$	-	4	1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-3	-4	**-3**	**-2**	**0**	**2**	2	2

Table A.10. Fifth iteration summary of P_3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathbf{c}_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,2}$	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0
$\mathbf{c}_{3,3}$	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	0	0	0	0
$c_{3,1}(B)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_{3,2}(B)$	-	4	6	8	8	8	8	8	8	8	8	8	6	4	2	0	0	0	0	0
$c_{3,3}(B)$	-	8	12	16	16	16	16	16	16	16	16	16	16	16	14	12	8	4	2	0
$\hat{\theta}_{3,1}$	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\hat{\theta}_{3,2}$	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12
$\hat{\theta}_{3,3}$	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8
$u_{3,1}(B)$	10	10	10	10	9	8	7	6	5	4	**3**	**2**	1	0	-1	-2	-3	-4	-5	-6
$u_{3,2}(B)$	-	6	4	2	2	2	2	2	1	0	-1	-2	**-1**	**0**	**1**	**2**	1	0	-1	-2
$u_{3,3}(B)$	-	2	-2	-6	-6	-6	-6	-6	-6	-6	-6	-6	-7	-8	-7	-6	**-3**	**0**	**1**	**2**

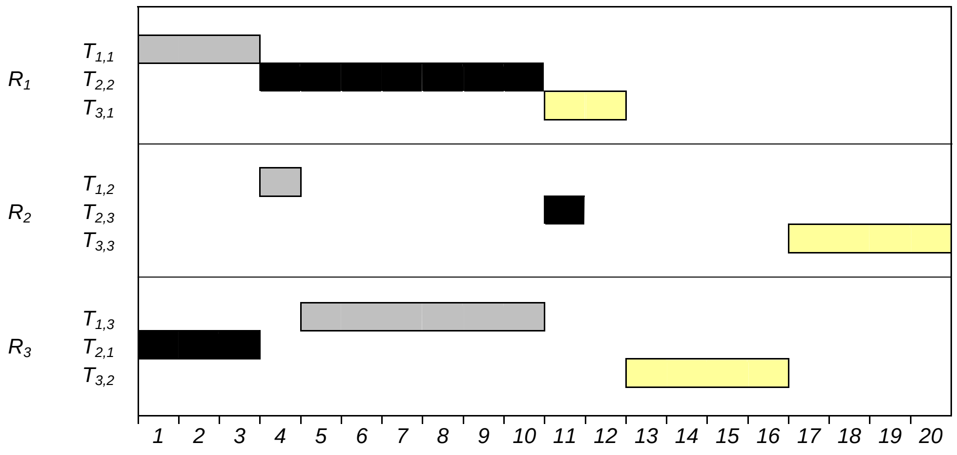


Fig. A.1. Final schedule generated by P-TÂTO.

Appendix B

FIPA SL0

The FIPA SL (Semantic Language) is a very expressive language, but for some agent communication tasks it is unnecessarily powerful. This expressive power has an implementation cost to the agent and introduces problems of the decidability of modal logic. To allow simpler agents, or agents performing simple tasks, to do so with minimal computational burden, FIPA introduced semantic and syntactic subsets of the full FIPA SL content language for use by the agent when it is appropriate or desirable to do so. They are FIPA-SL0, FIPA-SL1 and FIPA-SL2. FIPA SL0 is the minimal subset of FIPA SL, denoted by the normative constant `FIPA-SL0` in the `:language` parameter of an ACL message. It allows the representation of actions, the determination of the result a term representing a computation, the completion of an action and simple binary propositions. The following defines the FIPA SL0 grammar (Details can be found in (FIPA, 2000)):

```
Content           =  "(" ContentExpression+ ")".
ContentExpression =  ActionExpression
                  |  Proposition.
Proposition       =  Wff.
Wff               =  AtomicFormula
                  |  "(" ActionOp ActionExpression ")".
AtomicFormula     =  PropositionSymbol
                  |  "(" "result" Term Term ")"
                  |  "(" PredicateSymbol Term+ ")"
                  |  "true"
                  |  "false".
ActionOp          =  "done".
Term              =  Constant
```

```
                   |  Set
                   |  Sequence
                   |  FunctionalTerm
                   |  ActionExpression.
ActionExpression  =  "(" "action" Agent Term ")".
FunctionalTerm    =  "(" FunctionSymbol Term* ")"
                   |  "(" FunctionSymbol Parameter* ")".
Parameter         =  ParameterName ParameterValue.
ParameterValue    =  Term.
Agent             =  Term.
FunctionSymbol    =  String.
PropositionSymbol =  String.
PredicateSymbol   =  String.
Constant          =  NumericalConstant
                   |  String
                   |  DateTime.
Set               =  "(" "set" Term* ")".
Sequence          =  "(" "sequence" Term* ")".
NumericalConstant =  Integer
                   |  Float.
```

□

Bibliography

Laurence M. Ausubel. An effective ascending-bid auction for multiple objects. Working Paper No. 97-06, Department of Economics, University of Maryland, College Park, MD, College Park, MD, 1997.

Laurence M. Ausubel. A generalized Vickrey auction. Working paper, Department of Economics, University of Maryland, College Park, MD, College Park, MD, 1999.

Albert D. Baker. Metaphor or reality: A case study where agents bid with actual costs to schedule a factory. In Scott H. Clearwater, editor, *Market-Based Control - A Paradigm for Distributed Resource Allocation*, chapter 8, pages 184–223. World Scientific, Singapore, 1996.

Fabio Bellifemine, Agostino Poggi, and Giovanni Rimanssa. JADE - a FIPA-compliant agent framework. Internal technical report, CSELT S.p.A, 1999.

Hemant Bhargava, Kalyan Chatterjee, Soundar Kumara, and Yong-Han Lee. Dynamic coalition formation and search, with applications to distibuted resource allocation: A selective survey. *Management Science*, Special Issue on eBusiness and OR/MS, 2002. to appear.

Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, April 1986.

Donald E. Campbell. *Resource Allocation Mechanisms.* Cambridge University Press, New York, NY, 1987.

Kathleen M. Carley and Les Gasser. Computational organization theory. In Gerhard Weiss, editor, *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, chapter 7, pages 299–330. The MIT Press, Cambridge, MA, 1999.

Ai-Mei Chang, Andrew D. Bailey, and Andrew B. Whinston. A distributed knowledge-based approach for planning and controlling projects. *IEEE Transactions of Systems, Man, and Cybernetics*, 23(6):1537–1550, November 1993.

Kalyan Chatterjee. Taxonomy of resource allocation mechansims. Manuscript, Department of Economics, The Pennsylvania State University, University Park, PA, February 2002.

Deepika Chauhan and Albert D. Baker. JAFMAS: A multiagent application development system. In Michael Wooldridge and Tim Finin, editors, *Proceedings of Second ACM Conference on Autonomous Agents*, St. Paul, Minnepolis, 1998. ACM Press.

John Q. Cheng and Michael P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12(1):1–24, 1998.

Scott H. Clearwater, editor. *Market-Based Control - A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore, 1996.

Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

Danial D. Corkill and Susan E. Lander. Diversity in agent organization. Technical paper, Blackboard Technology, 1998.

R. Scott Cost, Tim Finin, Yannis Labrou, Xiaochang Luan, Yun Peng, Ian Soboroff, James Mayfield, and Akram Boughannam. Jackal: A Java-based tool for agent development. In *AAAI-98 Workshop on Software Tools for Developing Agents*, Madison, WI, 1998. AAAI Press.

Peter Cramton and Laurence M. Ausubel. Vickrey auctions with reserve pricing. Working paper, Department of Economics, University of Maryland, College Park, MD, College Park, MD, 1999. Preliminary version.

Mark R. Cutkosky, Robert S. Engelmore, Richard E. Fikes, Michael R. Genesereth, and Thomas R. Gruber. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, 26(1):28–38, January 1993.

Neil J. Davies. *Truth, Modality, and Action.* PhD thesis, Department of Computer Science, University of Essex, Colchester, UK, 1993.

Daniel C. Donnett. *The Intentional Stance.* The MIT Press, Cambridge, MA, 1987.

Brian Drabble. Artificial intelligence for project planning. In *Proceedings of the Colloquium on Future Developments in Project Management Systems*, pages 3/1–3/5, Savoy Place, London, UK, 1995. IEE.

Andreas Drexl. Scheduling of project networks by job assignment. *Management Science*, 37(12):1590–1602, 1991.

Hans-Erik Eriksson and Magnus Penker. *UML Toolkit.* John & Sons, Inc, New York, NY, 1998.

Donald Ferguson, Yechiam Yemini, and Christos Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. In *The Eighth International Conference on Distributed Computing Systems*, pages 491–499, 1988.

Tim Finin and Jay Weber. Specification of the KQML - agent-communication language. Draft specification, The DARPA Knowledge Sharing Initiative External Interface Working Group, 1993.

FIPA. FIPA 97 specification, version 2.0, part 2, Agent Communication Language. Specification, Foundation for Intelligent Physical Agents, October 1998.

FIPA. FIPA sl content language specification. Specification, Foundation for Intelligent Physical Agents, September 2000.

Michael Fisher. A survey of Concurrent MetateM - the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Proceedings of the First International Conference on Temporal Logic (LNAI Volume 827)*, pages 480–505. Springer-Verlag, Berlin, Germany, 1994.

Ernest J. Friedman-Hill. *Jess, The Java Expert System Shell.* Sandia National Laboratories, Livermore, CA, January 2000. Version 5.0.

Robert H. Frost and Mark R. Cutkosky. Design for manufacturability via agent interaction. In *Design for Manufacturing Conference*, pages 18–22, Irvine, CA, August 1996. ASME.

Michael R. Genesereth and Richard E. Fikes. Knowledge interchange format version 3.0 reference manual. Draft specification, The Interlingua Working Group of the DARPA Knowledge Sharing Effort, 1992.

Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle,WA, 1987.

Andrew Haas. A syntactic theory of belief and knowledge. *Artificial Intelligence*, 28(3): 245–292, 1986.

Aimo Hinkkanen, Ravi Kalakota, P. Saengcharoenrat, J. Stallaert, and A. B. Whinston. Distributed decision support systems for real-time supply chain management using agent technologies. In Ravi Kalakota and Andrew B Winston, editors, *Readings in Electronic Commerce*, chapter 12. AW Computer and Engineering Publishing Group, 1997.

Michael N. Huhns and Munindar P. Singh. All agents are not created equal. *IEEE Internet Computing*, 2(3):94–96, 1998.

Nick Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

Nick Jennings and Nir Vulkan. Efficient mechanisms for the supply of services in multi-agent environments. *International Journal of Decision Support Systems*, 28(1-2):5–19, 2000.

Heecheol Jeon, Charles J. Petrie, and Mark R. Cutkosky. JATLite: A Java agent infrastructure with message routing. *IEEE Internet Computing*, pages 87–96, March/April 2000.

Yan Jin and Stephen C.-Y. Lu. An agent-supported approach to collaborative design. *Annals of the CIRP*, 47(1):107–110, 1998.

Rinaldo A. Jose and Lyle H. Ungar. Auction-driven coordination for plantwide optimization. In *Proceedings of Foundations of Computer-Aided Process Operation (FOCAPO)*, Snowbird, UT, 1998.

Rainer Kolisch and Rema Padman. An integrated survey of deterministic project schedulingz. *Omega - International Journal of Management Science*, 29(3):249–272, 2001.

Rainer Kolisch and Arno Sprecher. PSPLIB- a project scheduling problem library. Technical Report 396, Institut für Betriebswirtschaftslehre der Universität Kiel, Germany, 1996.

Rainer Kolisch, Arno Sprecher, and Andreas Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. Technical Report 301, Institut für Betriebswirtschaftslehre der Universität Kiel, Germany, 1992.

Kurt Konolgie. *A Deduction Model of Belief.* London and Morgan Kaufmann, San Mateo, CA, 1986.

Vijay Krishna and Motty Perry. Efficient mechanism design. Working paper, Department of Economics, The Pennsylvania State University, University Park, PA, April 1998.

Erhan Kutanoglu and S. David Wu. On combinatorial auction and lagrangean relaxation for distributed resource scheduling. Technical Report 97T-012, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, 1997.

Yannis Labrou and Tim Finin. Semantics and conversations for an agent communication language. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 235–242. Morgan Kaufmann, San Fransisco, CA, 1997.

Yannis Labrou, Tim Finin, and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, 1999.

Susan E. Lander. Issues in multiagent design systems. *IEEE Expert*, 12(2):18–26, March-April 1997.

Susan E. Lander and Danial D. Corkill. Design integrated engineering environment: Blackboard-based integration of design and analysis tools. *Concurrent Engineering*, 4 (1):59–71, March 1996.

Jaeho Lee, Marcus J. Huber, Edmund H. Durfee, and Patrick G. Kenny. UM-PRS: an implementation of the procedural reasoning system for multirobot applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS '94)*, Houston, TX, March 1994.

Yong-Han Lee and Soundar R.T. Kumara. Market-based collaborative control of distributed multiple product development projects. In Nina M. Berry, editor, *Proceedings of SPIE Vol. 4208*, pages 73–83, 2000.

Yves Lespérance, Hector J. Levesque, Fangzhen Lin, Daniel Marcu, Raymond Reiter, and Richard B. Scherl. Foundations of a logical approach to agent programming. In Michael Wooldridge, Jörg P. Müller, and Milind Tambe, editors, *Intelligent Agents II (LNAI Volume 1937)*, pages 331–346. Springer-Verlag, Berlin, Germany, 1996.

Hector J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, Austin,TX, 1984.

Feng Liu, Peter B. Luh, and Bryan Moser. Scheduling and coordination of distributed design projects. *Annals of the CIRP*, 47(1):111–114, 1998.

Peter B. Luh, Feng Liu, and Bryan Moser. Scheduling of design projects with uncertain number of iterations. *European Journal of Operational Research*, 113:575–592, 1999.

Jeffrey K. MacKie-Mason and Hal R. Varian. Generalized Vickrey auctions. Working paper, Department of Economics, University of Michigan, Ann Arbor, MI, 1994.

Leora Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings of the Tenth International Conference on Artificial Intelligence (IJCAI-87)*, pages 867–874, Milan, Italy, 1987.

Jörg P. Müller, Markus Pische, and Michael Thiel. Modelling reactive behaviour in vertically layered agent architectures. In Michael Wooldridge and Nick R. Jennings, editors, *Intelligent Agents: Theorys, Architectures, and Languages (LNAI Volume 890)*, pages 261–276. Springer-Verlag, Berlin, Germany, 1995.

Hyachinth S. Nwana, Divine T. Ndumu, and Lyndon C. Lee. ZEUS: An advanced tool-kit for engineering distributed multi-agent systems. In *Proceedings of the Third International Conference and Exhibition on The Practical Application of Intelligent Agent and Multi-Agents*, pages 377–391, London, U.K., 1998.

H. Van Dyke Parunak. Industrial and practical applications of DAI. In Gerhard Weiss, editor, *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, chapter 9, pages 377–421. The MIT Press, Cambridge, MA, 1999.

James H. Patterson. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management Science*, 30(7):854–867, 1984.

Yun Peng, Tim W. Finin, Bill Chu, J. Long, Bill Tolone, and Akram Boughannam. Multi-agent system for enterprise integration. In *Proceedings of the 3rd international Conference on Practical Applications of Intelligent Agents and Multi-Agents (PAAM)*, pages 533–548, London, UK, 1998.

Charles J. Petrie. Process coordination. URL `http://cdr.stanford.edu/ProcessLink/papers/white-dpm.html`. a white paper, 1998.

Charles J. Petrie, Teresa A. Webster, and Mark R. Cutkosky. Using Pareto optimality to coordinate distributed agents. *AI for Engineering Design, Analysis and Manufacturing*, 9(4):269–282, 1995. Special issue on conflict management in design.

Charles J. Petriea, Sigrid Goldmann, and Andreas Raquet. Agent-based project management. In Michael Wooldridge and Manuela Veloso, editors, *Artificial intelligence today : recent trends and developments (LNAI Volume 1600)*. Springer-Verlag, Berlin, Germany, 1999.

Jie Qian. Algorithms for software project scheduling. Master's thesis, Department of Industrial Engineering, The Pennsylvania State University, University Park, PA, 1998.

Anand S. Rao and Michael P. Georgeff. Asymmetry thesis and side-effect problems in linear time and branching time intention logics. In *Proceedings of the Twelfth International Conference on Artificial Intelligence (IJCAI-91)*, pages 498–504, Sydney, Australia, 1991a.

Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Readings (KR&R-91)*, pages 473–484, 1991b.

Anand S. Rao and Michael P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence (IJCAI-93)*, pages 318–324, Chambéry, France, 1993.

Tuomas Sandholm. An implementation of Contract Net Protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., 1993. The MIT Press.

Goutam Satapathy. *Distrubuted and collaborative logistics planning and replanning under uncertainty: a multiagent based approach.* PhD thesis, The Pennsylvania State Univeristy, University Park, PA, 1999.

John A. Sauter, H. Van Dyke Parunak, and John Goic. ANTS in the supply chain. In *Workshop on Agent for Electronic Commerce at Agents '99*, pages 1–5, Seattle, WA, May 1999.

Weiming Shen and Douglas H. Norrie. Agent-based systems for intelligent manufacturing: State-of-the-art survey. *International Journal of Knowledge and Information Systems*, 1(2):129–156, 1999.

Munindar P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications.* LNAI Volume 799. Springer-Verlag, Heidelberg, Germany, 1994.

Reid G. Smith. The contact net protocol: high-level communication and control in a distrubuted problem solver. *IEEE Transaction on Computers*, 29(12):1104–1113, 1980.

SRI International. *Procedural Reasoning System - User's Manual.* Artificial Intelligence Center, SRI International, March 1999. Version 1.96.

Gek Woo Tan, Caroline C. Hayes, and Michael Shaw. An intelligent-agent framework for concurrent product design and planning. *IEEE Trans. on Engineering Management*, 43(3):297–306, August 1996.

Jui Chiew Tan and Patrick T. Harker. Designing workflow coordination: Centralized versus market-based mechanisms. *Information Systems Research*, 10(4):328–342, 1999.

Kevin J. Tilley. Machining task allocation in discrete manufacturing systems. In Scott H. Clearwater, editor, *Market-Based Control - A Paradigm for Distributed Resource Allocation*, chapter 9, pages 224–251. World Scientific, Singapore, 1996.

Hal R. Varian. *Microeconomic Analysis.* W. W. Norton & Company, New York, NY, 1984.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

Sven de Vries and Rakesh Vohra. Combinatorial auctions: A brief survey (draft). Working paper, Department of Managerial Economics and Decision Sciences, Northwestern University, Evanston, IL, January 2000.

Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed compuational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.

Léon Walras. *Elements of Pure Economics.* Allen and Unwin, London, UK, 1954.

William E. Walsh, Michael P. Wellman, Peter R. Wurman, and Jeffrey K. MacKie-Mason. Some economics of market-based distributed scheduling. In *The Eighteenth International Conference on Distributed Computing Systems*, pages 612–621, Amsterdam, The Netherlands, May 1998.

Wlliam E. Walsh and Michael P. Wellman. A market protocol for decentralized task allocation. In *The Third International Conference on Multi-Agent Systems*, pages 325–332, Seattle, WA, 1998.

Gerhard Weiss, editor. *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA, 1999.

Michael P. Wellman. A computational market model for distributed configuration design. *AI for Engineering Design, Analysis and Manufacturing*, 9:125–133, 1995.

Michael Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems.* PhD thesis, UMIST, Manchester, UK, 1992.

Michael Wooldridge and Nick R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24 (1):17–27, 1998.

Vita

Yong-Han Lee was born in Korea in 1965. He received his B.S. and M.S. degrees in Industrial Engineering from Seoul National University and Korea Advanced Institute of Science and Technology (KAIST) in 1988 and 1990 respectively. During his B.S. and M.S. study his major research interest was in the area of computer aided design and manufacturing. After he finished his M.S. study he joined Daewoo Motor Co., Inchon, Korea. For 6.5 years in the Tech Center of the company, he mostly worked in computer aided design and manufacturing, computer integrated manufacturing and rapid prototyping areas. In 1997 fall, he enrolled in the Ph.D. program in Industrial Engineering at the Pennsylvania State University. During his Ph.D. study he was employed as a research assistant in the department of Industrial Engineering, participating in two major research projects: (1) "National Infrastructure Emergency Warning System" funded by Army Research Office and (2) "Simulation Framework for Evaluating Order-to-Delivery Systems and Processes" funded by General Motors Research and Development Center, under the supervision of Dr. Soundar Kumara. Based on the achievement in the projects, he received a certificate of achievement from US Army War College, Carlisle, PA and a graduate research fellowship from General Motors, Warren, MI. He also taught senior level information systems analysis and design course in the department of Management Science and Information Systems in Smeal College of Business Administration at the Pennsylvania State University as a part-time instructor for two semesters in Fall 2001 and Spring 2002. His current research interests include multiagent based information systems theory and application, decentralized resource allocation using market mechanisms, and automated negotiation based on economic and computational models, all in the context of e-Business and e-Manufacturing.