Metaphor-based Negotiation and Its Application in AGV Movement Planning

Xueyue Huang

Ph.D.
University of Edinburgh
1996

Abstract

The theme of this thesis is "metaphor-based negotiation". By metaphor-based negotiation I mean a category of approaches for problem-solving in Distributed Artificial Intelligence (DAI) that mimic *some aspects* of human negotiation behaviour. The research in this dissertation is divided into two closely related parts. Cooperative interaction among agents in a *multiagent system* (MAS) is discussed in general, and the discussion leads to a formal definition of metaphor-based negotiation. Then, as a specific application, a "spring-based" computational model for metaphor-based negotiation is developed as an approach to solving movement planning, specifically the AGV scheduling problem (AGVSP) — determing the timings of AGVs' activities, of automated guided vehicles (AGVs) in a factory.

By formally addressing the multi-agent cooperative interaction problem and assuming that agents in a MAS are rational, benevolent and fully informed, an initial strategy set of cooperative interaction can be reduced to a strategy set by eliminating strategies that are irrational in a group sense. However, it is proved in this dissertation that, in the remaining strategy set, no unique strategy can be found that is acceptable to all agents according their individual preferences. More specifically, in this smaller strategy set, if one agent moves from one strategy to another in an attempt to better its individual goal achievement, then there is at least one agent whose goal achievement will be negatively affected by such a move. So, the cooperative interaction problem can only be partially solved if no further knowledge is given to those agents. The idea of a *common sense principle* is introduced in this dissertation to overcome the deficiencies of the assumptions of rationality, benevolence and full-informedness.

In reality, the assumption of full-informedness of agents may not be practical. Communication is needed for agents to (1) exchange their local problem solving information, and (2) exchange proposals for global problem solving, when their views are in conflict. Based on the discussion of cooperative interaction, a formal definition of metaphor-based negotiation is proposed to formally indicate what is a proposal and what is the condition for accepting a proposal from another agent. In this definition, the common sense principle is one of the most important features, not found in definitions of negotiation available so far in the literature, which guides agents to find an agreement when negotiation is running into difficulties.

The AGVSP involves timing activities for each AGV in a AGV-based factory. The AGVSP is naturally distributed: the whole problem can be easily divided into several subproblems each of which involves timing of activities of one AGV. Therefore, it is intuitively straightforward for us to seek DAI approaches to solving the AGVSP. In spired by Kwa's Iterative Negotiation Model [Kwa 88b][Kwa 88a] for the AGVSP, we developed a spring-based (metaphor-based) negotiation model for the AGVSP to overcome some vital problems in Kwa's model. The idea of the spring-based negotiation model is described below:

The AGVSP can be regarded as a Distributed Constraint Satisfaction Problem (DCSP) and solved in a MAS. Each agent in the MAS is designed to solve a subproblem — a local scheduling problem which is a small Constraint Satisfaction Problem (CSP). Con-

flicts exist when *intra-agent constraints* or *inter-agent constraints* are violated. These constraints can be classified into hard constraints— those that can not be relaxed at the agent level unless the system designer permits (e.g., by providing an arbitrator), and *soft constraints* — those that can be relaxed at the agent level when necessary. When agents are in conflict, i.e, when some inter-agent constraints are violated (or say, when one agent's timings of its activities overlap those of some other agents), these agents involved will resolve the conflicts through a (metaphor-based) negotiation procedure in which conflicts will be gradually resolved by each agent's relaxation of its intra-agent constraints, i.e, by yielding some amount of its initially allocated resources to other agents or by shifting its initially allocated resources. The negotiation can be viewed as a process of exchanging proposals (of cooperative strategies) between conflicting agents, where a cooperative strategy is a possible resolution to a conflict according to the viewpoint of the proposing agent. However, since agents are designed to be rational, each agent that is involved in the conflicts will try hard to relax its intra-agent constraints as little as possible. Further, it is reasonably acceptable that the more an intra-agent constraint has been relaxed the less the respective agent is willing to relax it further. This feature can be modeled by a spring — the more it has been compressed the harder it is to compress it further. Based on this inspiration, a spring-based computational model of metaphor-based negotiation is proposed: each agent's local schedule is represented by a local spring network in which each spring element represents a soft intra-agent constraint. Relaxation of an intra-agent constraint is likened to a spring being compressed by external forces from other agents. As a consequence, the compressed spring will also show a reacting force upon those compressing agents. An agreement will be reached when those forces and reacting forces are balanced. This is the common sense principle in the spring-based negotiation. The model solves some key issues, e.g., how to select negotiation techniques and skills during the process of negotiation, that have not been solved by Kwa's iterative negotiation model. Some experimental evidence of the value of this model is presented.

Acknowledgements

First of all, I would like to thank my two supervisors, John Hallam and Austin Tate, for their valuable and patient guidance. John introduced me to Kwa's work on AGV scheduling and directed me gradually into the DAI research field. Austin has always been there to help me to ensure that my research direction is correct.

Secondly, I would like thank those who proof-read this thesis or proof-read my published papers. They are David Wren, Andrew Tuson, Thomas Reed.

During my Ph.D period, many teachers have helped me in regard to the thesis. For example, Horward Beck helped me to have a seminar in the scheduling group led by him. Dave Corne spent lots of time on comparing my model with GA scheduling approaches. Peter Ross pointed out that there might be some work similar to my spring model and let his students survey some relevent references for me. ... I feel that it would be nearly impossible for a student to finish his thesis in an isolated environment.

Social life is another part of my Ph.D life. During my Ph.D time, I have made many friends inside and outside the department. Ashley Walker, David Wren made me more confident to communicate with local people. They correct my spoken English whenever they spotted any mistakes from me. I made several new Chinese friends here in Edinburgh too. Just mention some of them: Weiyuan Zhang, Changqing Zhu, Xiaofeng Li, Yongji Wang. Some of them also helped me in regard to the thesis. For example, in the final stage of the thesis, Weiyuan's family looked after my daughter and made it possible for me to concentrate on my thesis. Yongji Wang often came to exchange his viewpoints on my thesis. Hengchang Dai helped me in great deal to bind this these. I would like to use this opportunity to thank them.

John Polak, who is my current boss in Imperial College where I have been working for half years and will continue to work after I submited the thesis, made a very good arragement so that I could resume my thesis. I would like to thank him for his help.

Thanks go also to the Sino-British Friendship Studentship and Scholarship (SBFSS) which supported me to study here on the project.

Finally, I would like to thank particarly my wife, Yan Lu, and my daughter, Weiman (Mandy) Huang, for their accompanying me during my Ph.D life.

Papers Published

Defining Metaphor-based Negotiation, X. Huang, *Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics: Intelligent Systems for the 21st Century*, P.383-8 vol. 1, ISBN: 0 7803 2559 1, 1995

Spring-based Negotiation for Conflict Resolution in AGV Scheduling, X. Huang and J. Hallam, *Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics: Intelligent Systems for the 21st Century*, P.789-94, ISBN: 0 7803 2559 1, 1995

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Xueyue Huang
Edinburgh
May 13, 1997

Glossary of Acronyms

AGV	Automated Guided Vehicle
AGVSP	AGV Scheduling Problem
TA	Task Assignment
RP	Route Planning
DAI	Distributed Artificial Intelligence
DPS	Distributed Problem Solving
PAI	Parallel Artificial Intelligence
MAS	MultiAgent System
JSSP	Job Shop Scheduling Problem
CIP	Cooperative Interaction Problem
CSP	Constraint Satisfaction Problem
DCSP	Distributed CSP
CDPS	Cooperative Distributed Problem Solving
RNM	Recursive Negotiation Model
RToken	Recorded Token
JIT	Just-In-Time

Contents

List of Figures

Chapter 1

Introduction

The theme of this dissertation is metaphor-based negotiation. By metaphor-based negotiation I mean a category of approaches for problem-solving in Distributed Artificial Intelligence (DAI) that mimic human negotiation behaviour in some aspects. The research in this dissertation is divided into two closely related parts. In general, I shall discuss cooperative interaction between agents with their individual goals which are in conflict. This discussion will lead to a formal definition of metaphor-based negotiation (see Chapter 2, Chapter 3, and Chapter 4). specifically, based on this formal definition, I shall propose a negotiation model for movement planning of automated guided vehicles (AGVs) (Chapter 5, Chapter 6 and Chapter 7).

1.1 AGV Movement Planning

1.1.1 AGV Systems

The evolution of unmanned production in manufacturing of medium and low volume machine saw the development of flexible manufacturing systems which possess much higher economic advantages in production with reduced crew and increased efficiency. In an unmanned factory, one problem which must be solved is the automation of material flow from one place to another, for example, the movement of raw materials from a warehouse to machine tools and the ready parts back to warehouse. One economic solution to this problem is given by automatic guided vehicle systems. AGV systems can be used for computerised random linking of quite different working places. Today, various AGV systems for unmanned production are already available [Valery 87] [Elbracht & Plum 88]. AGV *movement planning* involves generating a set of AGV movement schedules which define the paths and the movement timings of the AGVs [Kwa 88a] [Kwa 88b]. More specifically, AGV movement planning has three major subproblems to achieve:

1. *task assignment* concerns the selection of an AGV to undertake a given task, such that a set of given tasks are assigned to the available AGVs optimally (i.e., cost for completing all the tasks, such as the overall distance required, time spent, energy consumed, etc., is minimised;

2. *route planning* concerns the selection of a route for each tasked AGV such the cost for complete all the tasks is minimised;

3. *movement timing* involves timing activities for each AGV, which has been assigned a task during task assignment and has been selected a route during route planning, to carry out the activities. Activities are operations at stations such as loading at, unloading at, and passing by stations. The objective of the movement timing is to meet the various requirements specified in tasks while meeting the requirement of collision-free movements.

Naturally, the order of solving the three subproblems is task assignment → route planning → movement timing, as illustrated in Fig 1.1. However, when a later subproblem, e.g., route planning, cannot be solved, then that problem-solver may report the failure to the former problem-solver, e.g., the one for task assignment, and seek alternative solutions. Loops between subproblems are possible and they are indicated by Re-TA, Re-RP in Fig 1.1, meaning *task re-assignment* and *route re-planning* respectively. We call the movement timing problem the AGV *scheduling problem* (AGVSP), since it is similar to many scheduling problems, such as the *job shop scheduling problems* (JSSPs), that involve allocating times to activities.

This dissertation focuses on the AGVSP.

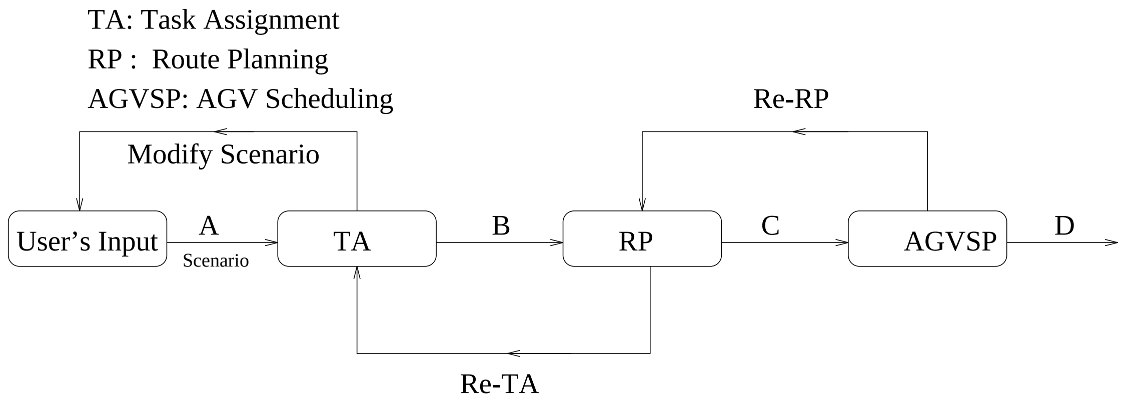


Figure 1.1: The AGV Movement Planning System

1.1.2 The AGVSP

The AGVSP involves timing activities for each AGV, which has been assigned a task during task assignment and has been selected a route during route planning, to carry out the activities. Activities are operations at stations such as loading at, unloading at, and passing by stations. For example, AGV a has been selected a route

$$(A,\ B,\ C) \tag{1.1}$$

where A, C are stations, B is a road junction, and there are traversable paths from A to B and from B to C. The task assigned to it is broken down into three activities:

1. Loading at A. The time needed for loading is estimated as ω_A;
2. Passing by B. The passage time needed is estimated as ω_B.
3. Unloading at C. The time needed for unloading is estimated as ω_C;

Of our interest are not the actual operations but the *processing-times* spent on the them, namely, ω_A, ω_B, ω_C, which are associated with A, B, and C respectively: The task of the AGVSP is then to select a sequence of times:

$$(t^{\alpha}_A, t^{\alpha}_B, t^{\alpha}_C) \tag{1.2}$$

with t^{α}_A, t^{α}_B and t^{α}_C as arrival times at which a is planned to arrive A, B, and C respectively: The timed or scheduled activities above are illustrated in Fig 1.2: Let us call timed or scheduled activities for each single AGV *an individual schedule* or *a local*

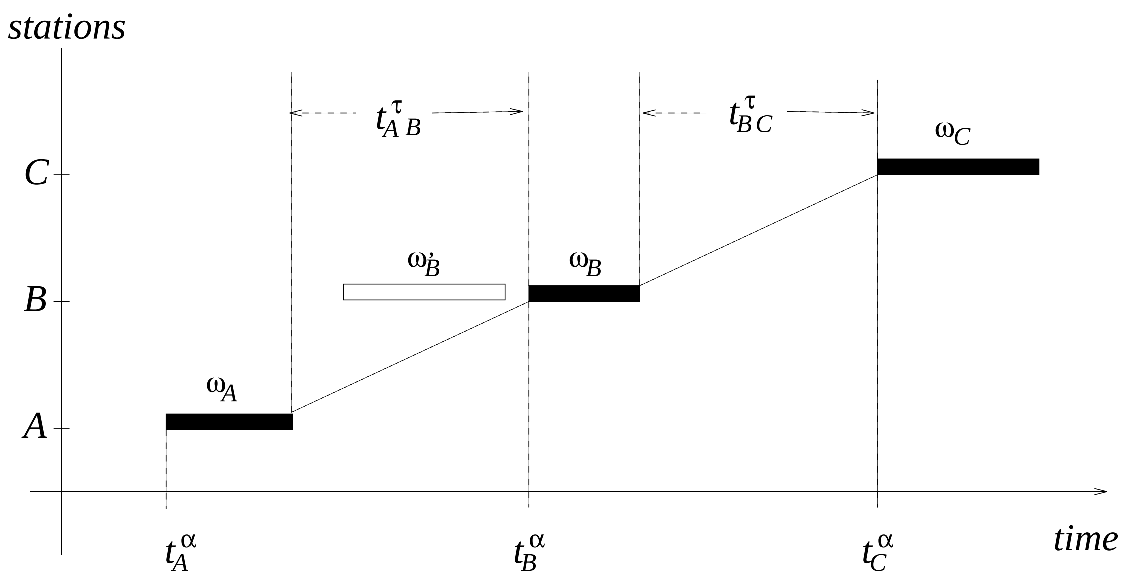


Figure 1.2: An Example of the AGVSP

schedule. In the Figure, t^{τ}_{AB} and t^{τ}_{BC} are traveling times for a to travel from A to B and from B to C respectively. Traveling times depend on an AGV's speed at which it moves from a station to another. Apart from the speed limit, each AGV's activities are also constrained by other requirements such as the earliest start-time of the task, the latest finish-time of the task. All these requirements are set for each single AGV, so we call them intra-agent constraints. These constraints are not relaxable by the problem solver of the AGVSP, so they are often called *hard constraints*.

Another type of requirement for a single AGV are the *soft constraints* which are the preferences by the user over timings of the activities, such as the preferred finish-time of the task, etc. Soft constraints are relaxable, so they need not be strictly satisfied. However, a good schedule is expected to satisfy them as well as is possible.

So far, we only considered the AGVSP with one AGV a. Now let consider the AGVSP problem with two AGVs, a and a'. Suppose a has the same specification as above, and a' has the following specification:

$$A', \quad \omega_{A'};$$
$$B, \quad \omega_B;$$

$$C', \quad \omega_{C'}$$

That is to say, a and a' have different routes but both of them have to travel cross junction B. a' is scheduled as follow:

$$\begin{array}{c} A', \quad \omega_{A'}, \quad t^{\alpha}_{A'}; \\ B, \quad \omega^{'}_{B}, \quad t^{\alpha'}_{B}; \\ C', \quad \omega_{C'}, \quad t^{\alpha}_{C'} \end{array}$$

The basic requirement is that one of the AGVs should cross the junction after the other. In other words, the arrival time $t^{\alpha'}_{B}$ at which a' is planned to arrive at B should be such that its processing-time (passage time) $\omega^{'}_{B}$ does not overlap with the processing-time of a, ω_B (see Fig 1.2). This requirement is set for safety. Since this requirement involves two AGVs, so we call it *an inter-agent constraint* which is a hard constraint. There are other inter-agent constraints, such as no-overtaking: one AGV moving along a route should not overtake another AGV which is moving along the same route. Some requirements may depend on the layout and the AGV guidance system: requirements such as no-overtaking etc. are essential in an AGVS with tracked guidance techniques (e.g., [Brussel *et al.* 88] [Elbrancht & Plum 83]) but may not be necessary with free-range guidance techniques (e.g., [Bohlander 87]).

The AGVSP can be easily extended to an arbitrary number N of AGVs. The AGVSP is to plan each AGV an individual schedule such that for any single AGV, (1) all intra-agent hard constraints are strictly satisfied; (2) all intra-agent soft constraints are best satisfied; and for any two arbitrary AGVs (3) the inter-agent hard constraints are strictly satisfied, and (4) the inter-agent soft constraints, if there are any, are satisfied as well as possible.

1.1.3 Features of the AGVSP

Taking stations and junctions as machines, we can see that the AGVSP is related to the JSSP which involves allocating start-times to activities (or operations). A JSSP can be described by four aspects: (1) m machines, (2) N *jobs* with each of them being a set of operations with each of them being specified by a machine where the operation is processed and a processing-time needed for the operation being processed; (3)

disciplines (or constraints) that restrict the manner in which allocations can be made and (4) the criteria by which a schedule will be evaluated. Many JSSPs are regarded as *constraint satisfaction problems* (CSPs) which are generally described by a set of variables with a set of constraints which describe the relations among variables. As we will see in Chapter 3, the AGVSP is a CSP problem as well. So, many existing approaches to the JSSP and the CSP may be in the candidates to be chosen as approaches to the AGVSP.

Efforts have been made by researchers in the last several decades to seek efficient methods of solving the JSSP and CSP and they are continuing. No general methods have been found that adapt to all kinds of the JSSP or CSP. Every existing approach is based on some assumptions and is appropriate for a certain kind of the JSSP or CSP. As for the case of the AGVSP, it has several requirements:

- *Strictly ordered activities* — for each AGV, its planned route prior to the AGVSP determines the order in which the AGV moves from one station (or junction) to another. So, the order of activities are fixed and cannot be changed during problem solving. This is not the case in many JSSPs where the order of activities may be changed;

- *Strong links* between activities — for each AGV, two consecutive activities are linked by the traveling time between a station for the first activity and another for the second. The constraints on the traveling time are critical: the shortest traveling time is limited by the AGV's speed, and longer traveling times will affect the feasibility of the overall system. By feasibility we mean whether a planning system can organise all the movements of the AGVs required by a factory. Obviously, the longer the traveling times, the fewer movements can be organised.

- *Strong preferences* over solutions: the parts should be delivered to each work station at the time when workers (or robots) need them, arriving earlier may cause a local inventory problem, while arriving later will make less efficient use of resources — workers, machines, robots, etc; ready products should be delivered to customers in time to satisfy them. In other words, JIT (—*just in time*) is required. Classical approaches to the JSSP are aimed at shortening the makespan

— the time period between the time when the first job is started and the time when the last job is finished. The makespan in the case of JIT as in the AGVSP is not as strongly desired as would be in the case of other JSSPs. For example, if a task for AGV a is to deliver some parts through route (ABC) to C and the best time for the worker at C to receive the parts is at 3:00 in the afternoon, then there is no point in making effort to finish the task in the morning.

- *Dynamic scheduling* — the need for dynamic scheduling arises in the AGVSP for two reasons: (1) re-scheduling — since activities for each AGV are strongly linked as explained above and activities between AGVs are constrained for the sake of safety, failure of one AGV's execution with respect to its planned timetable will need the timetable to be re-scheduled, and this may cause a chain reaction —- the timetables for others AGVs may be required to be re-scheduled consequently; (2) possibility of withdrawal of tasks from, and addition of tasks to, the existing task set — withdrawal of tasks means some resources which are allocated to the withdrawn tasks previously are now available for other purposes. So re-scheduling for the remaining AGVs is unavoidable in dynamic movement planning.

These distinguishing requirements have their advantages and disadvantages. One of the disadvantages is that many existing approaches to the JSSP or the CSP are not applicable since they are not designed to adapt to those requirements. The advantages include: (1) critical constraints narrow the solution space to the AGVSP, hence narrowing the search space — this may lead to an efficient, constraint-directed searching approach to the problem; (2) strong preferences may be taken as heuristics in searching for the optimal solution to the problem and lead to the development of preference-directed approaches.

1.2 Kwa's Work on the AGVSP

1.2.1 Brief Introduction to Kwa's Work on the AGVSP

Movement planning involves generating a set of AGV movement schedules which define the paths and the movement timings of the AGVs. On the movement timing, i.e., the AGVSP, Kwa argues that merely generating a logically correct order of actions to achieve a goal only solves the real problem in part [Kwa 88a] [Kwa 88b]. If it over-optimistically assumes that the states of the world will change as expected according to its model of the world, then there is a likelihood that the plan will fail. In other words, besides logical correctness, plans must also have a degree of executability in real world, i.e., the ability to cope with real world issues during execution. A plan with poor executability demands a dynamic replanning capability. However, dynamic replanning can be expensive or is liable to be futile if it fails to complete in time. Hence it is desirable to minimise or defer dynamic replanning as much as possible. As an approach to avoiding frequent dynamic replanning, Kwa proposes applying tolerant planning to AGVSP, i.e., allocating some redundant resources to each agent's (or AGV's) plan to allow leeway for execution errors. In this case, as long as an agent's execution deviation does not exceed the allowed by the redundant resources allocated, then replanning can be delayed.

While this approach is feasible, it raises another problem — more resource conflicts must be resolved during planning. Kwa then proposes a novel model of iterative negotiation for multi-agent coordination in resolving resource conflicts between agents (AGVs). The basic idea of his proposal is: (1) the AGVSP is solved in a multi-agent environment where each AGV is taken as an agent which is responsible for constructing its own local schedule; (2) initially, each agent will allocate some redundant resource (Kwa calls this *interval-based planning*) to its local schedule; (3) once local scheduling is finished, the respective agent will check whether its resource allocation is in conflict with that of other agents; and (4) if it is in conflict with other agents' allocation, then the agent will initialise a negotiation procedure in which the conflict agents exchange information and make concessions until the conflict is resolved. The concession techniques can be either shifting its original resources without yielding any amount of resources, or yielding some amount of resources, or both shifting and yielding.

Kwa examined the characteristics of negotiation, they are: conflict knowledge, willingness to negotiate, knowing the negotiables, selfishness of agents, negotiation being an iterative process, etc. Kwa also outlined the procedure of the iterative negotiation model and illustrated how agents with resource conflicts can resolve them by the negotiation procedure.

More detailed introduction to Kwa's work on the AGVSP is given in Chapter 5.

1.2.2 The Problems in Kwa's Iterative Negotiation Model

Although Kwa's iterative negotiation model can be understood easily, and is correct qualitatively, it lacks a quantitative model of how each agent should negotiate with other involved agents. More specifically, it does not tell each agent *quantitatively*: (1) when it should concede, and (2) how much it should concede (i.e., how much it should shift and how much it should yield). The main question is whether the iterative negotiation model can solve real problems.

In negotiation, agents are selfish but they must also be cooperative to keep the negotiation going. Selfishness of an agent means it will try to acquire as many resources for its own plan as possible, while cooperativeness of the agent means it should concede some of its originally allocated resources to other agents if without any good reason to keep them. What is the compromise between selfishness and cooperativeness? Kwa did not provide any satisfactory answer to this.

1.3 Metaphor-based Negotiation: a Form of Cooperative Interaction in Multi-agent Systems

Although negotiation is a novel approach for solving AGV movement planning, it is not new in Artificial Intelligence. Since Davis & Smith's work [Davis and Smith, 1983], negotiation has been used in many ways in multi-agent systems. There are many ways of interpreting the term, here are just a few of them:

Negotiation is a discussion in which the interested parties exchange information and come to an agreement [Davis and Smith 1983]

Negotiation is the communication established between two conflicting agents in which they try to develop or refine their plans jointly so that the goals of each are satisfied. [Adler 1989].

Negotiation is composed of two phases: a communication phase where information relevant to the negotiation is communicated to participating agents, and a bargaining phase where "deals" are made between individuals or in a group. The Negotiation process can be viewed as constraint-directed search in a problem space [Sathi & Fox 89].

Although there are many different interpretations of the negotiation term in DAI, negotiation is a metaphoric term for human negotiation. So, we call it *metaphor-based negotiation*. It is well understood that it is difficult to include all characteristics of human negotiation in a DAI negotiation model, however, a negotiation metaphor should at least include the following aspects:

- A multiagent system with more than one agent;
- Each agent is assigned a goal (a single goal or a compound goal which may have several separate sub-goals) and is required to achieve it against the objective as well as possible;
- One agent's goal achievement may interfere with that of others, i.e., there are conflicts among agents in goal achievements;
- Agents are required to resolve those conflicts through communication in which agents exchange information to reach an agreement that is acceptable to all the agents involved.

1.3.1 Deficiencies of Metaphor-based Negotiation in DAI

Suppose there are two agents A and B which are required to solve a problem which has three possible solutions: X, Y and Z. Each agent has its own preference over these solutions and their preferences are different from one another. Which is the best solution?

From a purely individual perspective, each agent wants its most preferred solution. Suppose in A's perspective, X is preferred to Y to Z. If B's preference is the same as A's, then it is not difficult for them to agree on solution X. However, if B's preference is reversed with A, then they will be in conflict over which solution to choose. Most negotiation models in DAI try to avoid this problem which, as far as the author is concerned, is a key problem in human negotiation that each negotiator faces. For example, in [Davis & Smith 83], even if a bidder might have a preference over two tasks X and Y, it will contract with the manager to undertake a task the manager wishes it to undertake. That is to say, the bidder's preference is completely ignored. This occurs very rarely in human negotiation: every negotiator's preference should be respected as far as possible.

1.4 Toward Formal Metaphor-based Negotiation

An agent in a multi-agent system is an automatic entity or computational model with knowledge provided by the system designer and the user of the system. This knowledge includes its own local problem-solving (local objective, solution space, searching techniques, etc.) methods as well as knowledge about the environment (about other agents' problem-solving knowledge, about communication mechanism e.g., speech act, about compromising in the case of conflict, etc). Agents are always assumed to be rational — they always aim to maximise their local objectives. From a system designer's view point, allowing each agent to achieving its individual goal maximally will in general assist in achieving the global goals maximally. However, agents in a multi-agent system are not independent. One agent's local goal achievement may interfere with that of others. When this happens, agents must seek a compromise for resolving the conflict. If only assuming rationality, then each agent will possibly stick to a solution that is the most preferred to itself but that is not globally consistent. So, in this case, an agent must be given knowledge about the global solution space and global preferences over global solutions.

In Chapters 2, 3 and 4, I shall discuss this problem using formal mathematical approaches. This discussion will finally lead to a formal definition of metaphor-based negotiation. The basic points of the definition can be expressed in words as follows:

Metaphor-based negotiation is a process of communication in which agents exchange proposals of conflict resolution through common language until a proposal is found that is acceptable to all in term of common sense.

Common language guarantees that a proposal introduced by one agent can be understood by other agents. For example, if an American salesman and a British customer are negotiating the price over a product, the salesman would prefer talking about the price in term of USA dollars, while the buyer might prefer talking about the price in term of Sterling pounds. In this case, they would probably agree to use USA dollars (or Sterling Pounds) as the price unit in their negotiation.

Common sense guarantees that there is conflict resolution when dispute occurs. Common sense could be any knowledge that is known to both agents involved in a negotiation and that could help to solve the dispute. For example, suppose the salesman offers a price of $100 for the product, and the customer offers $80, then the salesman would finally probably agree on $80 if the customer tells the salesman that he can buy the same product at a price of $80 somewhere else. Here, common sense is that if the sales price is over $80, buyers would leave. Common sense is represented in our definition by what we called *a common sense principle*. The common sense principle is introduced to guide conflicting agents to resolve conflicts between them when agents can not resolve them according to the rational agent assumption. The common sense principle also takes part in a role as a law in a society: each individual agent has a certain degree of freedom in its own problem solving, but, when disputes between agents occur, agents must compromise if the principle implies so. In a society (e.g., a nation), when individuals have disputes which can not be resolved by themselves or arbitrated by any third party, then these conflicting individuals may negotiate and compromise under the guidance of the law.

This definition specifically covers what a proposal is; what is a common language is; and what the conditions for accepting a proposal are. This definition is domain independent.

1.5 Spring Model for Local Scheduling in AGVSP

The formal definition of metaphor-based negotiation provides us with a view of what kind of knowledge an agent should have and what is an agreement. However, for a specific problem-solving scenario, an agent's knowledge must be a computational model. In the AGVSP, each agent must have a computational model for its local schedule solution space. The model must also express the preference of the agent over the solution space such that the agent will always choose the most preferred solution in a given circumstance. In this thesis, I shall propose a spring model for local scheduling. The basic idea is described in the following several subsections.

1.5.1 Solving the AGVSP in a Multi-Agent System

Another two properties of the AGVSP, which should be considered in addition to the features of the AGVSP as described in Section 1.1.3 are: (1) the AGVSP can be easily partitioned into a set of subproblems, each of which is to find an individual schedule for one AGV; (2) all subproblems are homogeneous, so an approach which is considered to be suitable for one subproblem will be suitable for any others. For these reasons, we believe that it is very natural to solve the AGVSP in a multi-agent system where each AGV is taken as an intelligent agent with knowledge of solving its own problem — scheduling its own activities. All agents are homogeneous, therefore the efforts on seeking the methods of solving each subproblem are reduced.

1.5.2 Incremental Scheduling

To meet the requirements of dynamic movement planning, the initial consideration could be incremental scheduling by which the overall scheduling is built up by scheduling one AGV after another, i.e., local AGV scheduling problems are solved in a particular order. In this way, dynamic scheduling is possible. For example, when a new task is added to the existing task, it can be simply considered as another local AGV scheduling problem.

1.5.3 Negotiation as Resolution for Resource Conflicts

The AGVSP, like many other JSSPs, is resource-based. In a multi-agent system, if the intra-agent constraints can be not satisfied, then they will cause intra-agent conflicts. Similarly, if the inter-agent constraints cannot be satisfied, then they will cause inter-agent conflicts. These conflicts are displayed as resource conflicts between two activities for one AGV, or among activities for several AGVs. Conflicts in a multi-agent system are often resolved by metaphor-based negotiation. We think there is much potential in applying negotiation as a mechanism for resolving resource conflicts in solving the AGVSP using a multi-agent system: the hard constraints are the reservation line for negotiation, and the soft constraints express the negotiability. More preferred requirements are less likely to be negotiated away than less preferred requirements. For example, a scheduled its activity at B before a' as shown by ω_B with arrival time t_B^α in Fig 1.3. Then a' scheduled its activity at B as shown by $\omega_B^{'}$ with arrival time $t_B^{\alpha'}$ for a'. Now, a' found that the times for the two activities overlap one another. The conflict can be resolved by only one of them shifting its arrival time until the overlap disappears, or — in the general case — by both shifting in different directions.

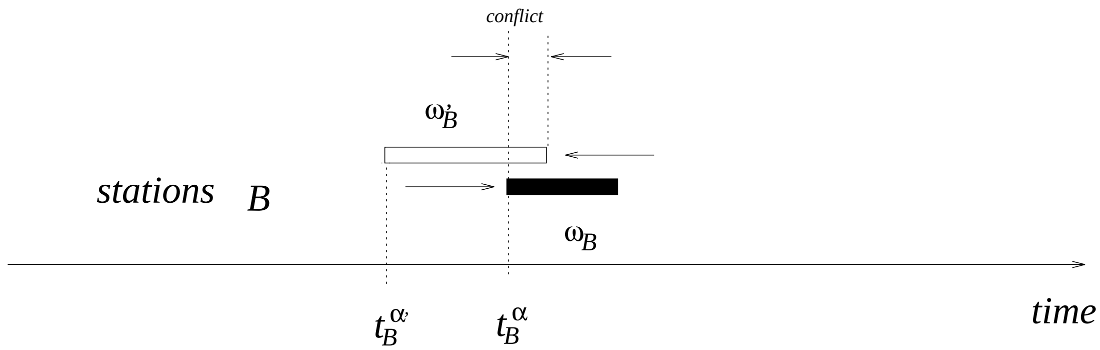


Figure 1.3: An Example of Resolving Resource Conflicts via Negotiation (a)

1.5.4 Using a Spring to Represent Soft Constraints

Soft constraints are negotiable (or relaxable), though each agent will try to relax its soft constraints as little as possible. There is also a tendency that the agent, once it relaxes some of soft constraints, will return to its initial state (i.e., the best solution in its local viewpoint). These features can be described by a spring: originally, without any external force, the spring will be in a state in which it preserves the lowest energy; when compressed or stretched by an external force, then the spring will be changed in form; when the external force disappears the spring will return to its original form. In any static situation, a spring will be in a static state (the optimal state) where it preserves the lowest possible energy in that situation.

A soft constraint with a lower degree of relaxability can be represented by a spring with a larger spring constant, while a soft constraint with higher degree of relaxability can be represented by a spring with a smaller spring constant.

1.5.5 A Simple Example of Applying Spring Model to Resolve Resource Conflict

The resource conflict shown in Figure 1.3 can be resolved by applying the spring models illustrated in Figure 1.4 and Figure 1.5. Each of the two conflicting intervals ω_B' and ω_B is connected to a spring which constrains its shifting (see Figure 1.4). ω_B will shift right and ω_B' will shift left. The shifting strategy will be applied continuously until the conflict (or the overlap) is completely resolved (see Figure 1.5). How much each interval has shifted will be discussed in Section 1.5.6.

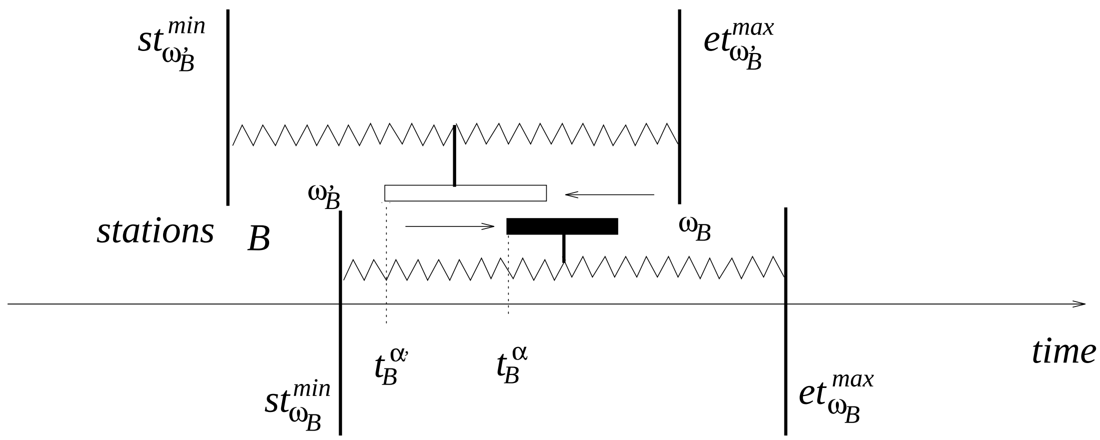


Figure 1.4: An Example of Resolving Resource Conflicts via Negotiation

1.5.6 Common Sense in the AGVSP Negotiation Model

The spring constant for a spring that represents a soft constraint expresses the degree of negotiability in a negotiation. If an agreement is achieved after the negotiation, then some of both agents soft constraints are relaxed (i.e., for both agents some springs are compressed). We can view each local schedule as a local spring network, and a global schedule is a global spring network properly connected by local schedules. What is the optimal state of the global spring network? The optimal state is one in which the total energy preserved by the spring network is minimal. This is the common sense that should be known to each agent to guide it in negotiation. In this state, the forces of $f_{\omega'_B}$ which is the reacting force of spring ω'_B being compressed by ω_B should be equal to f_{ω_B} which is the reacting force of ω_B being compressed by ω'_B.

1.6 Contributions of the Dissertation

As an overview, this section will present the reader with an outline of the major contribution of this dissertation.

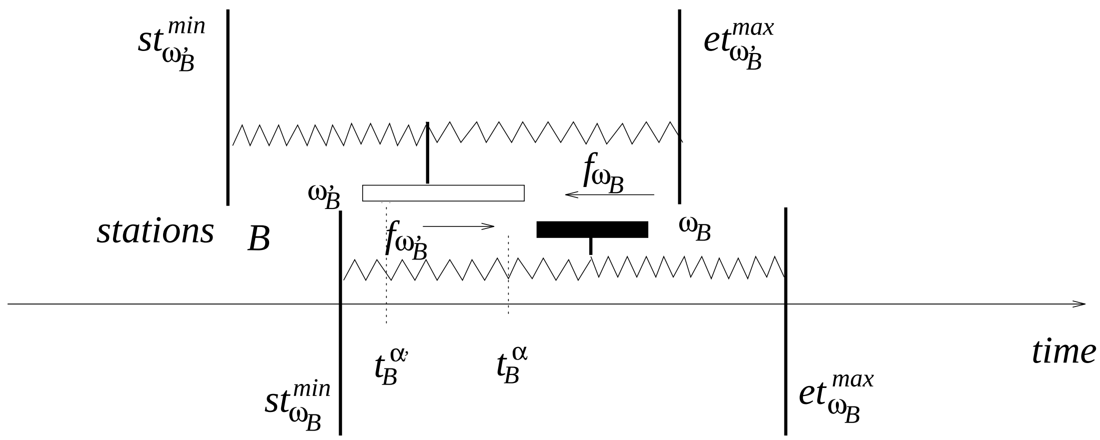


Figure 1.5: An Example of Resolving Resource Conflicts via negotiation

1.6.1 Cooperative Interaction among Fully Informed Rational Agents

When an agent in a multi-agent system is said to be fully informed (or having complete knowledge), it means that the agent has knowledge about itself (its own problem-solving — the solution space, the preference over solution space, the searching techniques, etc.) as well as knowledge about other agents (other agents' knowledge about their problem-solving). When an agent is said to be rational, it means that the agent will always try to maximise its own goal-achievement (or to maximise it objective function). When agents' goal-achievements are in conflict, cooperative interaction can be introduced to resolve conflicts.

As the reader will see from the literature review in cooperative interaction in Chapter 2, cooperative interaction has been a phrase widely used in DAI. There are two approaches to understanding multiagent cooperative interactions in general: formal mathematical approaches that attempt to develop theories of interaction and cooperation and system-building approaches that attempt to build systems that support real or simulated agent interaction. Unfortunately neither approach found so far is convincing.

Based on assumptions of rationality and complete knowledge, I shall construct a cooperative interaction environment from which I shall discuss in detail to what extent agents with conflicting individual goals can cooperate with each other. By assuming also that agents are benevolent — an agent will cooperate with other agents if that cooperation does not affect its own goal achievement (See Chapter 3 for formal definition), I shall prove that for any initial cooperative situation, agents' cooperative interaction will run into a difficult cooperative situation where one agent's move which is aimed at improving its own goal achievement will definitely impair at least one of the other agents' goal achievements. So, merely assuming agents are benevolent, rational, and fully informed cannot guarantee successful cooperative interaction.

1.6.2 Common Sense in Cooperative Interaction

When multiagent cooperative interactions run into a difficult situation where one agent's move which is aimed at improving its own goal achievement will definitely impair at least one of the other agent's goal achievements, the cooperation seems to get into a impasse. I shall introduce the use of common sense to break the impasse. The basic idea is that when no agent in a cooperative interaction is willing to compromise any more, all agents will agree on a solution that may not be satisfactory to all agents but that is not against a particular rule which I call the "common sense principle." For example, in a football game, the two teams often decide who kick off first by tossing a coin. Here the common sense principle is that both teams should abide by the result of the toss (which is a result of nature and fair to our common sense).

1.6.3 Negotiation: A Form of Cooperative Interaction among Agents with incomplete knowledge

In a situation where agents do not have complete knowledge, cooperative interaction will be even more difficult. Communication can bridge agents to exchange information. When conflicts occur, those agents can exchange proposals until a proposal is found that is acceptable to all agents. I shall propose a formal definition of metaphor-based negotiation which formally defines what is a proposal, and what is the condition under which an agent has to accept a proposal, etc.

1.6.4 Spring Model for the AGVSP

The Spring Model is used for two purposes:

- It is a model of local scheduling. An agent's intra-agent soft constraints are represented by springs which are connected properly as a local spring network.
- It is a model of guiding an AGV in negotiation about what negotiation strategies to apply quantitatively. When an agent's local schedule is in conflict with that of another agent, it will force that agent to concede just as it will push a local spring network that represents that agent's local schedule. The other agent will act in a similar way. How much will it concede? It will depend on their reacting force: the one with smaller reacting force should concede more. The negotiation will finish when their reacting forces are equal.

1.6.5 Repairing Schedules

During execution, one AGV has to change its original schedule for some reason. As a consequence, some other AGV's schedule may also need repairing. The spring model serves this purpose very well. For example, when an agent withdraws its schedule, then it will leave some resource space unused. In this case, other agents may intend to use that resource space by relaxing their springs that were compressed before.
In general, repairing schedules is just another round of negotiation.

1.7 Guide to the Reader

The dissertation can be viewed to have two closely related parts. The first part, which includes Chapter 2, Chapter3 and Chapter 4, investigates the general DAI problem, i.e., how rational agents can cooperate? The investigation leads to a formal definition of metaphor-based negotiation. The second part, which includes Chapter 5, Chapter 6, and Chapter 7, is to solve the AGVSP problem by a novel spring-based negotiation model which is based on the formal definition given in the first part and the spring-based representational model we developed. The outlines of each chapter are as follows:

- Chapter 2 presents a literature review in cooperative interaction in DAI.
- Chapter 3 discusses how agents that are assumed to be rational, benevolent and fully-informed can cooperate each other when they are situated in a cooperative interaction problem (CIP) scenario.
- Chapter 3 continues Chapter 3's discussion by assuming that agents are partially informed. A metaphor-based negotiation definition is proposed.
- Chapter 5 introduces Kwa's work related to the AGVSP problem, especially his iterative negotiation model for conflict resolution.
- Chapter 6 formally specifies the AGVSP domain and presents our representational spring-based model for AGV local scheduling and for guiding AGV to work out negotiating proposals.
- Chapter 7 outlines our implementation of the spring-based negotiation model and presents some experimental results.
- Chapter 8 concludes the dissertation by presenting our major contributions in this dissertation and some possible future work relating to this dissertation.

Chapter 2

Literature Review in Cooperative Interaction

2.1 Introduction

Distributed Artificial (DAI) is the subfield of AI concerned with studying a broad range of issues related to the distribution and coordination of knowledge and actions in environments involving multiple entities (or agents) [Bond & Gasser 88]. These agents are grouped into communities which work together to achieve the goals of the individuals and of the system as a whole [Jennings 94]. The world of DAI can be divided into three areas: Distributed Problem Solving (DPS), Multi-agent System (MAS) and Parallel AI (PAI). Research in DPS considers how the work of solving a particular problem can be divided among a number of modules, or "nodes" that cooperate at the level of dividing and sharing knowledge about the problem and about the developing solution. Research in MAS is concerned with coordinating intelligent behaviour among a collection of autonomous agents so that they can coordinate their knowledge, goals, and plans jointly to take action or to solve problems. Research in PAI is concerned with developing parallel computer architectures, languages, and algorithms for AI. The work in this dissertation can be regarded as in the area of MAS. Cooperation and negotiation is one of the major research areas of MAS and also is the theme of this dissertation. In the rest of the chapter, I shall survey the related research in cooperation and negotiation.

2.2 Formal Models of Cooperation and Negotiation

Formal models of cooperation and negotiation are mathematical models [Rosenschein & Genesereth 88] [Genesereth *et al.* 86] [Ephrati & Rosenschein 91] [Zlotkin & Rosensein 89] [Zlotkin & Rosensein 90] [Zlotkin & Rosensein 91]. These models are aimed at solving the following problem: given a scenario, what are the best strategies for agents to take in cooperative interactions, or what are the best compromises (or deals)?

Most formal models assume that agents in a multiagent system are rational, i.e., they are aiming at maximising their own local utilities. Negotiation is viewed as a separate process used to select a solution from a set of candidate solutions [Lander 94]. Each agent involved in the negotiation attempts to select the candidate that will maximise local utility with the understanding that all agents must ultimately agree on a single candidate. The negotiation process does not itself define the set of candidate solutions. It is usually assumed that only two agents are involved in the negotiation, that agents are rational (rationality), meaning that they will not select an action that will result in an avoidably poor payoff, and that each agent knows the other agent's potential payoffs for all candidate solutions (full-informedness or complete knowledge). Conflicts exist in general between these fully informed and rational agents since they have different preferences over possible solutions.

Deals among rational agents [Rosenschein & Genesereth 88] [Durfee & Lesser 87] [Ginsberg 87] can be made by assuming agents are benevolent. By benevolent is meant that all agents are fundamentally assumed to be helping one another, and will trade data and hypotheses as well as carry out tasks that are requested of them. Rosenschein and Genesereth made various assumptions about rationality (e.g., minimal move rationality, separate move rationality, etc.), and derived several strategies of cooperative deals. The role of communication is emphasised in this formal model: communication makes possible mutually beneficial activity that is otherwise impossible to coordinate. The work of Rosenschein and Genesereth in [Rosenschein & Genesereth 88], together with some other early formal models of cooperative interaction such as [Rosenschein 85], became the earliest work in AI that formalises multiagent cooperative interaction through a game-theoretic approach. However, these models suffer from some inadequacies: (1) incompleteness — models cannot guarantee that a compromise can always be made regardless of what the interactive situation is; (2) some assumptions (e.g., fully informed agent assumption, benevolent agent assumption (i.e., agents are willing to sacrifice their own goal achievements to help other agents, etc) are too strict to be practical.

Deals among rational agents can also be implicitly made even without communication [Genesereth *et al.* 86] [Kraus & Rosenschein 92] [Fenster *et al.* 95]. For example, agents can apply a coordination technique common to communication-free interactions, namely focal points, to resolve conflicts when there is more than one possible cooperative strategy. Focal points are the dominant properties of objects, plans, actions, etc. When several agents are in a situation where one object must be jointly chosen from several possible objects and where communication is not possible between those agents, they have a tendency of choosing unanimously the same object (hence no conflict happens) that has some dominant properties. Focal points are an important concept in psychological and economics study [Schelling 60]. In adapting the idea of focal points from human behaviour patterns to automated agents, as stated in [Kraus & Rosenschein 92], one major difference must be considered. Focal points are based on the naturalness and intuitiveness of certain objects (or solutions) in the world. Automated agents do not have the cultural background (common sense) needed to judge naturalness and intuitiveness. In other words, the formal algorithms developed in [Kraus & Rosenschein 92] depend very largely on what focal points the designer will endow them with. Finding focal points could be difficult, as far as the author is concerned. However, one important point made in [Kraus & Rosenschein 92] which is particularly of our interest is that when agents run into a difficult situation of cooperative interaction, agents must be endowed with some additional knowledge, such as knowledge of focal points, so that their cooperation process can be continued. I shall call this knowledge, the common sense principle, which is an important concept in the definition of metaphor-based negotiation given in Chapter 4.

Assumption-based cooperation is the main stream of the formal cooperation model introduced so far, i.e., besides the basic assumptions that agents are rational and fully informed, one agent must reason about cooperative strategies strongly based on assumptions on what assumptions the other agents are based on. For example, in focal points algorithms, one agent will assume that the other agents will assume the same focal points. Otherwise, it might be not possible for them to choose the same object. *Equilibrium-based negotiation* is another major stream of cooperative interaction and negotiation [Kraus & Wilkenfeld 91b] [Kraus & Wilkenfeld 91a] [Kraus *et al.* 95]. Agents' cooperation is based on a so-called negotiation process in which they jointly reason about the best compromises by proposing offers or counter-offers. The process usually stops at a time point when none of those involved agents has the motivation of moving away from a compromise proposal that is often called an equilibrium in game theory [vonNeumann & Morgenstern 44]. In this situation, agents are said to be cooperative since it is generally assumed (1) that agents agree to make a joint plan before an action to be taken rather than they make decisions independently, and (2) agents will be bound by any decision that is jointly made. However, regarding what is the compromise, this kind of negotiation is sometimes called *non-cooperative* interaction, since compromises are made in a competitive manner: in each step of the negotiation, an agent, as long as it see there is still something achievable, will propose a new plan to achieve the achievable, regardless of what (negative) effect it has on the other agents. For example, in [Kraus *et al.* 95], work has been concerned with how automated agents can be designed to interact effectively in both resource allocation and distribution environments. A strategic model of negotiation has been proposed as a way of reaching mutual benefit while avoiding costly and time consuming interactions which might increase the overhead of coordination. The work is based on the game-theoretic model of Rubinstein [Rubinstein 82]. This formal model does not assume that agents have common sense as in [Kraus & Rosenschein 92], i.e., the negotiation can converge to an agreement without exploiting any common sense. In this sense, the model developed in [Kraus *et al.* 95] does have some advantages in some situations where common sense is not attainable. However, according to my study, this model suffers at least such an inadequacy: even when all negotiators' initial states are the same, the negotiation cannot guarantee a negotiation result that is symmetric. In other words, the fairness of compromise using this formal model is questionable.

2.3 Computational Model of Cooperation and Negotiation

2.3.1 Contract Net

Davis and Smith [Davis & Smith 83] [Smith & Davis 88] could be considered as pioneers for introducing the negotiation concept to the AI research field. The use of negotiation for the allocation of tasks in their contract net protocol was studied. The process of negotiation was considered as one (the manager) announcing the task and the others (the bidders) bidding, and the bidder who bids at the lowest price will be awarded the task and hence become the contractor. The agreement is therefore reached. In this approach, the agents are hierarchically related, one agent (manager) will be in relation to several other agents (the bidders). The bidder who becomes the contractor may decompose the task and announce them to the public, other agents start to bid. So this is a totally hierarchical system: agents are hierarchically related, tasks are hierarchically decomposed, and conflict resolution is hierarchical. The work is extremely elementary. However the contract net protocol is a single-shot negotiation process, i.e., announce - bidding - decision, this is very rare in human negotiation in which there are many rounds of the information exchange.

A similar work to [Davis & Smith 83] is presented by [Sandholm 93] in which the bidding and awarding decision process that was left undefined in the original contract net task allocation protocol is formalised based on marginal cost calculations based on local agent criteria. In this way, agents having very different local criteria (based on their self-interest) can interact to distribute tasks so that the network as a whole functions more effectively. The application in the transportation domain is discussed. Work related to Contract Net can also be found in [Conry *et al.* 91] where the Contract Net approach is used to resolve conflict in *distributed constraint satisfaction problems*.

As indicated in [Davis & Smith 83], the Contract Net Negotiation cannot solve the following typical problem (which is also unsolved by the other computational models that are introduced below):

Consider for example a situation in which two managers (A and B) have both announced tasks, and two potential contractors (X and Y) have each responded by bidding on both tasks. Imagine further that from A's perspective, X's bid is rated 0.9 (on a 0 to 1 scale), while Y's is rated 0.8. Conversely, from B's perspective, X is rated 0.8 and Y is rated 0.2. Which bidder should be assigned task from A, and which bidder should be assigned task from B? From a purely local perspective, both of the managers want X as their contractor; form a more global perspective it may make more sense to have A "settle" for Y, and give X to B. However, this question has not been discussed in depth [Davis & Smith 83]. That is to say, the preferences of managers are totally ignored. I shall consider this is also not the case generally in a human negotiation. In our formal definition of metaphor-based negotiation, this issue is carefully addressed.

2.4 Constraint-directed Negotiation

Constraint Satisfaction Problems (CSPs) can be solved in DAI approaches. For example, in a related class of problems in which satisfaction of each goal presented to the network requires a coordinated set of actions distributed over a subset of the nodes for completion [Conry *et al.* 91], each node has limited resource available for satisfaction of global goals, and once a resource has been committed in satisfaction of one goal, it cannot be used for another. Planning relative to satisfaction of the set of global goals is nontrivial. The combination of local resource constraints and required coordination of actions among nodes that results gives rise to a complex set of global interdependent constraints. Such a problem is referred to as the *Distributed Constraint Satisfaction Problem* (DCSP) [Conry *et al.* 91] [Sycara *et al.* 90] [Yokoo *et al.* 90] [Sathi & Fox 89].

In [Sathi & Fox 89], Sathi and Fox describe constraint-directed negotiation in the domain of resource reallocation in an engineering organisation. In this domain, negotiation is composed of two phases: a communication phase where information relevant to the negotiation is communicated to participating agents, and a bargaining phase where "deals" are made between individuals or in a group. In resource reallocation, information about available bids has to be communicated minimally, while agents may individually or as a group make tradeoff decisions about how to satisfy requirements. Negotiation is performed among a set of agents. Each agent owns a set of resources employed by the agent to fulfill resource requirements. If there is a difference between the resource requirement and the resources owned by an agent, changes in resource ownership are solicited through buy and sell bids. Constraints can be used both for evaluation of existing alternatives as well as creating new ones. A set of qualitative evaluation and relaxation (alternative generation) techniques based on human negotiation problem solving is defined.

Work in [Sathi & Fox 89] has profoundly addressed several important issues:

- The constraint specifies preference for an alternative in the form of utility, minimally accepted or a threshold for a utility and a constraint importance relative to other constraints. That is to say, agents have preferences over solutions and these preferences are well considered. So, constraints can be relaxed if necessary but the relaxation must be in accordance with their respective preferences. This observation is directly related to the spring model developed in this dissertation in which (soft) constraints are rated according to their importance and springs is used to represent the preferences.

- The combination of these utilities for constraints are made to form a global utility to evaluate the global solutions.

- Constraints are taken as roles of heuristics for searching solutions. Although this issue is not directly relevant to this dissertation, it provides a view that searching techniques for efficiency of problem solving in DAI are as important as in centralised problem solving.

2.4.1 Iterative Negotiation Model for Conflict Resolution in AGV Movement Planning

Kwa [Kwa 88a] [Kwa 88b] developed an iterative negotiation model for conflict resolution in AGV movement planning. Since, to some extent, this dissertation is a continuation of Kwa's work in AGV planning, a reimplementation of Kwa's iterative negotiation model has been made by the author with the negotiation model proposed in this dissertation. The detailed introduction to and the re-implementation of the iterative negotiation model can be found in Chapter 5.

Kwa's model is procedural and straightforward. The most important points that remain unanswered are:

- How to define negotiation skills, and negotiation techniques?
- which agent involved in a resource conflict should concede more than the other quantitatively?

2.5 AI-based Negotiation Support System

2.5.1 PERSUADER — Resolving Goal Conflicts via Negotiation

PERSUADER [Sycara 88] acts as mediator between a company and a trade union. Sycara built a centralized planner, PERSUADER, for resolving conflicts between agents' goals. Given a particular conflict and the context in which the conflict occurs, the planner has two alternatives: one is to find a new compromise by using case-based reasoning or multi-attribute preference analysis strategies, the other is to use persuasive arguments to convince agents to accept the proposed compromise by using explanation-based reasoning or to try to improve the compromise by asking for justification of disagreements. Negotiation is coordinated by a centralised mediator, making it an explicit process that is outside the scope of the agents. The mediator has access to local and global information about the situation and about the negotiation participants. In this approach, the mediator is introduced to act as a persuader to give proposals.

2.5.2 Apply Genetic Algorithm in Negotiation Support System

Stan Matwin is a researcher who made many contributions to AI-based negotiation. Many problems in studying negotiation are pointed out [Matwin *et al.* 89] [Matwin *et al.* 91]: Many two-party multiple-issue negotiations have been described, but only informally; some have been cast in mathematical models. An expert system shell for negotiation support is described, called Negoplan which supports one party's negotiation during a negotiation process (asymmetrical system). The decision is made by production rules and these rules can be generated by using Genetic Algorithms.

2.6 Frameworks of Cooperation and Negotiation

There are several frameworks of cooperation and negotiation developed in recent years. The frameworks are emphasised on system structures of cooperation and negotiation, such as general stages of problem solving in cooperation and negotiation, communication protocols, mental representation of agents, etc.

2.6.1 TEAM

Lander and Lesser [Lander *et al.* 91] [Lander & Lesser 91] [Lander & Lesser 92] [Lander & Lesser 93], describe a framework, TEAM, for multi-agent cooperative design of mechanical systems using negotiated search. This work focused on integrating distributed search among a set of heterogeneous and reusable agents. In TEAM, negotiation is initiated through the detection of a conflict as an agent critiques or extends a proposal initiated by another. The detecting agent notifies the initiating agent of the conflict and communicates relevant information about the cause of the conflict. The initiating agent gathers critiques of its proposal and integrates the shared information into its own knowledge base, making choices about how best to do that. Those choices can include relaxing its own solution requirements to accommodate another agent or refusing to relax a particular requirement even though it is explicitly incompatible with a requirement stated by another agent. in consequence, the final proposed solution may be unacceptable to one or more agents under the complete set of preferences. Each agent participating in the negotiation has the option of accepting or rejecting a proposed solution. If a solution is initially rejected, it remains available and may be reconsidered in the future as agents continue to relax requirements in their search for a mutually-acceptable solution. The TEAM framework provides a flexible control structure that allows agents to share information about their problem-solving capabilities and preferences with respect to a particular problem. TEAM is structured as a blackboard system where overall solutions are integral, and agents build their subsolutions with their own expertises. A very important concept presented in TEAM is *linear compromise* which is claimed as a limited, but very efficient, negotiated-search strategy: a compromise is reached when two agent's linear utility functions over the value of a solution intersect at a point of price, then the point is taken as the "fairest" value, i.e., a compromise.

2.6.2 The Recursive Negotiation Model (RNM)

In [Laasri *et al.* 92], a generic model, called the *Recursive Negotiation Model* (RNM), is presented to serve as a basis for classifying and specifying where conflict resolution among multiple experts, viewpoints, or types of reasoning is needed in building a sophisticated *Cooperative Distributed Problem Solving* (CDPS) system. This model defines where and how negotiation can be applied during problem solving based on structuring problem solving into four stages: *problem formulation, focus-of-attention, allocation of goals or tasks*, and *achievement of goals or tasks.* It also categorises the various styles of negotiation that could occur depending on (1) whether or not negotiation is an integral part of the problem solving, (2) the particular problem-solving stage where negotiation may be used, and (3) the assumptions on which the agents' cooperation is based. It points out that negotiation may be a recursive, complex, and pervasive process that is relevant not only to domain problem solving but also to control (meta-level) problem solving. Some general aspects of negotiation are discussed, such as information exchange, conflict detection, propagation, conflict resolution, and the organisation of negotiation, etc.

2.6.3 The Decentralised Negotiation (DENEGOT) Model

In [Moehlman *et al.* 92], a distributed negotiation framework, the DENEGOT, is described. In the system, the negotiation search space is structured into a lattice of sets of potential compromise solutions based on hard constraints to estimate the quality of potential solutions. A solution in a higher set in the lattice, if it is achievable, will be preferable over a solution in a lower set. Agents search first under the hard constraint level representing the highest quality solution standard achievable in the current situation. By relaxing hard constraints, the set of compromises that qualify as a solution are enlarged. Agents search for a resolution under the relaxed hard constraint set when a solution cannot be found under the current set of constraints. Three iterative phases (coordinated search, negotiation state analysis, and constraint relaxation) of the system are discussed, among them of most interest to us is the second phase during which suboptimal solutions will be found if ones exist. Though not explicitly pointed out in the article, we would take this is a procedure of relaxing soft constraints, partially, or fully. If no solution is achieved, then it enters into the third phase to relax some hard constraints. Each agent is assumed to cooperatively work to optimise the overall goals and if necessary to sacrifice its own goal in order to achieve overall optimal solutions.

2.7 Summary

In this chapter, we reviewed some work in the DAI field concerning multi-agent cooperation and negotiation. Research on multi-agent cooperation and negotiation is classified into formal models and computational models. The formal models focus on what is the best compromise based on some axioms or assumptions while the computational models focus on how a complex application problem can be solved in the DAI manner by several cooperative agents. Our investigation in the first part of the dissertation (Chapter 3 and 4) falls into research on a formal model while our investigation in the second part (Chapter 5, 6 and 7) falls into research on a computational model.

Chapter 3

Cooperative Interaction

This chapter focuses on cooperative interaction among rational agents. Cooperative interaction here means joint agreement between willing entities in a situation where none of the agents have the total control of a problem solving, neither does there exist any third party that could possibly be involved in this matter. Agents are assumed to be fully informed, or say, have complete knowledge about themselves and about others. Each agent is also assumed to be rational. By rational I mean the agent is a utility maximiser. This assumption is obviously widely accepted in the DAI community as well as other research fields, such as game theory. However, our study here shows that even rational agents that are willing to cooperate could run into a conflict situation where one agent's gain means the others' loss. In this difficult situation, compromise is impossible if agents are only assumed to be rational. The idea of a common sense principle is therefore introduced as mediation between these conflicting agents.

3.1 A Scenario of Cooperative Interaction

In order to carry out our investigation, we construct a cooperative interaction scenario, denoted as $< \mathcal{A}, \Sigma, < \pi_1, \pi_2, ..., \pi_i, ... >>$:

1. $\mathcal{A}$ is the agent set: Each member, i, of $\mathcal{A}$ is an agent.
2. Σ is the problem domain, we call it the *cooperative strategy set.* Each member of Σ is called a *cooperative strategy*
3. π_i ($i \in \mathcal{A}$) is a payoff function (or utility function) which maps Σ into the

real number domain $\Re$ indicating the preference of agent i over Σ: for any two strategies, one that yields a higher payoff to agent i is preferable to the other

$$\pi_i : \Sigma \rightarrow \Re$$

4. The task of the CIP is to select a strategy from Σ that can satisfy all agents' preferences as well as possible.

5. No agent in $\mathcal{A}$ has the total control over which strategy to choose.

6. There exists no any third party that can take part in a role as an arbitrator.

3.2 Cooperative Interaction in DAI

In this section, we will illustrate some typical cooperative interaction problems in DAI.

3.2.1 Resource Allocation Problem: Cake-sharing Domain

Resource Allocation Problem

When a set of agents each having its own goal to achieve shares resources that are limited in quantities and that are not reusable or that can only be used by one agent at a time, then conflicts are likely to occur if more than one agent intend to use the same resources at the same time. Agreements acceptable to all are therefore sought so that the resources can be divided among them with good reasons.
Distributed constraint scheduling problems (DCSPs) are examples of the resource allocation problem. In a DCSP, an overall scheduling problem is decomposed into several subproblems. Each of the subproblems is assigned to an agent. According to the assignment, the assigned agent is responsible for solving the subproblem, i.e., to allocate resources to a subset of operations. Resources could be communication bandwidth, machines, routes, etc, depending on what the application domain is. Initially, each agent is always trying to allocate a piece of resource to each operation in its assigned subproblem such that the subproblem is optimally solved. If there is only one agent in the world we are concerned with, then traditional AI approaches can be applied to solve the problem. Now in a multiagent system, when two or more agents' initial demands

overlap, then agents involved should concede from their initial demands. Concession can be made through negotiation by one agent putting forward a concession proposal and the others evaluating it, criticising it, or putting forward their counter-proposals. Each such proposal expresses the proposer's will of how the conflict should be resolved. It is obvious that the more the proposer is willing to concede, the easier the conflict is to resolve. On the other hand, the more the proposer concedes, the worse for its own problem-solving. Each proposer must find a proposal that can balance between resolving the overall conflict and its individual problem-solving. This sometime proves to be very difficult.

The AGVSP is an resource allocation problem: several agents, each of which is assigned a sub-problem of the AGVSP and aims to build up an optimal local schedule to its sub-problem, and compete for limited resources. Conflicts will occur when one agent's resource requirement overlaps that of others. Overlap of resources violates the AGVSP constraints. Agents involved in an overlap can resolve these conflicts through negotiation. The following example illustrates a resource allocation problem.

Example 3.1 *Cake-sharing Domain:*

A piece of cake is to be shared between agent 1 and agent 2. Suppose the cake can be divided into two portions, $x(0 \leq x \leq 1)$, $y(0 \leq x \leq 1)$ *(the whole cake is 1 unit), and* x *is the share for agent 1 and* y *is for agent 2. Agent 1 evaluates its share by a payoff function* $\pi_1(x)$*, and agent 2 evaluates it share by* $\pi_2(y)$ *(payoffs can be considered as the quality of satisfaction — the larger the payoff for an agent the happier the agent is). Also none of them has the total control about how to divide the cake, nor there is any third party involved in this matter.*

Can they make a joint decision on how to share the cake?

We have given a similar cake-sharing example in [Huang 95] where the cake can only be divided into 7 equal portions. Example 3.1 assumes that the cake can be divided in a continuous domain. The cake-sharing domain is a CIP problem $< \mathcal{A}, \Sigma, < \pi_1, \pi_2 >>$:

$$\left.\begin{array}{l} \mathcal{A} = \{1, 2\} \\ \Sigma = \{(x, y) \mid 0 \leq x, y \leq 1\} \\ \pi_1(\sigma) = \left\{ \begin{array}{ll} 0 & \text{if } \ x + y \geq 1 \\ h_1(x) & \text{otherwise} \end{array} \right. \\ \pi_2(\sigma) = \left\{ \begin{array}{ll} 0 & \text{if } \ x + y \geq 1 \\ h_2(y) & \text{otherwise} \end{array} \right. \end{array} \right\} \tag{3.1}$$

Here $h_i(x)$ $(i = 1, 2)$ is a function measuring agent i's satisfaction over an agreement according to which he gets a portion of x of the piece of the cake. We may take $h_i(x)$ as a function returning a value positively proportional to x, i.e., the larger the portion x agent i gets, the more satisfied he is.

3.2.2 Task Allocation Problem

In the resource allocation problem, we are interested in how agents that have been assigned their specific goals (or tasks) overcome resource conflict rather than how these goals are assigned to them. The task distribution problem is concerned with how these goals are assigned to agents in a multiagent system. When all agents have a common goal to achieve, they can divide the overall goal into several subgoals each of which is undertaken by a subgroup of these agents. Since undertaking a task is costly, each agent intends to do as little as possible. So, agents have to negotiate with each other to reach an agreement. An example of task distribution is the postman domain (see Example 3.2). In AGV Planning, it is obvious that task assignment is a task distribution problem if this problem is to be solved by an multiagent approach. In this case, several agents (AGVs) can negotiate over how these tasks should be divided. Each agent is trying to undertake a subset of the tasks that costs it little. Conflicts will occur when a task cannot be allocated, or when several agents compete for one task. Work relating to the task allocation problem can be found in, for example, Davis and Smith's Contract Net ([Davis & Smith 83], [Smith & Davis 88], [Smith 88]) in which several agents (bidders) compete for one task announced by the manager; and the partial Global Planning of Durfee and Lesser ([Durfee & Lesser 89], [Durfee & Lesser 87]) in which agents negotiate over task decomposition and distribution of tasks by sharing all of their information, or by exchanging proposals and counter-proposals.

Example 3.2 Postman Domain *[Zlotkin & Rosenschein 93]:*
Agents have to deliver sets of letters to mailboxes, which are arranged on a weighted graph $G = G(V, E)$*. There is no limit to the number of letters that can fit in a mailbox. After delivering all letters, agents must return to the starting point (the post office). Agents can exchange letters at no cost while they are at the post office, prior to delivery.*
Task Set: The set of all addresses in the graph, namely V*. If address* x *is in an agent's task set, it means that he has at least one letter to deliver to* x*.*
Cost Function: The cost of a subset of addresses $X \subset V$*, i.e.,* $c(X)$*, is the length of the minimal path that starts at the post office, visits all members of* X*, and ends at the post office.*

We can view the postman domain as a CIP problem $< \mathcal{A}, \Sigma, < \pi_1, \pi_2 >>$:

$$\left.\begin{array}{l} \mathcal{A} = \{1, 2\} \\ \Sigma = \{(X, Y) \mid X \cup Y = X_0 \cup Y_0\} \\ \pi_1(\sigma) = C(X) \quad X \subseteq X_0 \cup Y_0 \\ \pi_2(\sigma) = C(Y) \quad Y \subseteq X_0 \cup Y_0 \end{array}\right\} \tag{3.2}$$

The two agents may wish to cooperatively reallocate their tasks if there exists a new allocation (X, Y) such that:

$$\pi_1(X) > \pi_1(X_0)$$

and

$$\pi_2(Y) > \pi_2(X_0)$$

However, if there exist more than one such allocation, then which of them should be chosen by them as an agreement of reallocation?

3.2.3 Distributed Constraint Satisfaction Problem (DCSP)

The postman domain (see Example 3.2) and the cake-sharing domain (see Example 3.1 can all be classified as a Constraint Satisfaction Problems (CSPs) as their variable assignments are limited in one way or another. For example, in the cake-sharing domain, $x + y = 1$ is a constraint on assignment of x and y. In fact, most DAI problems can be represented as CSPs.

Constraint Satisfaction Problem (CSP)

A CSP problem can be described by

$$< X,\ C,\ \pi >$$

where X is a set of variables, and C is a set of constraints that must be satisfied and π is the global objective function that evaluates assignments of X to indicate preference of the problem solving. We call C *the hard constraint* and π *the soft constraint.*
C decides the possible solution space Σ:

$$\Sigma = \{X|C(X) = \text{TRUE}\} \tag{3.3}$$

So, each member, σ ($\in \Sigma$), of Σ is an legal assignment of X.
The target of Solving the CSP is to find an assignment, namely σ^*, (1) that is a member of Σ and (2) that $\pi(\sigma^*)$ yields the maximal value, i.e.,

$$\forall\sigma(\in \Sigma)\pi(\sigma^*) \geq \pi(\sigma) \tag{3.4}$$

When the size of Σ is very large, efficient searching approaches are essential to guarantee that an optimal solution can be reached within a time allowed. In some difficult CSPs, it is not computationally possible to guarantee an optimal solution. In this case, a sub-optimal solution will be sought. Heuristics are often used to increase the possibility of obtaining a sub-optimal solution and the quality of the solution.

Distributed Constraint Satisfaction Problem (DCSP)

When a CSP is naturally (i.e, logically or geometrically) distributed, then the problem could possibly be solved in DAI approaches. We describe the Distributed Constraint Satisfaction Problem (DCSP) as

$$< \mathcal{A}, \overbrace{< X_i,\ C_i, \pi_i >}^{CSP_i},\ C_{ij},\ \pi\ (i, j \in \mathcal{A}) >$$

Basically, a DCSP problem is composed by a set of CSP problems, CSP_i ($i \in \mathcal{A}$) and a set of inter-agent constraints, C_{ij} ($i, j \in \mathcal{A}$) concerning the relation between X_i and X_j and a global objective function π evaluating synthetically the overall assignments of all X_i ($i \in \mathcal{A}$). For each subproblem, CSP_i, it has a solution space Σ_i:

$$\Sigma_i = \{X_i\ |C_i(X_i) = \text{TRUE}\} \tag{3.5}$$

And a solution σ_i^* is said to be optimal for agent i, if

$$\forall \sigma_i \ (\in \Sigma_i) \pi(\sigma_i^*) \geq \pi(\sigma_i) \tag{3.6}$$

For the whole DCSP problem, the solution space is Σ:

$$\Sigma = \{(X_1, X_2, ..., X_i, ...) \mid X_i \in \Sigma_i; \ C_{ij}(X_i, X_j) = \text{TRUE} \ (i, j \in \mathcal{A})\} \tag{3.7}$$

A solution σ^* $(\in \Sigma)$ is said to be optimal if

$$\forall \sigma \ (\in \Sigma) \pi(\sigma^*) \geq \pi(\sigma) \tag{3.8}$$

Solving DCSP in a Multi-agent System (MAS)

We are interested in a class of the DCSP problem in which π has the following property:

$$\frac{d\pi}{d\pi_i} \geq 0 \tag{3.9}$$

That is to say, each local objective π_i is related with the global objective π such that the value of π can be increased by increasing the value of π_i. In this case, the whole DCSP might be solved in a multi-agent system: each agent, namely i, solves one CSP problem CSP_i, aiming at maximising π_i.

But these agents cannot solve their individual problems independently. Their local assignments are constrained by inter-agent constraints C_{ij}: two agents' local optimal assignments might not satisfy their inter-agent constraints and hence are not legal in the global sense. Such a potential conflict could be resolved by these conflicting agents' cooperation.

However, how can it be possible for agents with their own local objectives to cooperate? This is the our main concern in this chapter. When discussing agents' cooperative interaction, we can view the DCSP as a CIP Problem

$$< \mathcal{A}, \Sigma, < \pi_1, \pi_2, ..., \pi_i, ... > \ (i \in \mathcal{A}) > \tag{3.10}$$

Note that the global objective $\pi(\sigma)$ $(\sigma \in \Sigma)$ has been omitted in Equation 3.10. It will be considered in later sections in this chapter.

3.3 Notations, Definitions and Assumptions

Definition 3.1 Individual Preference *In a CIP situation,* $<\mathcal{A},\Sigma,<\pi_1,\pi_2,...>>$, *suppose* i *is an agent in* $\mathcal{A}$, σ *and* $\sigma^{'}$ *are two arbitrary strategies, then we say*

1. σ *is preferable to* $\sigma^{'}$ *if*

$$\pi_i(\sigma) > \pi_i(\sigma^{'}) \tag{3.11}$$

This relation is denoted by

$$\sigma \succ i \quad \sigma^{'}$$

2. σ *is indifferent to* $\sigma^{'}$ *if*

$$(\sigma \not\succ i \quad \sigma') \wedge (\sigma' \not\succ i \quad \sigma) \tag{3.12}$$

This relation is denoted by

$$\sigma \approx i \quad \sigma^{'}$$

3. *We also use* $\sigma \succeq i \quad \sigma^{'}$ *to denote:*

$$(\sigma \succ i \quad \sigma^{'}) \vee (\sigma \approx i \quad \sigma^{'})$$

Definition 3.2 Group Preference *In a CIP situation,* $<\mathcal{A},\Sigma,<\pi_1,\pi_2,...>>$, *suppose* σ *and* $\sigma^{'}$ *are two arbitrary strategies, then we say*

1. σ *is preferable to* $\sigma^{'}$ *for* $\mathcal{A}$, *if*

$$\forall i(\in \mathcal{A})\ \exists j(\in \mathcal{A})\ (\sigma \ \succeq i \quad \sigma^{'}) \wedge (\sigma \succ j \quad \sigma^{'}) \tag{3.13}$$

In other words, for all agents, σ *is preferable or indifferent to* $\sigma^{'}$ *and there exists at least one agent for which* σ *is preferable to* $\sigma^{'}$. *This relation is denoted by* $\sigma \succ \mathcal{A} \quad \sigma^{'}$.

2. σ *is indifferent to* $\sigma^{'}$ *for* $\mathcal{A}$, *if*

$$\forall i(\in \mathcal{A})\ \sigma \approx \mathcal{A} \quad \sigma^{'}. \tag{3.14}$$

In other words, for all agents in $\mathcal{A}$, σ *is indifferent to* $\sigma^{'}$.

3. *We also use* $\sigma \succeq \mathcal{A} \quad \sigma'$ *to denote*

$$(\sigma \succ \mathcal{A} \quad \sigma') \vee (\sigma \approx \mathcal{A} \quad \sigma') \tag{3.15}$$

Assumption 3.1 Rationality: *All agents in* $\mathcal{A}$ *are rational. We say an agent* i *is rational if its intention, action and preference are consistent. More specifically, suppose* σ *and* σ' *are two possible strategies known to agent* i*, and* $\sigma \succ i \quad \sigma'$ *then it will not intend to use strategy* σ'*, and further more it will not use strategy* σ'*. In this case, we say* σ' *is* **irrational** *for agent* i*.*

Assumption 3.2 Benevolence *All agents in* $\mathcal{A}$ *are benevolent. We say an agent* i *in an agent set* $\mathcal{A}$ *is benevolent if the agent will help other agents under condition that its own goal is not affected. More specifically, suppose* σ *and* σ' *are two possible strategies known to agent* i*, and* $\sigma \succ \mathcal{A} \quad \sigma'$ *is also known to agent* i*, then all agents in* $\mathcal{A}$ *will not intend to use strategy* σ'*, and further more it will not use strategy* σ'*. In this case, we say* σ' *is irrational for* $\mathcal{A}$*.*

Assumption 3.3 Full-Informedness

(1) *Each agent in* $\mathcal{A}$ *knows it is rational and benevolent;*

(2) *Each agent assumes that all other agents are rational and benevolent;*

(3) *Each agent has complete knowledge about the problem domain* Σ*;*

(4) *Each agent assumes that all other agents have complete knowledge about the problem domain* Σ*;*

(5) *Each agent has complete knowledge about its own payoff function;*

(6) *Each agent assumes that all other agents know their own payoff functions;*

(7) (1) (2) (3) (4) (5) (6) *are common sense, i.e., each agent assumes they are true and it assumes other agents assume they are true as well.*

We say Assumption 3.2 is a benevolent agent assumption for the following consideration:

Suppose there are two agents, agent 1 and agent 2, and they have to jointly make a decision on which strategy to choose from a strategy set of two strategies, A and B; suppose also that $A \approx 1 \quad B$ and $A \succ 2 \quad B$. In this situation, agent 1 can choose any of the two strategies, because they are indifferent to each other from agent 1's perspective. If he is hostile, or malevolent, then he might choose B which is less preferable to A in agent 2's perspective. However, by Assumption 3.2, then agent 1 has to agree on A which (1) will not affect his own payoff compared with B, (2) is preferable to B in agent 2's perspective. In other words, agent 1 helped agent 2 by choosing agent 2's preferable strategy while its own payoff has not been affected negatively. Therefore, we say agent 1 is benevolent.

The benevolent agent assumption was first seen in [Rosenschein & Genesereth 88] where the benevolent agent assumption was described as "All agents are fundamentally assumed to be helping one another, and will trade data and hypotheses as well as carry out tasks that are requested of them". Since there was no formal definition about what does the term "benevolent" mean, this assumption is criticised in [Durfee *et al.* 87] as "Rosenschein and Genesereth misleadingly call agents that share goals benevolent agents". Following this criticism, Durfee and his co-authors present another view point — "Actually, these agents are completely self-interested since each performs actions only to satisfy its own local interpretations of these goals. Benevolence is neither assumed nor needed for the agents to cooperate." We think this arguing can be stopped only if the term benevolent is clearly defined. We assume agents are benevolent only according to Assumption 3.2.

Relevant to Assumption 3.2 is selfish agent assumption, malevolent agent assumption, altruistic agent assumption, etc. These assumptions are very often used in human negotiations. But we think that are not necessarily assumed. For example, the altruistic agent assumption requires each agent to sacrifice its own goal achievement for the sake of other agents' goal achievement. This is contradictory to the rational agent assumption which requires each agent to aim at its own goal achievement.

3.4 Cooperative Games

In this section, we will show another example of the CIP problem: cooperative games.

3.4.1 Brief Introduction to Game Theory

Game theory can be defined as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers. Game theory provides general mathematical techniques for analysing situations in which two or more individuals make decisions that will influence one another's welfare. As such, game theory offers insights of fundamental importance for scholars in all branches of the social sciences, as well as for practical decision-makers. In this section we introduce some basic concepts about game theory. Detailed knowledge about game theory can be found from [vonNeumann & Morgenstern 44] [Luce & Raiffa 57][Myerson 91] [Shapiro 89], etc. We limit our introduction to the two-person game.

3.4.2 Basic Strategic Model of Two Person Game

The two person-game can be modelled by a payoff matrix: There are two agents (or

agent 2

		σ_1^2	σ_1^2
agent 1	σ_1^1	(π_1^1, π_1^2)	(π_1^1, π_2^2)
	σ_2^1	(π_2^1, π_1^2)	(π_2^1, π_2^2)

Figure 3.1: Two-Person Game Payoff Matrix

players), namely agent 1 and agent 2, meeting an interactive situation where each agent i $(\in \mathcal{A} = \{1, 2\})$ has two strategies, σ_1^i and σ_2^i. This game matrix can be easily extended to a two-person game in which each agent has more than two strategies. π_{lk}^i is the payoff for agent i when agent 1 uses strategy σ_l^1 and agent 2 uses σ_k^2. If

$$\forall l, k \;\; \pi_{lk}^1 + \pi_{lk}^2 = 0 \tag{3.16}$$

then the game is called the zero-sum game, otherwise it is the non-zero-sum game. In a zero-sum game, agents' interests are diametrically opposed — one agent's gain is exactly equal to the loss of the other agent. Therefore, game theorists say that there is no cooperation possible in the zero-sum game. We will not follow this assumption in this dissertation. We treat both types of the two-person game in the same way. Example 3.3 shows an interaction situation called the Battle of the Sexes [Thomas 84]:

Example 3.3 The Battle of the Sexes: *A married couple are trying to decide where to go for a night out. She would like to go to the theatre, and he would like to go to a football match — they have been married a few months! However, they are still very much in love and so they only enjoy the entertainment if their partner is with them. If the first strategy, namely* σ_1^i, *for each is to go to the theatre, the second to go to the football match, agent* 1 *represents the wife and agent* 2 *represents the husband, then payoff is shown in Figure 3.2.*

agent 2

		σ_1^2	σ_2^2
agent 1	σ_1^1	$(4,1)$	$(0,0)$
	σ_2^1	$(0,0)$	$(1,4)$

Figure 3.2: Payoff Matrix of the Battle of the Sexes

3.4.3 Concept of Equilibrium

In game theory, agents are assumed to be rational, fully informed. Another important assumption is that communication between agents is prohibited and agents must decide which of their individual strategies to use simultaneously. What is the best strategy for each agent? This is the major concern of game theory. The concept of an equilibrium is the answer to this question. An equlibrium is a strategy pair (σ_*^1, σ_*^2) (σ_*^1 is a strategy of agent 1, σ_*^2 is a strategy of agent 2) such that no agent will get a higher payoff by moving to any of its another strategies. In Example 3.3, (σ_1^1, σ_1^2) (the couple both go to the theatre) is an equilibrium pair, since, if any of then has definitely decided to go to the theatre, then the other will get a zero payoff (the lowest) if she/he decides to go to the football match instead of going to the theatre. If there exists only one equilibrium pair, then it seems each agent can choose its strategy from the equilibrium pair. Unfortunately, a game often has more than one equilibrium pair. In Example 3.3, for obvious reason, (σ_2^1, σ_2^2) (both go to the football match) is a equilibrium pair as well.

3.4.4 Mixed Strategies

Each agent may not just simply use the pure strategies — going to the theatre and going to the football match. They can use mixed strategies. For example, going to the theatre with a probability of 30% and going to the football with a probability of 70%. If the two agents are not willing to cooperate, then they might use a fixed pattern:

$$\sigma_i =< p_1^i, p_2^i > \quad (i \in \{1, 2\})$$

where p_1^i is a probability with which agent i is going to the theatre and p_2^i is a probability with which agent i is going to the football, we have

$$p_1^i + p_2^i \equiv 1$$

Pure strategies are the special cases of the mixed strategy: $< 1, 0 >$ means going to the theatre and $< 0, 1 >$ means going to the football match. When $< p_1^i, p_2^i >$ is used, the payoff, π_i for agent i is calculated by the expected payoff:

$$\pi_i = \sum_{a=1}^{2} \sum_{b=1}^{2} \pi_i(\sigma_a^1, \sigma_b^2) p_a^1 p_b^2 \tag{3.17}$$

where $\pi_i(\sigma_a^1, \sigma_b^2)$ is the payoff for agent i if agent 1 uses pure strategy σ_a^1 $(a = 1, 2)$ and agent 2 uses pure strategy σ_b^2 $(b = 1, 2)$. It has been proved that when mixed strategies are used, the Battle of the Sexes has a third equilibrium (σ_1^*, σ_2^*) [Owen 82].

$$\left.\begin{array}{l} \sigma_1^* =< \frac{4}{5}, \frac{1}{5} > \\ \sigma_2^* =< \frac{1}{5}, \frac{4}{5} > \end{array}\right\} \tag{3.18}$$

3.4.5 Cooperative Games

The prisoner's dilemma (see Example 3.4) [Luce & Raiffa 57] is often used by game theorists as an example of cooperative game.

Example 3.4 Prisoner's Dilemma*:*

Two prisoners, agent 1 and agent 2, are accused of conspiring in two crimes, one minor crime for which their guilt can be proved without confession, and one major crime for which they can be convicted only if at least one confesses, the confessor will go free now but the other will go to jail for 6 years. If both confess, then they both go to jail for 5 years. If neither confesses then they both go to jail for only 1 year. So each agent i has two possible strategies: to confess (σ_1^i) or to deny (σ_2^i). The payoffs, measured in the number of years of freedom that the player will enjoy over the next 6 years, are shown in the payoff matrix in Figure 3.3.

agent 2

		σ_1^2 (deny)	σ_1^2 (confess)
agent 1	σ_1^1 (deny)	$(5, 5)$	$(0, 6)$
	σ_2^1 (confess)	$(6, 0)$	$(1, 1)$

Figure 3.3: Payoff Matrix of the Prisoner's Dilemma

Intuitively we would possibly agree that (deny, deny) is the best strategy pair for both prisoners. However, this game has a unique equilibrium pair that is (confess, confess): From agent 1's point of view, he thinks if agent 2 uses deny strategy, then the best strategy for him is to confess so that he will receive a payoff of 6 larger than 5, a payoff for him if deny is used; also if agent 1 uses confess then the best strategy for him is to confess as well since he will receive a pay of 1 instead of 0. Similarly, from agent 2's point of view, confess is the best strategy too. Therefore, the game will converge to (confess, confess) which is in fact a strategy pair worse than (deny, deny) from both agents' point of view. This is a typical example in game theory — rational agents' good will at the beginning will turn out a bad result. The problem arises as communication is not allowed between them. If they can communicate before they choose their strategies, they might not choose (confess, confess). There might be some cooperation possible. For example, they might sign an agreement on using (deny, deny) and act accordingly. And indeed, many game theorists suggest that (deny deny) is the best cooperative strategy pair (e.g., [Thomas 84] (page 64), [Myerson 91] (page 97), etc.). *Cooperation in cooperative games means that both agents choose unanimously a joint strategy.* For example, in the Prisoner's Dilemma, there are four possible pure joint strategies: (deny, deny), (confess, confess), (deny, confess) and (confess, deny). A mixed cooperative strategy has the similar meaning to that of an agent's mixed strategy. A mixed cooperative strategy σ can be represented by

$$\sigma =< p_1, p_2, ..., p_n, ... >$$

with

$$\sum_i p_i \equiv 1$$

where p_n is the probability with which a pure cooperative strategy, e.g., (deny, deny), is used. Agent i's payoff is calculated by expected payoff:

$$\pi_i = \sum_n \pi_i^n p_n \tag{3.19}$$

where π_i^n is the payoff for agent i when a respective pure cooperative strategy is used. A cooperative game can be then described by a CIP problem $< \mathcal{A}, \Sigma \, \pi >$:

$$\left.\begin{array}{l} \mathcal{A} = \{1, 2\} \\ \Sigma = \{< p_1, p_2, ..., p_n, ... > \ |0 \leq p_n \leq 1; \sum_n p_n = 1\} \\ \pi_i = \sum_n \pi_i^n p_n \ (i \in \mathcal{A}) \end{array}\right\} \tag{3.20}$$

For example,

pure strategy	Index n	probability p_n	payoff π_1^n	payoff π_2^n
(confess, deny)	1	p_1	6	0
(deny, deny)	2	p_2	5	5
(deny, confess)	3	p_3	0	6
(confess, confess)	4	p_4	1	1

Figure 3.4: Cooperative Game for Prisoner's Dilemma

Generally, it is possible for cooperative agents to get higher payoffs than that they can achieved from a non-cooperative (independent game). For example, Figure 3.5 shows the non-cooperative region of payoff pairs and the cooperative region of payoff pairs for the Battle of the Sexes. The cooperative region is extended from the non-cooperative region. So, the whole area of triangle is the cooperative region.

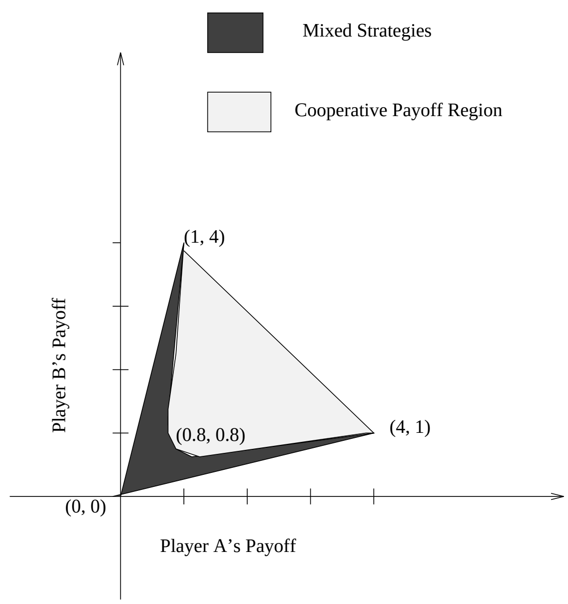


Figure 3.5: The Cooperative and Non-cooperative Payoff Regions in the Battle of the Sexes

3.5 Summary So far

In previous sections in this chapter, we formally addressed the CIP problem. We showed also that many DAI problems can be regarded as CIP problems. Cooperative games are CIP problems too. We also made several assumptions of agents. All agents are rational, benevolent, and fully informed. We defined a set of preference notations which would be useful for the discussion in the rest of the chapter and some other chapters that follow.

3.6 Decision-making in a Single Agent Environment

Suppose a rational agent i has the total control over decision-making on which strategy to choose from a problem domain Σ, then the problem is already theoretically solved. The agent will choose one strategy from Σ that is the most preferable.

Definition 3.3 *In a CIP situation $< \mathcal{A}, \Sigma, < \pi_1, \pi_2, ... >>$, suppose i is an agent in $\mathcal{A}$, σ^* is a member of Σ, then we say σ^* is the most preferable for agent i, if*

$$\forall \sigma (\in \Sigma) \sigma_i^* \succeq i \quad \sigma$$

In practice, we may obtain one of the most preferable strategies from the remainder of Σ by moving all strategies that are irrational.

Definition 3.4 *A strategy $\sigma(\in \Sigma)$ is irrational for agent i, if there exists another strategy $\sigma^{'}(\in \Sigma)$ such that*

$$\sigma^{'} \succ i \quad \sigma$$

We denote a procedure for removing all irrational strategies by

$$\textbf{remove-irrationals}(\Sigma, i)$$

which can be an algorithm as simple as follow:

Algorithm 3.1

procedure remove-irrationals(Σ, i)

1 until all strategy in Σ has been considered do

2 Select an strategy, namely σ not considered before

3 if $\exists\sigma^{'}(\in \Sigma)\sigma^{'} \succ i \quad \sigma$ (σ is therefore irrational) then

4 $\Sigma \Leftarrow$ **remove**(σ, Σ) **endif**

5 enduntil

6 Return Σ

endprocedure

When procedure **remove-irrationals**(Σ, i) is applied to a strategy set Σ, all irrational strategies will be deleted from it and the remaining strategies in the set are therefore rational.

Theorem 3.1 *In a CIP situation $< \mathcal{A}, \Sigma, < \pi_1, \pi_2, \ldots >>$, suppose i is an agent ($i \in \mathcal{A}$), and Σ_i^* is a subset of Σ given by:*

$$\Sigma_i^* = \textbf{remove-irrationals}(\Sigma, i) \tag{3.21}$$

we have

1. *If* Σ *is not empty, then* Σ_i^* *consists of at least one strategy, formally,*

$$\Sigma \neq \phi \Rightarrow \Sigma_i^* \neq \phi \tag{3.22}$$

2. *If there exists two or more strategies in* Σ_i^* *then any strategy in it is indifferent to others, i.e.,*

$$\forall \sigma \sigma^{'} (\in \Sigma_i^*) \sigma \approx i \ \ \sigma^{'} \ \ (|\Sigma_i^*| \geq 2) \tag{3.23}$$

where $|\Sigma_i^*|$ *is the size of* Σ_i^**.*

3. *Any* σ_i^* *in* Σ_i^* *is the most preferable for agent* i*, i.e.,*

$$\sigma_i^* \in \Sigma_i^* \Rightarrow \forall \sigma (\in \Sigma) \sigma_i^* \succeq i \ \ \sigma \tag{3.24}$$

Proof:

(1) Suppose there is only one strategy in Σ, then **remove-irrationals**(Σ, i) procedure will simply return Σ. If there is more than one strategy in Σ, and suppose **remove-irrationals**(Σ, i) will return an empty set, then, since strategies are moved from Σ one by one, there is definitely a time when there is only one strategy in Σ. However, when there is only one strategy left in Σ, the procedure will simply return it and Σ^* is therefore not empty. This is contradictory to the hypothesis that Σ^* is empty.

(2) Suppose there are two strategies, σ and $\sigma^{'}$, in Σ, such that

$$\sigma \not\approx i \ \ \ \sigma^{'}$$

Then it means

$$(\sigma \succ i \ \ \sigma^{'}) \vee (\sigma^{'} \succ i \ \ \sigma)$$

Suppose $\sigma \succ i \ \ \sigma^{'}$ then $\sigma^{'}$ is irrational and should be removed from Σ^*. Similarly, if $\sigma^{'} \succ i \ \ \sigma$ then σ is irrational and should be removed from Σ^*. This contradicts. $\Box$.

So, all strategies in Σ_i^* are the optimal strategies for agent i, therefore agent i can choose any strategy from Σ_i^* as its best plan. So, theoretically, **remove-irrationals**(Σ, i) can be taken as a single agent decision-making procedure which will yield a set of strategies that is the most preferable for agent i.

(3) Suppose there exists σ $(\in \Sigma)$ such that

$$\sigma \succ i \ \ \sigma_i^*$$

then σ_i^* should have been removed. This is contradictory to the hypothesis that σ_i^* is in Σ_i^*. $\Box$.

3.7 Joint Decision-making

We are interested in the CIP problem where there is more than one agent involved. What is the best cooperative strategies for these agents? We would like to obtain such strategies by removing all irrational strategies from Σ.

3.7.1 Ideal Cooperative Interaction

In last section, we illustrated that in a single agent environment, agent i selects a best strategy from a set Σ_i^* which is the remainder of Σ by removing all irrational strategies. We also proved that all strategies in Σ_i^* are indifferent to one another. Let us now consider the CIP problem $< \mathcal{A}, \Sigma, \pi >$ when there is more than one agent in $\mathcal{A}$. Let us first define Σ^* as a set of unification of Σ_I^* $(i \in \mathcal{A})$, i.e.,

$$\Sigma^* = \bigcap_{i \in \mathcal{A}} \Sigma_i^* \tag{3.25}$$

then there are two possibilities:

1. $\Sigma^* \neq \phi$ — **the Ideal cooperative interaction**: In this case, there is at least one member in Σ^*. Since, according to Equation 3.25, every member, namely σ^*, in Σ^* should be in Σ_i^* $(i \in \mathcal{A})$, σ^* is one of the most preferable strategies for all agents in $\mathcal{A}$. In this case, the cooperation is ideal and becomes simple: agents only need to coordinate to find a member of Σ^*.

2. $\Sigma^* = \phi$ — **the Difficult cooperative Interaction**: In this case, there does not exist any strategy that is the most preferable for all agents. In this case, cooperation will be difficult.

We are interested in the difficult cooperative interaction in which there does not exist any strategy that is the most preferable to all agents in $\mathcal{A}$.

3.7.2 Cooperative Interaction between Benevolent Agents

Definition 3.5 *In a CIP situation* $< \mathcal{A}, \Sigma, < \pi_1, \pi_2, ..., \pi_i, ... >>$, *we say a cooperative strategy* σ $(\in \Sigma)$ *is irrational for* $\mathcal{A}$ *if and only if there exists a strategy* $\sigma^{'}(\in \Sigma)$ *such that* $\sigma^{'}$ *is preferable to* σ *for* $\mathcal{A}$, *i.e.,*

$$\sigma^{'} \succ \mathcal{A} \quad \sigma \tag{3.26}$$

Borrowing the way we used to deal with decision-making in a single agent environment, we construct a set $\Sigma^*_{\mathcal{A}}$:

$$\Sigma^*_{\mathcal{A}} = \textbf{remove-irrationals}(\Sigma, \mathcal{A}) \tag{3.27}$$

Then $\Sigma^*_{\mathcal{A}}$ is the remainder of Σ by removing all irrational strategies for $\mathcal{A}$.

Theorem 3.2

$$\Sigma^* \neq \phi \Rightarrow \Sigma^*_{\mathcal{A}} = \Sigma^* \tag{3.28}$$

In other words, if there exist strategies in Σ *that are the most preferable for* $\mathcal{A}$, *then* **remove-irrationals**$(\Sigma, \mathcal{A})$ *is a feasible procedure for choosing all of these strategies.*

Proof:

(1) Suppose σ^* is a member of Σ^* then it is a member of Σ^*_i $(i \in \mathcal{A})$ according to Equation 3.25. Since σ^* is a member of Σ^*_i $(i \in \mathcal{A})$, according to Theorem 3.1, we have,

$$\forall i(\in \mathcal{A})\sigma(\in \Sigma)\sigma^* \succeq i \quad \sigma \tag{3.29}$$

This is equivalent to

$$\forall i(\in \mathcal{A}) \ \nexists\sigma(\in \Sigma)\sigma \succ i \quad \sigma^* \tag{3.30}$$

This is equivalent to saying,

$$\nexists\sigma(\in \Sigma)\sigma \succ \mathcal{A} \quad \sigma^* \tag{3.31}$$

Therefore, according to Definition 3.5, σ^* is not irrational for $\mathcal{A}$, therefore, it should not be removed from Σ when **remove-irrationals**$(\Sigma, \mathcal{A})$ is applied. Therefore, σ^* should be a member of $\Sigma^*_{\mathcal{A}}$. We can now conclude

$$\Sigma^* \subseteq \Sigma^*_{\mathcal{A}} \tag{3.32}$$

(2) Suppose a strategy, namely $\sigma^*_{\mathcal{A}}$ is a member of $\Sigma^*_{\mathcal{A}}$, then

$$\nexists\sigma(\in \Sigma)\sigma \succ \mathcal{A} \;\; \sigma^*_{\mathcal{A}} \tag{3.33}$$

since, otherwise, it will be removed. However, Equation 3.33 is equivalent to

$$\forall i(\in \mathcal{A}) \;\; \nexists\sigma(\in \Sigma) \succ i \;\; \sigma^*_{\mathcal{A}} \tag{3.34}$$

or

$$\forall i(\in \mathcal{A}) \;\; \sigma(\in \Sigma)\sigma^*_{\mathcal{A}} \succeq i \;\; \sigma^*_{\mathcal{A}} \tag{3.35}$$

Therefore, $\sigma^*_{\mathcal{A}}$ should be a member of each Σ^*_i $(i \in \mathcal{A})$ according to Definition 3.3. Since Σ^* is the unification of Σ^*_i $(i \in \mathcal{A})$, therefore we conclude

$$\Sigma^*_{\mathcal{A}} \subseteq \Sigma^* \tag{3.36}$$

(3) According to Equation 3.32 and Equation 3.36, we conclude

$$\Sigma^*_{\mathcal{A}} = \Sigma^* \tag{3.37}$$

□

Theorem 3.2 says that if there exists any strategy that is the most preferable for all agents in $\mathcal{A}$, then agents can cooperatively obtain it by assuming that they are all benevolent.

Theorem 3.3

$$\Sigma \neq \phi \Rightarrow \Sigma^*_{\mathcal{A}} \neq \phi \tag{3.38}$$

In other words, if Σ is not empty, then there exists at least one member in $\Sigma^*_{\mathcal{A}}$.

Proof

The proof is similar to Theorem 3.1. □

Theorem 3.4 *Suppose there are more than two members in $\Sigma^*_{\mathcal{A}}$, then*

$$\forall\sigma \;\; \sigma^{'}(\in \Sigma^*_{\mathcal{A}})\exists i(\in \mathcal{A})\sigma \succ i \;\; \sigma^{'} \Rightarrow \exists j(\in \mathcal{A})\sigma^{'} \succ j \;\; \sigma) \tag{3.39}$$

In other words, if there exist two arbitrary strategies, namely σ and $\sigma^{'}$ in $\Sigma^*_{\mathcal{A}}$ such that σ is preferable to $\sigma^{'}$ for one agent, namely agent i, then there exist at least one agent, namely agent j such that $\sigma^{'}$ is preferable to σ for agent j.

Proof

Suppose there does not exist an agent j such that

$$\sigma^{'} \succ j \quad \sigma \tag{3.40}$$

then we will have

$$\forall j(\in \mathcal{A})\sigma \succeq j \quad \sigma^{'} \tag{3.41}$$

since we also assume that

$$\sigma \succ i \quad \sigma^{'} \tag{3.42}$$

According to Definition 3.5, we will have

$$\sigma \succ \mathcal{A} \quad \sigma^{'} \tag{3.43}$$

And therefore, $\sigma^{'}$ is a irrational strategy for $\mathcal{A}$ and it must not be in $\Sigma^{*}_{\mathcal{A}}$. This is contradictory to our hypothesis. □

3.7.3 Discussion

Theorem 3.3 and Theorem 3.4 illustrate what rational, benevolent agents can do and what they cannot do. More specifically, when agents in $\mathcal{A}$ all have a strategy that is the most preferable for them, then Theorem 3.3 says that this strategy can be achieved by assuming agents are rational, benevolent. For example, in a game with the payoff matrix illustrated in Figure 3.6, agent 1 could easily drop σ^1_2 and choose strategy σ^1_1 which is the most hopeful for agent 2 which would like to choose σ^2_1 when agents are assumed to be benevolent. If agents are not assumed to be benevolent, then agent 1 could possibly choose σ^1_2 by which its payoff is also guaranteed to be 10. However, in many interactive situations cooperation is not expected to be as easy as given in Figure 3.6. In Example 3.3, when they are assumed to be benevolent, they can easily drop plans that say that they do not go together. However, they could not be able to unanimously concentrate on any of the two remaining interactions: going to the theatre together or going to the football match together.

		agent 2	
		σ_1^2	σ_1^2
agent 1	σ_1^1	$(10, 10)$	$(9, 0)$
	σ_2^1	$(10, 0)$	$(10, 0)$

Figure 3.6: An Example of Ideal Cooperative Interaction

3.8 The Idea of a Common Sense Principle

3.8.1 Solutions to Cooperation from Game Theorists

Perhaps the most important contribution to cooperative games is the Nash Solution to the Two-person Bargaining Set [Nash 51]. Nash constructed a bargaining set which consists of several possible joint cooperative strategies for the the two players, agent 1 and agent 2. For each strategy agent i ($\in \mathcal{A} = \{1, 2\}$) receives a payoff of $v_i (\in \Re)$. The payoff pair (v_1, v_2) is therefore a point in $\Re$x$\Re$ domain. Nash located all possible payoff pairs on a continuous convex region on the $\Re$x$\Re$ plane as shown in Figure 3.7 (each point on the convex is a payoff pair for a strategy). Nash defined several axioms each of which points out what a solution should be. For example, if the bargaining set is symmetric, (i.e., existence of payoff pair (a, b) means the existence of payoff (b, a)), then the solution should be symmetric as well. According to these Axioms, Nash proved that there exists a unique solution to the bargaining set, and the solution maximises $f(v_1, v_2)$

$$f(v_1, v_2) = (v_1 - v_1^0) * (v_2 - v_2^0)$$

For each strategy agent i ($\in \mathcal{A} = \{1, 2\}$) receives a payoff of $v_i (\in \Re)$. The payoff pair (v_1, v_2) is therefore a point in $\Re$x$\Re$ domain. It is also required that possible payoff pairs should be located on a continuous convex region on the $\Re$x$\Re$ plane as shown in Figure 3.7 (each point on the convex is a payoff pair for a strategy). As pointed out by Harsanyi [Harsanyi 77], Nash's solution is equilibrium cooperation. We can consider the Nash Solution is obtained in the following way: the axioms limit the

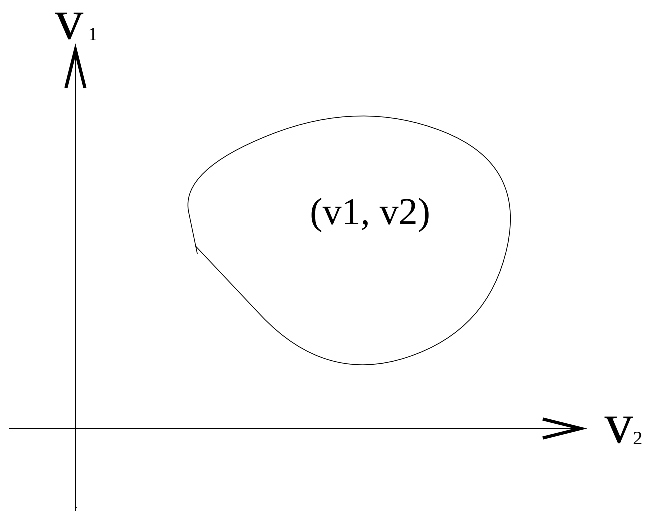


Figure 3.7: The Nash Bargaining Set

agents' attempts to achieve higher payoffs, while rational agent assumptions encourage agents to achieve higher payoffs. Agents will finally agree on a strategy from which none of the agents has the incentive to move: such a move will either be inconsistent with the axioms or yield a lower payoff.

Focal point is another concept for cooperative game. Schelling [Schelling 60] argued that, in a game with multiple equilibria, anything that tends to focus the players' attention on one equilibrium may make them all expect it and hence fulfill it. For example, in the Battle of the Sexes (see Example 3.3), if the couple are living in a maternal society, then they will all agree that ladies' wish should be more respected than men's. In this case, the couple may focus on the strategy of going to the theatre together.

3.8.2 Solutions to Cooperation from DAI Researchers

Assumption-based cooperative interaction by Rosenschein and Genersereth (e.g., [Rosenschein 85] [Rosenschein & Genesereth 88]) suggests that cooperative solutions can be obtained based on agents' behaviour motivated by their assumptions about themselves and about other agents. For example, in the Prisoner's Dilemma (see Example 3.4), Rosenschein and Genersereth argued that agents can finally concentrate on the (deny, deny) strategy according to reasoning about each others' behaviour based on their assumptions about each others' behaviour.

The focal point concept is also studied in DAI [Kraus & Rosenschein 92] [Fenster *et al.* 95]. For example, in DAI [Kraus & Rosenschein 92], a model of cooperative interaction is presented in which agents are given knowledge about focal points which are the distinguishing properties of strategies. Uniqueness, for example, may become a focal point that could guide two conflicting agents' cooperation to converge to a cooperative strategy. The following example which we quoted from [Kraus & Rosenschein 92] may explain to the reader further the concept of a focal point:

> Imagine two players on a TV game show. The MC explains to the players the simple rules of the game: each is to go to a separate, private room, where they will be handed a pile of 100 $1 bills. They are each, in isolation, to divide the single pile into two piles, A and B, with any distribution of bills between the piles. Their distributions will then be announced, and if they are identical (i.e., the players' A piles are the same size, and their B piles are the same size), they will each win a Mercedes. If their distributions are not identical, they will receive the consolation prize (a home version of the game). ...
> ... experiments with a game of this type [shows] that the overwhelming majority of players chose to divide the 100 $1 bills into two equal piles, 50 bills in each.

Hence, the strategy $< 50, 50 >$ becomes the focal point of the game.

3.8.3 Cooperative Interaction in Open systems and Non-open Systems

Game theory, with its interesting name and interesting models, has fascinated many researchers in many areas such as social science, psychological studies, economics studies, etc throughout the past several decades. Recently, game theory has become an interesting topic in DAI field (e.g. [Ephrati & Rosenschein 94], [Zlotkin & Rosensein 96], [Kraus & Wilkenfeld 90], etc.). One important assumption under which this work was carried on is that agents do not have a social goal. The research objective of this work is almost the same: investigating mechanisms that could guide cooperation between conflicting agents to converge to a resolution. In this work, there is a common implicit assumption: apart from agents' rational, benevolent and fully-informed assumptions, agents need have some more knowledge such as focal points. Where does this extra knowledge come from? It comes from the view points of these researchers and these view points come from these researchers' investigation on human behaviour and imagination (e.g. focal points, symmetrical strategies — if the two players are symmetrical (e.g., in the Battle of the Sexes), since the payoff matrix is symmetrical, therefore, the cooperative strategy must yield a symmetrical pair of payoffs — the two agents have to have the same payoff, no matter what cooperative strategies they choose). In DAI, in term of the global objective, there are mainly two types of the multi-agent system. The first type is called *the open system*, and the second type is what we call the non-open system. In an open system, there are no possibility for global control or global success criteria, or even a global representation of a system [Hewitt 86][Agha & Hewitt 85][Agha & Hewitt 86]. Open systems are systems that have the following features:

- They are composed of independently developed parts in continuous evolution;
- They are concurrent and asynchronous, and they have decentralised control based on debate and negotiation;
- They exhibit many local inconsistencies;
- They consist of agents with bounded knowledge and bounded influence;
- They have no fixed global boundaries visible to the agents constituting the system.

Since in an open system, there is no global success criteria, game-thoretical approaches to modelling agents' behaviour in conflicting situations may be suitable. A recent example is Zlotkin and Rosenschein's work on automated negotiation in stated oriented domains [Zlotkin & Rosensein 96]. They assume that automated agents are built by separate, self-interested designers and their research focuses on designing protocols for specific domains that will get those agents to interact in useful ways. In this cooperative interaction situation, interaction is carried out according to intentions of different designers. However, there are many DAI systems where global success evaluations are necessary. For example, a DCSP problem is often associated with a global goal (e.g., in [Adler *et al.* 89] the global objective is to maximise the total number of constraints being satisfied). In this case, agents' behaviour must be bounded to actions that can lead to the global goal achievement. So, for example, a DCSP with the global objective aforementioned, each agent must make its effort to satisfy as many of its own constraints as possible. In a case of a conflict in which two or more agents' local solutions are not compatible, then they must agree on an agreement that (1) is locally consistent, (2) is globally consistent, and (3) satisfies the maximal number of constraints. Most application DAI models have global objectives, they can be explicitly expressed or implicitly described. These global objectives come from the user's requirements. For example, in the Battle of the Sexes, game theorists could not find a cogent decision on choosing going to the theatre together or going to the football together. We may suggest many different *reasonable* ways of solving the problems (e.g., let the two agents toss a coin to decide where they will go). When there is no social goal for the two agents, we may say any solution is good or we may say none of them is reasonable. So, what is the best solution is not discussed between the two rational agents, but between ourselves — those who are studying the game. However, when there are social goals, then some games may be easy. For example, if we have a global objective $\pi(\sigma)$ which is the sum of the wife's utility $\pi_1(\sigma)$ and the husband's utility $\pi_2(\sigma)$, i.e.,

$$\pi(\sigma) = \pi_1(\sigma) + \pi_2(\sigma) \tag{3.44}$$

then the arguing between the husband and the wife will be easily solved: either solution (going to the theatre together or going to the football together) is an optimal solution in the global sense, since they both yield the same value, 5, by the global objective function $\pi(\sigma)$. So, in this case, if the wife proposes going to the theatre then the husband should agree on it without delay. In other words, the husband's local objective (which suggests that it is better that they both go to the football match) should be subordinated to the global objective. To summarise the discussion in this paragraph we have:

- In an open system, when agents are in conflict, a conflict resolution can be sought according to a set of axioms or assumptions. These axioms or assumptions comes from the researchers' viewpoints which might come from these researchers' own social sense or come from their investigation on human behaviour exhibited in conflicting situations.
- In a non-open system, agents, like those in an open system, have their own local objective to achieve. Conflicts exist between them due to their local goals. If conflicts happen, then agents should subordinate their local objectives to the global objective.

3.8.4 Our Solution to Cooperation: Common Sense Principle

We have discussed that in non-open systems, there are global objectives expressed explicitly or implicitly. The knowledge about global objectives should be downloaded to each agent. However, since agents have already been downloaded with their own local objectives, there is possibility that these local objectives are in conflict with the global objective in some situations. As we mentioned before, in this case, agents' local objectives should be subordinated to the global objectives.

In most DAI problems, agents are computational entities created by the system designers to carry out some tasks in a distributed ways. Agents have been told their own objectives, and are expected to achieve their own objectives as well as possible so that the global objectives can be best served. In this sense, all agents must be rational. The benevolent agent assumption is also necessary for improving social performance. When agents are assumed to be rational and benevolent, to some extent, as we have proved in previous sections, agents are able to cooperate: they will be able to concentrate only on a subset, $\Sigma^*_{\mathcal{A}}$, of Σ. In this subset, all interactive strategies are *not* irrational. Sometimes, the size of $\Sigma^*_{\mathcal{A}}$ can be very small and we believe that the smaller the size of $\Sigma^*_{\mathcal{A}}$, the easier it will be for the cooperation to converge to a cooperative strategy. On the other hand, however large $\Sigma^*_{\mathcal{A}}$ is, as long as it consists of more than one strategy and agents have conflicting viewpoints about which strategy to choose, the cooperation will not possibly continue if agents are merely assumed to be rational and benevolent. *Agents must be given new knowledge.* The new knowledge we propose here is called *the* *common sense principle*. The idea of a common sense principle is described as follows:

> By rational and benevolent assumptions, $\Sigma^*_{\mathcal{A}}$ becomes the possible cooperative interaction strategies. Assume it is a difficult cooperative situation, i.e., $\Sigma^* = \phi$ (see Section 3.7.1), then any agent's move in attempt to gain a higher payoff will mean at least another agent's lost of payoff. In this case, cooperation will be guided by a common sense principle which suggests what a cooperative strategy should be (therefore it also implies what a cooperative strategy should not be). Once agents are given the common sense principle, they can continue their cooperation by dropping all those strategies that can not meet the principle. When there is only one strategy that satisfies the principle then the strategy will be the cooperative strategy. If no cooperative strategy can be found that meets the principle, then further common sense principle will be introduced to carry the cooperation forward until such a strategy is found.

Figure 3.8 illustrates the difference between approaches to CIP problems in open systems and approaches to CIP problems we proposed to non-open systems. A common sense principle takes part in a role the same as that of axioms or assumptions in open system approaches. However, a common sense principle is an incarnation of an global objective. For example, in the Battle of the Sexes, if the global objective is to maximise the sum of their individual payoffs, then a common sense principle might be "the strategy will be decided by a result of tossing a coin". Whichever the result of tossing a coin is, the sum is always 5. In this case, the result is decided by nature.

3.9 Example of Common Sense Principles

There are many ways of transforming an objective into other forms. The following are a few examples. We only consider two-agent cooperative interaction.

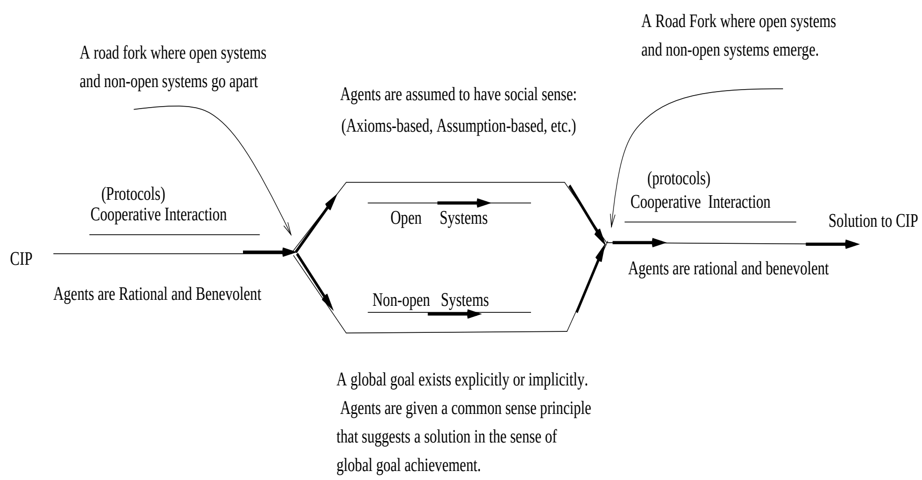


Figure 3.8: Road Forks of Open Systems and Non-open Systems

3.9.1 The Global Objective Function is the Sum of Local Objective Functions

In a CIP situation $<\mathcal{A}, \Sigma, <\pi_1, \pi_2>>$, suppose there is a global objective π

$$\pi(\sigma) = \sum_{i \in \mathcal{A}} \pi_i(\sigma) \quad (\mathcal{A} = \{1, 2\}) \tag{3.45}$$

Suppose Σ has a continuous domain and $\frac{d\pi_i}{d\sigma}$ exists $(i \in \mathcal{A})$, then an optimal solution might satisfy

$$\frac{d\pi}{d\sigma} = \sum_{i \in \mathcal{A}} \frac{d\pi_i}{d\sigma} = 0 \tag{3.46}$$

or

$$\frac{d\pi_1}{d\sigma} = -\frac{d\pi_2}{d\sigma} \tag{3.47}$$

So, Equation 3.47 has transformed the global objective function $\pi(\sigma)$ which is a soft constraint into a hard constraint. Since it is a constraint concerning two agents, it is an inter-agent hard constraint.

For example, in the Cake-sharing Domain (see Example 3.1), suppose σ is the portion for agent 1, then agent 2 will get a portion of $1 - \sigma$, the sum of their local objectives will be:

$$\pi(\sigma) = h_1(\sigma) + h_2(1 - \sigma) \tag{3.48}$$

suppose also

$$h_i(x) = x^2 \tag{3.49}$$

then from Equation 3.48 and Equation 3.49, we have a common sense principle:

$$\sigma = 1 - \sigma \tag{3.50}$$

The principle says that the two agents' portions should be equal.

3.9.2 The Global Objective Function is the Product of Local Objective Functions

In a CIP situation $< \mathcal{A}, \Sigma, < \pi_1, \pi_2 >>$, suppose there is a global objective π

$$\pi(\sigma) = \prod_{i \in \mathcal{A}} \pi_i(\sigma) \quad (\mathcal{A} = \{1, 2\}) \tag{3.51}$$

Suppose Σ is a continuous domain and $\frac{d\pi_i}{d\sigma}$ exists ($i \in \mathcal{A}$), then an optimal solution might satisfy

$$\frac{d\pi}{d\sigma} = \frac{d\pi_1}{d\sigma}\pi_2(\sigma) + \frac{d\pi_2}{d\sigma}\pi_1(\sigma) = 0 \tag{3.52}$$

or

$$\frac{d\pi_1}{d\sigma}/\pi_1(\sigma) = -\frac{d\pi_2}{d\sigma}/\pi_2(\sigma) \tag{3.53}$$

Equation 3.53 transformed the global objective Equation 3.52 which is a soft constraint into a hard constraint. It is also an inter-agent constraint.

So, in Example 3.1, suppose $h_i(\sigma)$ remains the same as given in Section 3.9.1, then from Equation 3.53 we have:

$$\frac{1}{\sigma} = \frac{1}{1 - \sigma} \tag{3.54}$$

3.10 Characteristics of a Common Sense Principle

It is obvious that to achieve different social goals we need different common sense principles. Different application domains will have different common sense principles. We are not able to give a general common sense principle that can be applied to all application domains, but the following characteristics may be necessary to be exhibited by a common sense principle.

1. A common sense principle is a hard constraint. Since a common sense principle only points out what is a solution or what is not a solution, it is not relaxable at the agent level. This characteristic is required based on the following consideration:

   > In a multi-agent system, each agent has been given a local objective to achieve. It is a soft constraint and it is relaxable at the agent level. A global objective should not be relaxable at the agent level. Otherwise, it seems not necessary from the beginning to let each agent have a local objective, because each local objective should be always subordinated to the global objective. On the other hand, if all agents, when their local solutions are in conflict, are seeking compromise based on the global objective, then the research topic should be classified into the multi-agent coherence coordination problem which concerns how several agents can effectively coordinate to avoid "extraneous" activity and conflicts to performing some collective activity (e.g., [Gasser 84] [Gasser 86] [Bendifallah & Scacchi 87]). Further, a common sense principle in form of hard constraints is analogical to common sense in human negotiation that drives negotiators to agree on an agreement that, from either negotiator's perspective, is in accordance with the common sense. For example, in the Cake-sharing domain, although each agent wishes have a large portion of the cake, if they both have common sense "the elderly should have a larger portion than the junior", then they will agree at least on that the elderly should have more than 50% of the cake. Here, the common sense is given in form of hard constraints rather than in form of soft constraints. In short, common sense principles given in form of hard constraints make it possible that all agents in the whole process of cooperative interaction consistently and continuously stick to their local objective.

2. A common sense principle suggests a social equilibrium. By social equilibrium I mean this principle will imply a strategy and if any agent attempts to move from this strategy it will not increase the social goal achievement.

3. A common sense principle should be efficient. By efficient we mean it can help conflicting agents to reach an agreement as quickly as possible.

4. A common sense principle should be simplistic. By simplistic, we mean the common sense principle involves little computational time and little communication. Although a complex common sense principle may be necessary in some difficult situations, simplicity is always sought if cooperative situations allow.

3.10.1 Social Languages

A common sense principle can be expressed as

$$\mathrm{PRINCIPLE}(L_1(\sigma), L_2(\sigma), ..., L_i(\sigma), ...) \quad (i \in \mathcal{A})$$

we call $L_1(\sigma)$ the social state of agent i if σ becomes an agreement. For example, in two-agent negotiation, if the global objective is the sum of the two agents' local objective, then

$$\left.\begin{array}{l} L_1(\sigma) = \frac{d\pi_1(\sigma)}{d\sigma} \\ L_2(\sigma) = \frac{d\pi_2(\sigma)}{d\sigma} \end{array}\right\} \tag{3.55}$$

and the common sense principle can be:

$$\mathrm{PRINCIPLE}(L_1(\sigma), L_2(\sigma)) = \begin{cases} \mathrm{TRUE} & \textit{if } L_1(\sigma) = L_2(\sigma) \\ \mathrm{FALSE} & \textit{otherwise} \end{cases} \tag{3.56}$$

If the global objective is the product of the two agents' local objective, then

$$\left.\begin{array}{l} L_1(\sigma) = \frac{d\pi_1(\sigma)}{d\sigma}/\pi_1(\sigma) \\ L_2(\sigma) = \frac{d\pi_2(\sigma)}{d\sigma}/\pi_2(\sigma) \end{array}\right\} \tag{3.57}$$

and the common sense principle is in the same form as Equation 3.56.

Note that, here an agent's (agent i) social state is described by $L_i(\sigma)$ instead of $\pi_i(\sigma)$. The reason is very simple, $\pi_i(\sigma)$ which evaluates agent i's state individually is not comparable. This is the same reason as we found in human interactions. For example, when an American buyer and a Canadian seller are bargaining over the price of a product, they would probably offer their price in term of the US dollars instead of the Canadian dollars. Here the US dollar becomes the common language. For this reason, we call $L_i(\sigma)$ the social language. Briefly, when agents exchange proposals in discussion of choosing an agreement, they must describe their own state in social language. Now, when asked how much σ is worth, agent i can say "it is worth $L_i(\sigma)$".

Figure 3.9 illustrates the conflict resolutions in the Cake-sharing domain suggested by common sense principles. (a) (c) show the conflict resolution in a CIP situation where the global objective is the sum of the local objectives and (b) (d) show the conflict resolution in a CIP situation where the global objective is the product of the local objectives.

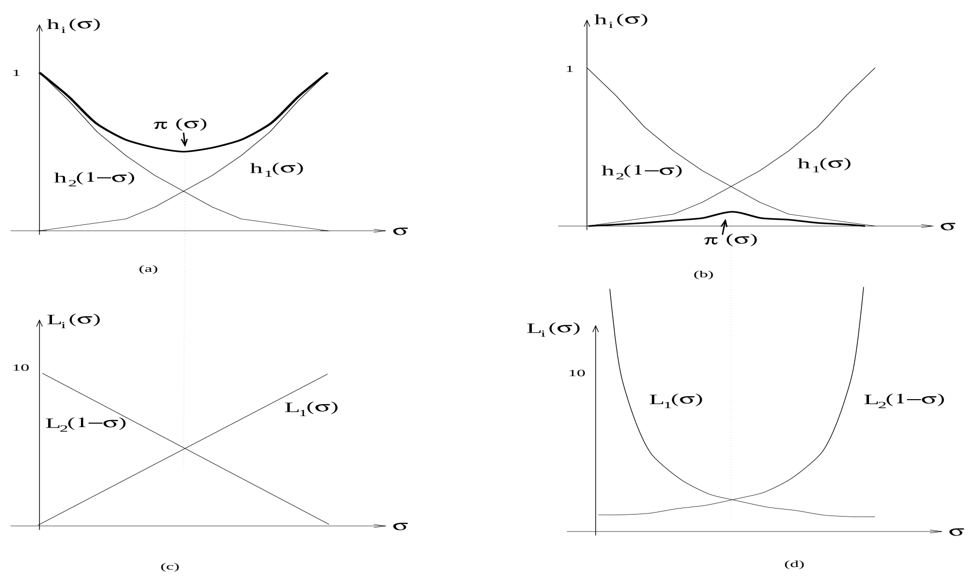


Figure 3.9: An Example of the Common Sense Principle

3.10.2 An Example of Using Common Sense Principle to Solve the Prisoner's Dilemma

As we mentioned before, in Example 3.4, cooperative agents will probably choose (deny, deny) as their joint strategy. This is also the solution suggested by many researchers' work. For example, Rosenschein [Rosenschein & Genesereth 88] derived this solution by assuming that each agent has various assumptions about itself and about the other agent. Axelrod ([Axelrod 80], [Axelrod 87]) shows that the (deny, deny) strategy can be achieved through the use of evolutionary algorithms. Now, we construct a situation of the Prisoner's Dilemma (see Figure 3.10 and Figure 3.11). In this situation, according to Rosenschein's reasoning in [Rosenschein & Genesereth 88], the two agents' cooperative interaction would still converge to the (deny, deny) strategy. His reasoning does not involve any arithmetic calculation. In this situation, there might be a possibility for agents to agree on the (confess, deny) strategy or the (deny, agent 2

		σ_1^2 (deny)	σ_1^2 (confess)
agent 1	σ_1^1 (deny)	$(5, 5)$	$(4.990, 100)$
	σ_2^1 (confess)	$(50, 4.990)$	$(4.999, 4.999)$

Figure 3.10: Payoff Matrix of the Prisoner's Dilemma

pure strategy	Index n	probability p_n	payoff π_1^n	payoff π_2^n
(confess, deny)	1	p_1	50	4.990
(deny, deny)	2	p_2	5	5
(deny, confess)	3	p_3	4.990	100
(confess, confess)	4	p_4	4.999	4.999

Figure 3.11: Another Version of the Prisoner's Dilemma

confess) strategy, since the denying agent only loses a payoff by 0.002 compared with that of his payoff from the (deny, deny) strategy. We can construct a CIP problem $< \mathcal{A}, \Sigma, < \pi_1, \pi_2 >>$ for this situation:

$$\left.\begin{array}{l} \mathcal{A} = \{1, 2\} \\ \Sigma = \{< p_1, p_2, p_3, p_4 > \ \mid 0 \leq p_n \leq 1;\ \sum_i p_n = 1;\ n = 1, 2, 3, 4\} \\ \pi_i(\sigma) = \sum_{n=1}^{4} \pi_i^n p_n \end{array}\right\} \tag{3.58}$$

It can be easily proved that a subset, $\Sigma_{\mathcal{A}}^*$, of Σ, which is given by

$$\Sigma = \{< p_1, 0, p_3, 0 > \ \mid 0 \leq p_1, p_3 \leq 1;\ p_1 + p_3 = 1\} \tag{3.59}$$

satisfies (1) for all strategies in $\Sigma_{\mathcal{A}}^*$, it is preferable or indifferent to any strategy in Σ in both agents' perspectives. We can also prove that for any two different strategies, $< 0, p_1, p_3, 0 >$ and $< 0, p_1^{'}, p_1^{'}, 0 >$ suppose agent 1 and 2's payoffs are respectively $\pi_1, \pi_1^{'}$ and $\pi_2, \pi_2^{'}$, then

$$\pi_1 \succ 1 \ \ \pi_1^{'} \Rightarrow \pi_2 \prec 2 \ \ \pi_2^{'} \tag{3.60}$$

So, $\Sigma_{\mathcal{A}}^*$ is obtained from Σ by removing all irrational strategies, i.e.,

$$\Sigma_{\mathcal{A}}^* = \textbf{remove-irrationals}(\Sigma, \mathcal{A}) \tag{3.61}$$

Actually, the subset construct a line as shown in Figure 3.12. Let us suppose that

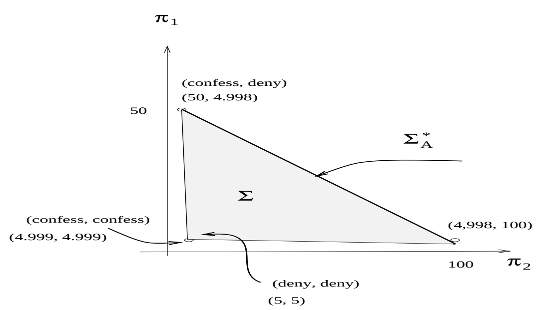


Figure 3.12: Another Version of the Prisoner's Dilemma

there is a global objective which is the product of the two agents' utilities

$$\pi = \pi_1 \pi_2 \tag{3.62}$$

In Equation 3.59, each strategy can also be expressed by

$$\sigma =< 0, \sigma, 1-\sigma, 0 > \quad 0 \leq \sigma 1 \tag{3.63}$$

$$\left.\begin{array}{l} \pi_1 = 50\sigma + 4.998(1-\sigma) \\ p_2 = 4.998\sigma + 100(1-\sigma) \end{array}\right\} \tag{3.64}$$

and we then have:

$$\left.\begin{array}{l} L_1(\sigma) = \frac{d\pi_1}{d\sigma}/\pi_2 = \frac{45.002}{50\sigma+4.998(1-\sigma)} \\ L_2(\sigma) = -\frac{d\pi_2}{d\sigma}/\pi_2 = \frac{95.002}{4.998\sigma+100(1-\sigma)} \end{array}\right\} \tag{3.65}$$

Solving this equation, we have

$$\sigma = 0.7580 \tag{3.66}$$

So, if this global sense, they would probably agree on a strategy:

$$< 0, 0.758, 0.242, 0 > \tag{3.67}$$

That is, they would agree on that they use the (confess, deny) strategy with a probability of 0.758 and the (deny, confess) strategy with a probability of 0.242. In this case, their payoff pair is $(39.110, 27.988)$, much better in both agents' perspectives than that of (deny, deny) strategy which yields a payoff pair (5, 5).

3.11 Further Discussion on the Concept of a Common Sense Principle

3.11.1 The Role of a Common Sense Principle

Our understanding of the role of a common sense principle is that it guides agents' cooperative behaviour in a cooperative interation situation. The common sense principle's role can be likened to a law in a society which guides the members' behavior in the society. The law can not fully control these members' behavior and these members have a certain degree of freedom in how they should behave. Each individual in this society may use any means to achieve what he/she wants to achieve as long as he does not break the law. When disputes happen due to conflicts between individuals' goal achievements, the law will be used to resolve the conflicts. Each individual must obey the law. Each society (e.g., a nation) has a law different from that of other societies. The law reflects the respective goverment's wishes about how its society should be run. Likewise, a common sense principle reflects the system designer's wishes about what should be achieved from the system's performance. The common sense principle can not fully control the behaviour of the agents in the system. They have a certain degree of freedom in their own goal achievments. They can use any means to achieve what they wish. When there is any conflict between agents' goal achievements, the common sense principle could be applied to resolve it.

So, a common sense principle is not exploited to degrade the distributedness of a multi-agent system but to upgrade the performance of the system so that it produces results that are prefered by the system designer especially when the system is a non-open system.

3.11.2 Practical Considerations

The following points should be considered in an application domain:

1. So far, $L_i(\sigma)$ is given in a form of the first order differential of agent i's utility. This might not be practical in some application domains where $\pi_i(\sigma)$ is not explicitly expressed, or where $\frac{\pi_i(\sigma)}{d\sigma}$ does not exist due to σ is not a continuous domain. In this case, a common sense principle may be given in other forms.

For example, a common sense principle may be given in the form of a production rule:

If $L_1(\sigma)$ is ... and $L_2(\sigma)$ is ..., then choose σ as the cooperative strategy.

Here $L_1(\sigma)$ describes agent i's social state which may not be in the form of the first order differential. Whatever, the form of a common sense principle is, it should be endowed with the characteristics we examined in Section 3.10.

2. In some situations, we may not be able to derive a satisfactory common sense principle in the first trial due to the complexity of the application domain. In this case, a common sense principle may be obtained through experiments, through learning processes. We may also borrow some assumptions and axioms that are used for cooperative interactions in open systems.

3. There might exist also situations where a common sense principle suggests several strategies. For example, for a principle,

$$L_1(\sigma) = L_2(\sigma) \quad (\sigma \in \Sigma) \tag{3.68}$$

there might be several strategies $\sigma_1, \sigma_2, ...$ which satisfy this equation. In this case, a further common sense principle is necessary to carry the cooperative interaction forward. We can view a common sense principle as a heuristic in the sense that it pruned the cooperative interaction strategy set gradually, until it consists of only one strategy (which is then the cooperative strategy).

3.12 Summary: Cooperative Interaction Under a Common Sense Principle

In this section, we will summarise the discussion in the last two sections, and formally present a solution for a cooperative interactive problem under a common sense principle. We denote such a CIP problem by

$$< \mathcal{A}, \Sigma, \overbrace{< \pi_1, \pi_2, ... >}^{\pi}, \overbrace{< L_1, L_2, ... >}^{L}, \text{PRINCIPLE}(L_1, L_2, ...) >$$

or by a brief notation

$$< \mathcal{A}, \Sigma, \pi, L, \text{PRINCIPLE} >$$

A solution to this CIP problem is in Σ_c^*:

$$\left.\begin{array}{l}\Sigma_c^* = \{\sigma \mid \sigma \in \Sigma_{\mathcal{A}}^*;\ \mathrm{PRINCIPLE}(\sigma) = \mathrm{TRUE}\} \\ \Sigma_{\mathcal{A}}^* = \textbf{remove-irrationals}(\Sigma, \mathcal{A})\end{array}\right\} \tag{3.69}$$

In other words, the whole cooperative domain Σ will be reduced to $\Sigma_{\mathcal{A}}$ by cooperation between conflicting agents under the rational agent assumption and the benevolent agent assumption and $\Sigma_{\mathcal{A}}$ will be further reduced to Σ_c by cooperation between these conflicting agents under the common sense principle. If Σ_c exists only one member then the member is the agreement between those conflicting agents. If there is more than one member in it, then a further common sense principle must be introduced to let the cooperation continue. That is to say, a cooperative strategy may be found after several common sense principles are called. However, how many common sense principles are needed depends on specific application domains.

Chapter 4

Metaphor-based Negotiation

4.1 Introduction

This chapter focuses on cooperative interactions between agents with incomplete knowledge about other agents. We will present a formal definition of negotiation. We emphasise the distinction between situations in which our definition can be applied, and the human negotiation situation. Due to these distinctions, we call negotiation defined in this thesis *metaphor-based negotiation*. As a complement to this definition, we constructed a computational negotiation model which details the metaphor-based negotiation.

4.2 Revisit CIP Problems Under Fully informed Agent Assumption

A solution to the CIP problem

$$< \mathcal{A}, \Sigma, \pi, L, \text{PRINCIPLE} >$$

is a member of Σ_c^* given by Equation 3.69. For the reason of simplicity, we denote a procedure for obtaining Σ_c^* by:

$$\textbf{best-proposals}(\mathcal{A}, \Sigma, \pi, L, \text{PRINCIPLE}) \tag{4.1}$$

A member of Σ_c^*, according to the discussion in Chapter 3, can be achieved by two steps:

1. The first step is to obtain a subset, $\Sigma^*_{\mathcal{A}}$, of Σ:

$$\Sigma^*_{\mathcal{A}} = \textbf{remove-irrationals}(\Sigma, \mathcal{A}) \tag{4.2}$$

   This subset is achieved based on the rational agent assumption and the benevolent agent assumption. In other words, a solution to the CIP problem must be a strategy that is not irrational for the group $\mathcal{A}$.

2. The second step is to obtain a subset, Σ^*_c, of $\Sigma^*_{\mathcal{A}}$:

$$\Sigma^*_c = \{\sigma \mid \sigma \in \Sigma^*; \mathrm{PRINCIPLE}(L(\sigma)) = \mathrm{TRUE}\} \tag{4.3}$$

   This subset is derived by the common sense principle. In other words, a solution to the CIP problem must be a strategy that does not contradict the common sense principle.

So, it looks as if there were forces driving agents to a cooperative strategy (see Figure 4.1). To obtain such a strategy, agents must be endowed with complete knowledge

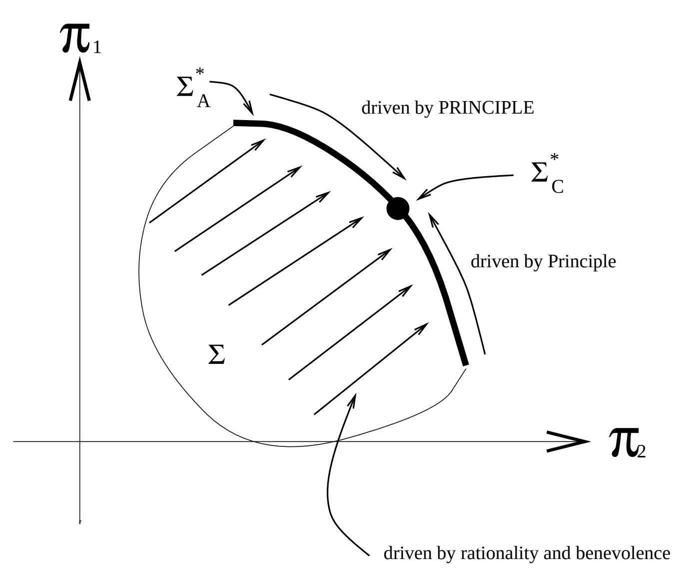


Figure 4.1: The Compromise Driven by Rationality, Benevolence and Principle

about the problem domain, Σ, about their payoff functions $\pi_i(\sigma)$ $(i \in \mathcal{A})$ which represent their individual preferences over Σ, about the common sense principle PRINCIPLE, and about their social states $L_i(\sigma)$ $(i \in \mathcal{A})$. When agents are all assumed fully informed, then they have the complete knowledge about them. Under this assumption, a solution can be worked out by any of the agents. Once it is worked out by an agent, no other agents can possibly object to it. So, communication is not necessary. The discussion in Chapter 3 has two other related implicit assumptions that were not mentioned there, i.e.,

1. Computational time is not considered. There we are concentrating on what knowledge is required by cooperative agents to construct a cooperative strategy. Computational time is not the major focus. This assumption should be removed since computational time is often one of important requirements for a problem solving application.

2. Agents have unlimited capacities of problem-solving. We assume that agents can know Σ prior to cooperation. For example, in the Postman Domain (see Example 3.2), cooperative interaction between two postmen means exchanging letters so that each postman's delivery cost, i.e., the travel distance, can be shorter than in the situation where they do not cooperate. Even if they agree on a common sense principle, e.g.,

$$C(X) - C(X_0) = C(Y) - C(Y_0) \quad (X \cup Y = X_0 \cup Y_0) \tag{4.4}$$

   which says that the two agents' benefits from the cooperation should be equal, there still exists a question of how to calculating $C(X)$ and $C(Y)$ which is actually one of the Travelling salesman Problem [Aarts & Korst 89] which concerns minimising the length of a salesman's tour that starting from and returning his home city after visiting a certain number of other cities. This is known to be a NP-hard problem. However, in Chapter 3, we implicitly assume that agents are able to solve this kind of difficult problem. This is also an assumption not acceptable in real problem-solving situations.

Considered in this way, the cooperative interaction problem under a common sense principle is only solved theoretically.

4.3 Partially Informed Agent Assumption

In application domains, the fully informed agent assumption (Assumption 3.3) must be replaced by a partially informed agent assumptions:

Assumption 4.1 Partial-Informedness *Agents do not have the complete knowledge about the CIP problem-solving.*

We will breakdown this assumption into details in a later section.
The assumption of full-informedness of agents is not acceptable in many situations: agents may not have enough memory for storing all sorts of information; communication is limited due to bandwidth considerations; etc. How can agents with incomplete knowledge cooperatively solve a problem? When agents are partially informed, a proposed cooperative strategy made by one agent according to its incomplete knowledge may be objected to by the other agents from their perspectives that are based on their own incomplete knowledge. When agents are assumed to be rational, such a conflict will be very difficult resolve since no agent will concede to an alternative that could possibly affect its own goal achievement. To get out this difficult situation, more information must be given to agents. For example, in the ideal cooperative interaction given in Figure 3.6, any misunderstanding between the two agents who are willing to cooperate could lead to a strategy that is not (σ_1^1, σ_1^2) which is the result that can be achieved if agents are fully informed. If the two agents can have some ways of exchanging their preferences, then no conflicts could arise during their cooperation.

4.4 Limited Communication between Agents

Communication between agents can help agents to improve their knowledge about each other. However, communication must be limited due to (1) bandwidth limit, (2) computational speed limit, and (3) memory limit. Therefore, we must be very careful about what message these cooperative agents to communicate. Communication between agents serves different purposes in different situations, for instance

- For sharing problem-solving knowledge. For example, if agent 1 has has solved one of his local problems. The result might be communicated to another agent which has a similar problem to solve. For example, in the Distributed Vehicle Monitoring Testbed developed at University of Massachusetts at Amherst [Corkill 82] [Durfee & Lesser 87], experts (Blackboard) communicate data monitored and interpreted by each expert from a set of sensors to construct a global picture of vehicle traffic through an acoustically sensed area. Agents are helping each other in this form of communication.

- For coordinating agents' activities. For example, if two agents are moving a pile of blocks from one place to another, they need to communicate to let each other know which blocks each of them is to move, the time order in which each block is moved. They communicate in order to avoid potential conflict (e.g., the two agents unnecessarily go to move the same block which only needs one agent).

- For exchanging information for resolving conflicts monitored among agents. We are particularly interested in conflicting situations where agents are all rational, benevolent and try to gain something from the communication. In this case, communication may be different from that in the above two situations where agents are helping each other. The content of information that is exchanged between agents through communication must be those that can efficiently help them to concentrate on conflict resolutions. When we assume agents are rational, communication is self-interest-oriented, and much like a process found in human negotiation, so we call it metaphor-based negotiation. We emphasise it is metaphor-based in order to distinguish between it and real human negotiation. A definition of metaphor-based negotiation is given in Section 4.7.

So, negotiation and communication in this context has the same interpretation: exchanging information between rational and benevolent agents in order to resolve their conflicts. In human negotiation, there are many characteristics of negotiators' communicating behaviour such as cheating, threatening, honestness, boasting, etc. Some of these characteristics are necessary to endow to automated cooperative agents, some are not. We will discuss this further in Section 4.7.2.

4.5 Definitions of Negotiation

Although there is a large amount of DAI research on negotiation, as Gasser [Gasser 90] pointed out, negotiation is a term that has been used in literally dozens of different ways in the DAI literature. In [Zlotkin & Rosenschein 91], it is claimed that there does not yet exist a universally accepted definition of what the word negotiation even means. Smith and Davis can be considered as pioneers of introducing the negotiation concept into DAI. Their definition is as follows:

> [Negotiation] [Davis & Smith 83] is a discussion in which the interested parties exchange information and come to an agreement. For our purposes negotiation has three important components: (a) there is a two-way exchange of information, (b) each party to the negotiation evaluates the information from its own perspective, and (c) final agreement is achieved by mutual selection.

Durfee and Lesser's [Durfee & Lesser 87] definition is:

> In general, negotiation is a complex process of improving agreement (reducing inconsistency and uncertainty) on common viewpoints or plans through the structured exchange of relevant information.

We can also find a definition in [Adler *et al.* 89]:

> Negotiation is a process of communication established between two conflicting agents in which they try to develop or refine their plans jointly so that the goals of each are satisfied.

To my knowledge, nearly all negotiation definitions found in DAI are similar to those listed above: they are easily understood, no one could even possibly point out they are wrong. The similar definitions can also be found in politics studies, economics studies, etc. For example, social scientist Brams [Brams 90] defined negotiation as

> By negotiation I mean exchanges between parties designed to reconcile their differences and produce a settlement.

More definitions of negotiation can be found in [Pruitt 81] [Sathi & Fox 89].

The common problem of these definitions of negotiation is that they provide no domain-independent knowledge about how a problem can be solved in a negotiation mechanism, no indication about what is a proposal expressed formally. We may find some implication about the real meanings of negotiation from the computational models built behind these definition, but often, we can find that these models makes us even more confused. For example, in [Davis & Smith 83], negotiation is communication between the managers and the contractors (bidders) to allocate tasks. When a manager announced a task to potentially interested bidders, each of these bidders can show their interest by sending a piece of message to the manager. The manager then select a best bidder. There is only a single round of information exchange. This seems very rare in human negotiation.

4.6 Two Phases of Negotiation

In human negotiation, negotiation is a complex process. There exists no universally acceptable procedure controlling such a process. But, according to our understanding as presented in Chapter 3, the following two phases might be included in a negotiation process:

- **Phase 1**: Negotiating over common sense principle. In human negotiation, negotiators are often situated in a so-called open system where no social goals can be found. When these negotiators try to resolve their conflicts, they need to resolve their dispute based on common sense. These so-called common sense principles are learned from their life-time experiences, or genetically inherited from their ancestors. What is more, the precise form of common sense in use can be argued about during their negotiation. For example, when the two agents are discussing how to share a cake (see Example 3.1), the elder one might say "The elder should have more than the junior, this is common sense", and the younger one might say "Young people usually eat more than old people". Each will bring what they called common sense as a principle for sharing the cake. So, in this sense, common sense is negotiable. So, each agent is trying to convince his negotiating opponent(s) to agree on using a common sense principle that, in his current knowledge, would yield a result in his favour. The convincing ability of a negotiator depends very largely on his knowledge about the world and his personality.

- **Phase 2**: Exchanging information relating to finding an equilibrium in accordance with a common sense principle agreed by all agents at **Phase 1**. Again, the Cake-sharing Domain, if the two agents do not know each others' ages, and they agreed that "The Elder should have more than the junior". Then they will now tell each other their exact ages. In this case, the junior might lie by telling the Senior his age bigger than his true age.

The two phases are exchangeably used in human negotiation, we sometime even cannot distinguish which is which. We, in this dissertation, do not focus our investigation on the first phase. There are several difficulties to let agents to have abilities of negotiating common sense principles. Firstly, automated agents do not have a cultural environment as human negotiators. Human being are living a society where there are certain standards about whom to respect, whom to hate, whom to blame, etc. These standards constrain their behaviour. Automated agents do not have such an environment, hence they do not feel shy, embarrassed, respected, etc. Secondly, automated agents do not have the ability of genetic inheritance. Thirdly, automated agents do not have the ability of re-creating a goal. For example, in human negotiation, for example, in the Cake-sharing Domain, one of the two agents who at the beginning pretends to require a large portion of the cake might intend to have a small portion to please the other. We will avoid this source of confusion in our discussion. However, in future, there might be the possibility of introducing this, more intelligent, phase into automated agent systems. Actually, there are a certain number of researchers in AI field who are interested in how agents can be created to be as intelligent as human beings (e.g., [Bratman 87], [Burmester & Sundermeyer 92], [Jennings 94] etc.). So, we would like to make it clear that although we do not assume agents' ability to negotiate over common sense principles, we do not exclude the possibility of that this ability may be assumed in future.

4.7 Defining Metaphor-based Negotiation

Motivated by the fact that there does not exist a formal definition of negotiation and for the purpose of solving our application problem — AGVSP, we propose a formal definition of metaphor-based negotiation.

4.7.1 Our Definition of Negotiation

Definition 4.1 Metaphor-based Negotiation *is a process of communicating proposals between rational, benevolent and partially informed agents in a cooperative interaction situation*

$$< \mathcal{A}, \Sigma, \pi, L, PRINCIPLE >$$

The process terminates when a proposal by one agent is accepted by all other agents in $\mathcal{A}$ *or when no new proposal could be possibly made by any of these agents.*

1. *A proposal,* p_i*, by an agent* i $(\in \mathcal{A})$ *is a two-tuple:*

   $$p_i = (\sigma_i, l_i)$$

   where σ_i *is a cooperative strategy that is constructed according to agent* i*'s current knowledge, and* l_i *is a value describing its social state when* σ_i *can be accepted.*

   $$\sigma_i \in \Sigma_c^* = \textbf{best-proposals}(\mathcal{A}, \Sigma, \pi, L, PRINCIPLE)$$

   $$l_i = L_i(\sigma_i)$$

   where Σ*,* $\pi \ =< \pi_1, pi_2, ... >$*,* $L \ =< L_1, L_2, ... >$*, and* $PRINCIPLE$ *are all agent* i*'s current belief about the respective pieces of knowledge.*

2. *The condition for any receiving agent, namely agent* j $(\in \bar{i})$*, to accept a cooperative strategy* σ_i *from another agent* i*, is that agent* j *cannot construct a better proposal. More specifically, suppose* σ_j *is the best cooperative strategy for agent* j *to propose according to its current incomplete knowledge, i.e.,*

   $$\sigma_h \in \Sigma_c^* = \textbf{best-proposals}(\mathcal{A}, \Sigma, \pi, L, PRINCIPLE)$$

Note that here Σ, $\pi = <\pi_1, pi_2, ...>$, $L = <L_1, L_2, ...>$, *and PRINCIPLE are all agent j's current belief about the respective pieces of knowledge. Then, if*

$$\sigma_j \preceq j \quad \sigma_i$$

then agent j *must accept* σ_i *instead of proposing* σ_j.

4.7.2 Features of the Definition

Our definition has the following main features:

1. It clearly states the problem domain, i.e., the CIP problem, to which the negotiation definition fits. As we have mentioned in the last chapter, many DAI problems, such as the task allocation problem, resource allocation problem, DCSP problem, etc., can be transformed into a CIP problem, our definition can be taken as application domain-independent. As an application, we can see how the AGVSP problem is solved by metaphor-based negotiation mechanism in the later chapters.

2. It gives formal representation of a proposal.

3. A proposal consists a proposed cooperative strategy σ_i and associated information l_i which represents the social state of the proposing agent. In other words, when an agent puts forward a proposed cooperative strategy, it also tells the opponents why the strategy is good by showing its social state under this cooperative strategy. A proposal in most negotiation models found in DAI means only a cooperative strategy.

4.7.3 Why Negotiation is Metaphor-based?

As we have emphasised before, our negotiation strategy is metaphor-based which means that the negotiation is not aimed at modelling human negotiation, but at applying our knowledge about human negotiation for solving real problems. The following reasons will further support our statement.

1. As we mentioned before, we do not assume agents' ability of negotiating over common sense principles which is often negotiable in human negotiation.

The system designer must download knowledge about global objectives to agents that are created by him. In our definition, PRINCIPLE is this type of knowledge that is given to agents at initial times. Common sense principle are not negotiable in our definition.

2. We assume that agents are benevolent. In human negotiation it is not necessary to assume negotiators are benevolent. Cheating, threatening, hesitation, malevolence, etc., are often useful characteristics to bring a dispute to a settlement. These methods are not acceptable in our definition. Although characteristics, such as cheating, may benefit the cheating agent on some occasion in open systems [Zlotkin & Rosensein 96], we think they are not beneficial in non-open systems. They only make an agent to spend longer time to process information received from other agents and, as a consequence, degrade the quality of performance of the whole system.

3. Agents in our definition are created by the system designer for subproblem solvings. Each agent is given an objective for solving its subproblem. It is assumed that no attempt is made to alter the objective unless the system designer tell it to do so. However, in human negotiation, negotiators are able to change their goals during negotiation.

4.8 A Computational Negotiation Model

According to Definition 4.1, we develop a computational negotiation model. This model has two-fold meanings. On one hand, it can be taken as a complement to Definition 4.1. Many points such as communication procedure, updating knowledge, conditions for agents to propose, etc., that are not clearly stated in the definition, are clarified in this model. On the other hand, this model provides a framework for domain applications. The computational model is given in Algorithm 4.1 — Algorithm 4.5.

Algorithm 4.1
procedure negotiation-main()
localvars: $p_j, p_k, i, j, k, \mathcal{A}$;

(1) **initialising-agent**(j) $(j \in \mathcal{A})$; *;; see Algorithm 4.2.*

(2) $i = 1$; *;; agent 1 is the first proposer*

(3) $\bar{i} \Leftarrow \{j \mid j \in \mathcal{A};\ j \neq i\}$ *;; a set of receivers*

(4) $p_i \leftarrow$ **making-proposal**(i)

(5) **communication**(i, j, p_i); $(j \in \bar{i})$ *;; see Algorithm 4.4.*

(6) $p_j \leftarrow$ **making-proposal**(j); $(j \in \bar{i})$ *;; see Algorithm 4.3.*

(7) **if** $\forall j (\in \bar{i})$ **car**$(p_j) =$ **car**(p_i) **then exit return car**(p_i) **endif**

(8) **if** $\forall j (\in \bar{i})$**car**$(p_j) = \phi$ **then exit return** ϕ **endif**

(9) **Select an agent** k $(\in \bar{i})$ **with** $p_k \neq \phi$;

(10) $i \Leftarrow k$ *;; agent k becomes the proposer.*

(11) $\bar{i} \Leftarrow \{j \mid j \in \mathcal{A};\ j \neq i\}$ *;; the rest become the receivers.*

(12) **goto 3**

endprocedure

Algorithm 4.2

procedure initialising-agent(i)

agentvars: $\Sigma, \Sigma_p, p_j, p_i$;

functionvars: $L_j(\sigma)$, $\pi_j(\sigma)$, *PRINCIPLE*(L);

(1) **initialising** Σ, $L_j(\sigma)$, $\pi_j(\sigma)$ *PRINCIPLE*(L)
according to domain specification

(2) $p_j \leftarrow \phi$ $(j \in \mathcal{A})$

(3) $\Sigma_p = \phi$;

endprocedure

Algorithm 4.3

procedure making-proposal(i)

localvars: $j, k, \sigma_i, \sigma_k, \Sigma_c^*, \Sigma_c^{**}, p$;

agentvars: p_j, p_k, Σ_p;

functionvars: $L_i(\sigma)$;

(1) **if** $\Sigma_p = \phi$ **then goto 5 endif**;

(2) **Choose an agent, namely agent** k, **with** $p_k \neq \phi$

(3) $\sigma_k \leftarrow$ **car**(p_k)

(4) **updating-knowledge**(i, k, p_k)

(5) $\Sigma_c^* \leftarrow$ **best-proposals**(i)

(6) $\Sigma_c^{**} \leftarrow \{\sigma \mid \sigma \in \Sigma_c^*; \sigma \notin \Sigma_p\}$

(7) **if** $\Sigma_c^{**} = \phi$ **then** $p = \phi$ **goto 13 endif**

(8) **Select an strategy, namely** σ_i, **from** Σ_c^{**};

(9) $l_i \leftarrow L_i(\sigma_i)$;

(10) **if** $\Sigma_p = \phi$ **then** $p = (\sigma_i, l_i)$ **goto 13 endif**

(11 **if** $\sigma_i \succ i\ \ \sigma_k$ **then** $p = (\sigma_i, l_i)$ **goto 13 endif**

(12) $p = (\sigma_k, \phi)$;

(13) **return** p

endprocedure

Algorithm 4.4
procedure **communication**(i, j, p)
agentvars: p_i;
$p_i \leftarrow p$
endprocedure

Algorithm 4.5
procedure **updating-knowlege**(i, j, p)
agentvars: Σ
functionvars: $L_k, \pi_k \ (k \in \mathcal{A})$
(1) updating Σ
(2) updating $L_k, \pi_k \ (k \in \mathcal{A})$
endprocedure

4.8.1 Some Notations

In the above two algorithms, there are several points to make:

1. **localvars** declares variables that only have effect in the procedure where they are declared.

2. **agentvars** declares variables that belong to the respective agents. We can view them as objects in object-oriented programming [Page & Thomas 89]. They can be updated at any time and they can be used in only by the agent for which the variables are declared. Note that these agent variables in principle can only be assigned a value by the respective agent.

3. **functionvars** declares variables whose values are functions (or procedures). **agents variables** and **functionvars** in regard to an agent are the agent's belief about the knowledge represented by these variables or functions. This knowledge can be modified when new information is received.

4. p, p_j, p_k, p_i represents proposals in form of (σ, l).

5. σ_i, σ_k are strategies.

6. Σ_p is a set of strategies that have been proposed by agent i before.

7. Σ_c^* has the same interpretation as in Chapter 3.

8. Σ_c^{**} consists of strategies in Σ_c^* that are not in Σ_p (i.e., that have not been considered before).

9. $L_i(\sigma), \pi_i(\sigma)$, PRINCIPLE are procedures and their interpretations are the same as in Chapter 3.

10. i, j, k represent agents.

4.8.2 Comments on the Algorithms

Algorithm 4.1 is a procedure responsible for initialising a negotiation, coordinating communication between agents, and terminating the negotiation procedure. Initialising a negotiation means downloading agents with initial knowledge about the problem domain, about objectives, about their initial beliefs about themselves as well about others. Communication is controlled by the procedure based on the principle that every time there is only one proposer and the rest of the agents become the receivers that could potentially become candidates for proposers in future. In this algorithm, **initialising-agent**(i) (see Algorithm 4.2) is a procedure that assigns initial values to agent i's agent variables and function variables. The procedure **communication**(i, j, p_i) (see Algorithm 4.4) is a process in which the sender sends out the proposal, and the receiver receives the proposal, and as a result agent j's agent variable, P_j is reassigned a value p_i. They are very straightforward, therefore we did not list them here. The negotiation procedure terminates according to the following two conditions:

- If all receivers respond to a proposer's proposal σ_i with the same strategy then the procedure terminates and returns σ_i as the agreement (see step (7)).

- If, after a proposer proposes a proposal, no receiver responds, then the procedure terminates and returns no agreement (ϕ) (see Step (8)). We can view no agreement as a special agreement.

Algorithm 4.3 is a procedure that each agent uses to reason about the world. However, each agent uses its own knowledge in its reasoning activities. When an agent receives a proposal from another agent, its reasoning procedure can be divided into three stages (1) updating current knowledge; (2) creating a set of proposal candidates, and (3) deciding whether to accept the proposal from that agent, or waiting for new knowledge coming, or propose a new proposal. The third stage can further be detailed as:

- If it can not find a proposal better than that just received, than it will accept that proposal just received (see step (10)).
- If it can not find any proposal (i.e., $\Sigma_c^* = \phi$), then it will make no proposal ($p = \phi$) (see step (7)).
- Otherwise, it will make a proposal that has never been made before (i.e., the proposed strategy σ_i is in Σ_c^{**}).

We may add more rules of accepting a proposal between step (11) and (12) in Algorithm 4.3. For example, we can add a no regret rule:

$$\textbf{if } \forall \sigma \in \Sigma_p \sigma_k \succeq i \;\; \sigma \textbf{ then } \text{PROPOSAL} = (\sigma_k) \textbf{ endif}$$

which states that *agent i must accept a proposal σ_k by another agent k, if it is preferable or indifferent to any proposal (in Σ_p) agent i has made before.* Adding such a rule could possibly speed up the negotiation process. We leave this room for the application domain system designer.

We also left a procedure **updating-knowledge**(i, j, p_j) undefined. What knowledge should be updated and how to update them largely depend on the application domain the model is applied to. So, we think it is better to leave it open. For example, in the cake-sharing example, agent i will have some incomplete knowledge about the procedure $L_j(\sigma)$ of agent j. After communication, agent i received some proposals (σ_j^1, l_j^1), (σ_j^2, l_j^1), ... Based on these knowledge, an approximate procedure can be worked out by polynomial approximate approaches. The more the proposals received from agent j, the closer the approximate procedure will be. Sometimes, $L_j(\sigma)$ could be linear function. In this case, from a mathematical point of view, two proposals from agent j are enough for agent i to precisely predict $L_j(\sigma)$. We will use this linear approximation method in the AGVSP negotiation model we developed (see Chapter 7).

4.9 Summary

In this chapter, we defined the term metaphor-based negotiation which tells us what is a proposal, and what is the condition for an agent to accept a proposal made by another agent. The definition is given in a more formal way than any negotiation definition found in the literature to date. Our computational negotiation model based on this definition describes the points that have not been explicitly expressed in the definition. In other words, the computational model is a complement to the definition.

Chapter 5

Introduction to Kwa's Iterative Negotiation Model

In this chapter, I shall introduce Kwa's Iterative Negotiation Model [Kwa 88a], [Kwa 88b] that was designed to resolve resource conflicts in the AGVSP. His work in this direction motivated us to take the subject of negotiation as the theme of this dissertation to investigate further some major issues relating to cooperative interaction problem solving that have not been solved in Kwa's model and other negotiation models found in the DAI literature. We first overview Kwa's work on AGV movement planning briefly (see Section 5.1), and then introduce his work on AGVSP in detail (see Section 5.2). For the purpose of this dissertation, we reimplemented his negotiation model. An outline of the reimplementation is given (see Section 5.3). After this, we raise some important unsolved issues in Kwa's model (see Section 5.4).

5.1 Overview Kwa's Work on AGV Movement Planning

As we mentioned in Chapter 1, AGV movement planning involves task assignment, route planning, and movement timings of the AGVs. Kwa [Kwa 88a], [Kwa 88b] investigated these subjects at some depth.

5.1.1 Optimal Task Assignment

The task assignment problem is to determine how best to match a set of given tasks to the available AGVs such that each AGV is assigned not more than one task. Optimal task assignment means minimising the total distance which must be traversed to achieve the given tasks. By minimising the travel distance, the running cost for an AGV-based manufacture can be cut. The optimal assignment thus meets an economical objective. Kwa developed a novel implementation of a global assignment algorithm — the Hungarian method — which has been found to be more efficient than the conventional implementation.

5.1.2 Optimal Route Planning

Also, in order to satisfy the objective of economy, the same objective also requires that every AGV minimise the route taken to achieve its assigned task. Basically, the shortest path problem is the objective of optimal route planning. This is a searching problem, one of the most important issues concerning search is searching efficiency or computational speed. By examining existing searching approaches, especially the bidirectional heuristic search (BS) and the A^* searching algorithm, Kwa developed an improved BS algorithm — BS^* — which exploited all opportunities to achieve early termination of search and information available during search to eliminate unpromising nodes. Therefore the searching speed is greatly increased.

5.1.3 Tolerant Planning and Negotiation for AGVSP

Kwa argued that planning systems need to consider the difficulties of successful execution to be useful in solving real world problems. A plan that is only logically correct will be likely to fail due to possibility that the states of the world could change unexpectedly according to our model. Therefore, plans must have a degree of executability — the ability to cope with real world issues during execution. Also, when a plan fails, it often needs to be repaired during execution. Therefore dynamic replanning may be necessary. However, dynamic replanning can be expensive or is liable to be futile if it fails to complete in time. Hence it is desirable to minimise or defer dynamic replanning as much as possible. Tolerant planning, which allows an activity to have more resources than it logically needs, can make plans more robust, i.e. permitting the goal to be achieved in spite of events which work against its successful execution. Therefore tolerant planning increases the executability and at the same time reduces the need for frequent replanning.

However, tolerant planning raises another problem, i.e., resource conflict. Resources are limited, allowing redundant resource allocation means more than logically needed resources are requested. So, the chance of resource conflict during tolerant planning is higher than that during non-tolerant planning. Kwa developed an Iterative Negotiation Model for resource conflict resolution. The idea and algorithm of the model are introduced in the next section.

5.2 Iterative Negotiation Model for Resource Conflict Resolution

5.2.1 Point-based vs Interval-based

In the case of the AGVSP, tolerant planning means interval-based scheduling instead of point-based scheduling. A point-based schedule defines exactly the time instant when an AGV will arrive at a position (Figure 5.1). It shows that an AGV departing from A at time t_a, moving at constant speed towards B, will arrive B at time t_b. It remains at B until time $t_{b'}$ (perhaps to pick up a load) before proceeding to C, arriving there at t_c. An interval-based schedule defines the range of time when an AGV is given sole

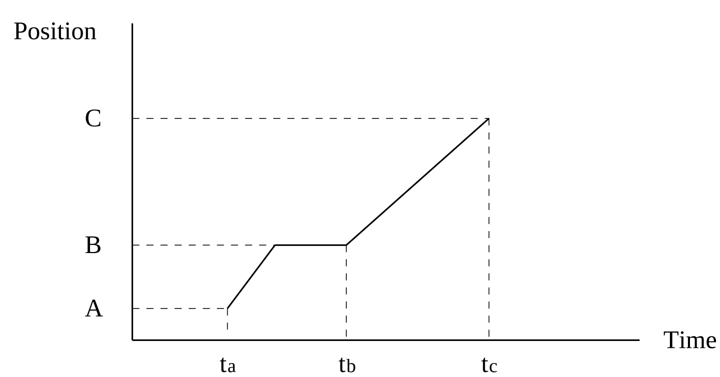


Figure 5.1: Graph of a Point-based Schedule

access to a position (Figure 5.2). The interval widths may vary: since an AGV is likely to be late when a greater distance has to be traversed, the interval width should be monotonically increasing with respect to distance travelled. As shown in Figure 5.3,

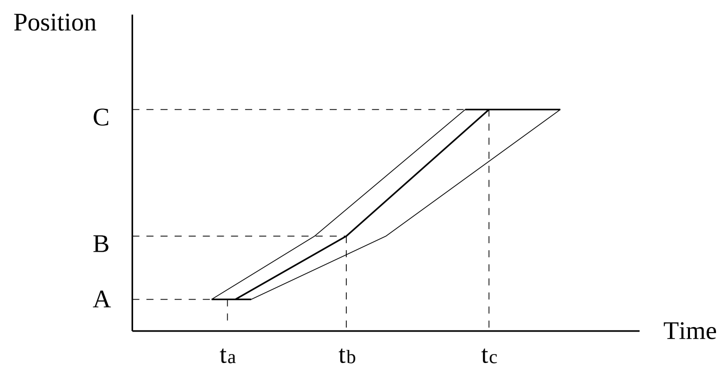


Figure 5.2: Graph of an Interval-based Schedule

an interval usually comprises *front hedge*, *planned arrival time*, *waiting time*, *tolerance* and *rear hedge*. In a point-based scheduling scheme, an interval means the waiting time only. So, the rest of the intervals listed above are all tolerance intervals for different purposes. The front hedge and the rear hedge are two constant time-spaces in distinguishing with the tolerance time space which varies according to travel distance. Externally, point-based representation and interval-based representation of an resource allocation are the same: each activity is assigned a piece of resource in terms of time-space. However, they are different internally. In point-based representation, each activity is assigned a fixed amount of resource which can be expressed by two variables, i.e., starting-time and station of the time-space. In interval-based representation, each activity is assigned some resource that could be shifted, reduced afterwards. So, the resource must be expressed by at least three variables, i.e., starting-time, length, and station of the time-space. Basically, interval-based representation requires more memory space.

5.2.2 Tolerant planning aggravates resource conflicts

From Figure 5.3, we can obviously see that an interval-based schedule needs more time-space resource than point-based schedules do, therefore eliminating chances for other AGVs being assigned the resource used as redundant resource for the interval-based schedule that could otherwise be used for these AGVs. Resources are often limited. The more resources reserved by one activity, the less they will be available for other

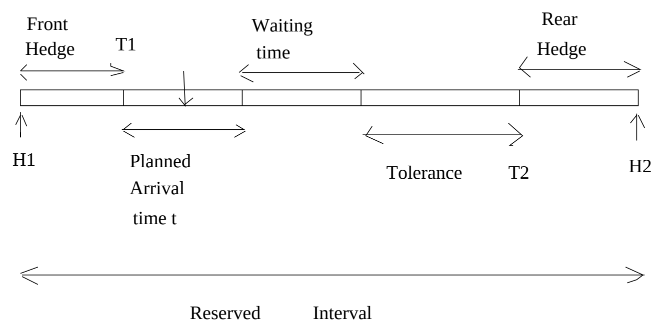


Figure 5.3: Component of a Reserved Interval

activities. Therefore there should be balance between ease of plan generation and plan robustness.

5.2.3 Solving AGVSP in a Multiagent Environment

Without much arguing, Kwa suggested that an AGVSP problem can be solved in a multiagent environment, taking each AGV as an agent responsible for assigning a local schedule for itself. In this case, resource conflicts can be viewed as conflicts between agents. Initially each agent constructs an interval-based schedule without considering other agents' schedules. Then the agent checks whether this schedule has been assigned any interval that overlaps with other agents' assignments. If there is overlapping interval, then we say a conflict has occured.

5.2.4 Negotiation as a mechanism for Conflict Resolution

There are various possible approaches to resolving a conflict. Kwa's negotiation mechanism is one of them. During negotiation, conflicting agents interchangeably concede their initial demands (which are certainly the best assignments for their respective schedules) by shifting their intervals or by shrinking (yielding) their intervals (see Figure 5.4 and Figure 5.5).

5.2.5 Iterative Negotiation Model

A computational negotiation model is proposed by Kwa (see Algorithms A.1, A.2, A.3 and A.4). As claimed by Kwa, this model mimics many characteristics of negotiation,

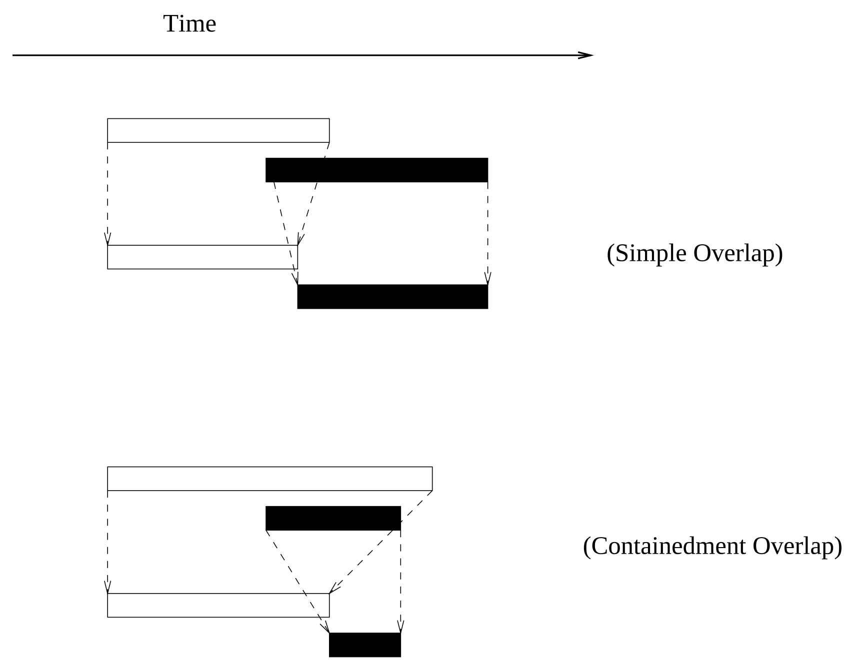


Figure 5.4: Conflict Resolved by Yielding

some of which are described below:

- *Conflict knowledge.* An agent knows which other agent(s) it is in conflict with.
- *Willingness to negotiate.* Between any two agents in dipute, at least one must be willing to attempt to resolve the conflict. In the absence of capital gains, it is assumed that this motivation comes from altruism or benevolence imposed by the system designer.
- *Knowing the negotiable.* An agent knows its own bottom line of negotiation, although not necessarily that of other agents.
- *Agents are selfish.* An agent will minimise what it gives up to settle the conflict. Each would hope that the gap of dispute can be narrowed by the other party's effort rather than its own. In the context of multi-agent planning, the initial tolerant plan of each agent is an "optimal" tolerant plan in the sense that any yielding reduces its robustness and hence executability. Thus every agent should want to practise the policy of being selfish, i.e., sticking as close as possible to its initial plan.
- *Negotiation is an iterative process*

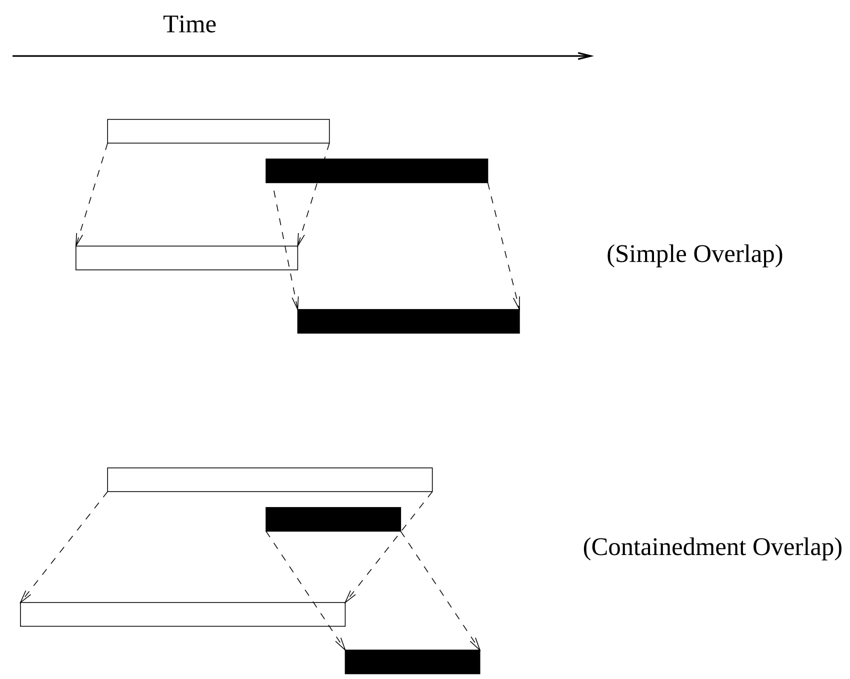


Figure 5.5: Conflict Resolved by Shifting

- *Skillful negotiation.* Every agent has a set of negotiation techniques to use.

- *Nested negotiation.* During negotiation between two agents, the concession by one of the two agents by shifting its initially allocated resource may cause a further conflict with a third party. In this case, the new conflict must be resolved first through negotiation between that conceding agent and the third party.

5.2.6 Defining Negotiation Techniques and Skills

Kwa then defined various negotiation techniques each of which determines the way of conceding. Techniques includes ShiftLeft, ShiftRight, YieldFrontHedge, YieldEnd-Hedge, YieldFrontHedge&Shift, YieldEndHedge&Shift, YieldTolerance, YieldTol&Shift, MoveLeft&Below and MoveRight&Above. The meanings of most of these techniques are very plain, as we bear in mind that shifting means shifting an interval without reducing the width of it and yielding means reducing the width of an interval without shifting it. For example, ShiftLeft is a technique for reducing the overlap on its right by shifting the whole interval to the left without reducing its width. MoveLeft&Below and MoveRight&Above will be further explained as follows:
MoveLeft&Below is a technique for reducing the right overlap by shifting to the left by

the required amount after intervals to the left and below have modified their tolerances and/or hedges to allow such a shift. Likewise, MoveRight&Above is a technique for reducing the left overlap by shifting to the right by the required amount after intervals to the right and above have modified their tolerances and/or hedges to allow such a shift.

Note that in a negotiation, some of techniques can not be used at the same time. For example, ShiftRight and ShiftLeft are two techniques that move an interval into two different direction, so they are not compatible and therefore can not be used at the same time.
A negotiation skill is simply a set of negotiation techniques that are compatible. For example,

(ShiftLeft YieldEndHedge YieldFrontHedge&Shift YieldTolerance MoveLeft&Below)

is a set of compatible techniques suitable for an agent that has an initially allocated interval (which Kwa called *RToken*) in conflict with another Rtoken on its right reserved by another agent. Several such skills have been defined by Kwa.

5.2.7 Implementation

Kwa implemented his iterative negotiation algorithm using the objected-oriented language Loops [Stefik & Bobrow 86]. Here is a simple example in his implementation to show how conflicts can be resolved using the iterative negotiation model. The following example is cited from [Kwa 88a] to show how does his model work (in Chapter 6, we will give an example to show how does our spring-based negotiation model work when the scenario is the same as one in this example).

Example 5.1 *Suppose there are two agents X and Y. X will travel along a path (see Figure 5.6):*

$$(x_9, x_{10}, x_{11})$$

and initially it reserves three Rtokens R_1, R_2 and R_3 at the respective stations, and Y *will travel along a path*

$$(x_{12}, x_{10}, x_{13})$$

and it reserves three Rtokens R_6, R_7 *and* R_8 *at the respective stations.*

R_1*: [198.75, 258.75]*

R_2*: [217.50, 289.01]*

R_3*: [248.33, 311.33]*

R_6*: [240.42, 301.92]*

R_7*: [259.17, 332.17]*

R_8*: [290.00, 364.50]*

Since R_2 *and* R_7 *are two RTokens at the same station and they overlaps with each other, therefore they are in conflict. This conflict can be resolved by Kwa's implemented model in a way illustrated as below:*

R_2 *and* R_7 *are in conflict. Amount (amt) of conflict is 29.83.*

R_7*: ShiftRight by 4.17, i.e. [259.17, 332.17]* $\rightarrow$ *[263.33, 336.33]*

Residue: 25.67

R_2*: ShiftLeft by 2.08, i.e., [217.50, 289.01]* $\rightarrow$ *[215.42, 286.92]*

Residue: 23.58

R_7*: YieldFrontHedge by 10, i.e., [263.33, 336.33]* $\rightarrow$ *[273.33, 336.33]*

Residue: 13.58

R_2 *YieldEndHedge by 10, i.e., [215.42, 286.92]* $\rightarrow$ *[215.42, 276.92]*

Residue: 3.58

R_7*: MoveRight&Above: (* R_8 *will shift right first before* R_7 *can be shifted right).*

ShiftRight by 3.58, i.e., [273.33, 336.33] $\rightarrow$ *[276.92, 336.33]*

Residue: 0

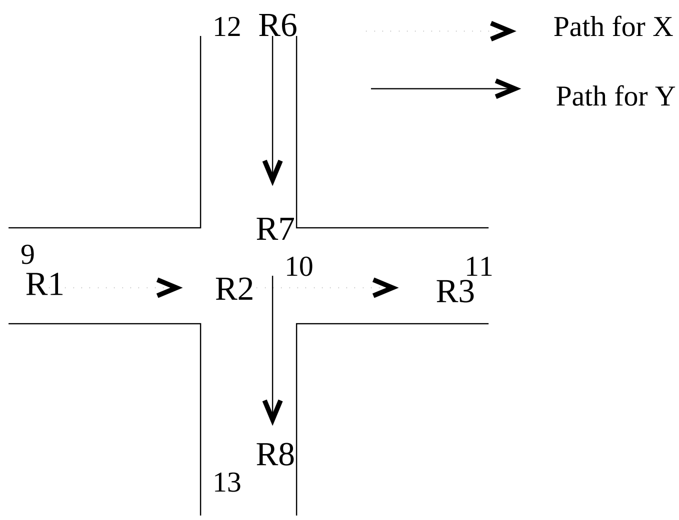


Figure 5.6: Disposition of RTokens

5.3 Our Reimplementation of Kwa's Iterative Negotiation Model

We reimplemented Kwa's iterative negotiation model for the purposes of (1) examining closely how various negotiation skills work, and (2) preparing some initial work for further investigation — the implementation provides some fundamental work for implementation of our Spring-based negotiation model. To reimplement the model, we must consider the following requirements that have to be met by the negotiation programme :

- The *ability* to find a resolution for the conflict. By ability is meant that a resolution can be found if any exists. For a conflict there may exist several resolutions, but it is of interest whether the negotiation process can find one of them.

- The *minimum influence* on other agents' plans. If two agents are in conflict, it is better to resolve the conflict between them only without involving a third agent for yielding resources. Otherwise there are problems which might result from the chain reaction of nested negotiation. This does not mean that calling a third agent is strictly prohibited. Over-emphasizing the spirit of altruism will result in another problem of unbalanced overall robustness of the multi-agent system, that is, the agents involved in the negotiation will severely compromise their robustness for the sake of not asking a third agent to yield its resources.

- The *fast speed* of completing the negotiation. The speed must be fast enough because negotiation is a dynamic procedure from the viewpoint of the whole AGV planning system. If the speed is slow the negotiation will not serve the purpose of overcoming the problem of frequent replanning.

To decrease the influence on other agents' plans, the negotiation skills can be combined by using yield skills first and using shift skills after that. The fastest speed of completing the whole negotiation process depends on several factors:

- The *search strategy* we choose.
- The *computer programming language* we use.
- The *design* of the negotiation program.

In this chapter we implement Kwa's interactive negotiation model, aiming to preserve all the necessary information during negotiation so that it can be utilized to perform backtracking in future work. We found that how the negotiation proceeds depends largely on what negotiation techniques are used and how much resource is yielded in every round. In our implementation, these two problems are solved by assuming that a negotiation technique is chosen by selecting the first technique in a prioritized technique list, and that the concession amount in every technique is a fixed number which is given prior to the negotiation process. However, by doing so, the characteristics of negotiation could not be sufficiently expressed. In the real case of negotiation, negotiation techniques and concession amount should be chosen dynamically according to the updated information the negotiator can obtain.
The detail of our implementation is given in Appendix B

5.4 Discussion on Kwa's Model

During our reimplementation, the most difficult thing that we met is to choose appropriate negotiation skills. According to Kwa's argument, these skills have to reflect the characteristics of negotiating agents (e.g., selfishness, altruism, etc.). We believe the characteristics presented by Kwa are right according to understanding about the term negotiation. However, Kwa did not model these characteristics into agents' knowledge. For example, if an agent is selfish, then it will concede as little resource to others as possible. However, what does "as little as possible" mean? Basically, we need a kind of model that can tell how much the agent should concede in a formal way. As a complete account for issues that have not been properly addressed, we listed them as follows:

- **what to communicate?**

  Negotiation in Kwa's model means communicating proposals and a proposal means a resolution to the conflict. So, during negotiation, agent will argue in this way:

  agent A I want the conflict to be resolved in this way.

  agent B No, I do not agree. I want to the conflict to be resolved in that way.

  ....

  In other words, negotiators do not present reasons why they want the conflict resolution to resolved in their proposed way. This seems not the general way in human negotiation. In human negotiation, negotiators often present resolutions associated with supporting arguments.

- **To what degree are agents autonomous?**

  Agents' autonomy is very limited. All negotiation skills are set for each agent prior to negotiation. Choosing a negotiation strategy here means simply selecting the first technique in a skill set. So, the strategies are not reasoned at agent level, but by the system designer. This will bring a serious consequence, i.e., agents cannot deal with situations during negotiation that have not been expected.

- **Are there any global objectives?**

  It is obvious to us that the answer to this question is "yes". We need to let agents have a sense of the global problem solving. However, this issue has not been discussed in Kwa's model. To let agents have the sense of the global problem solving does not mean to let agents to be controlled by a global controller (e.g., a manager) but to let agents seek compromises according to wishes of the system designer. We have clarified in Chapter 3 that to set a common sense principle (i.e., to let agent to have the sense of the global objective) is not for degrading distributedness of a multiagent system but for upgrading individuals' performance. We mean, in this sense, Kwa's model has to be improved by downloading to agents some knowledge of the global objective. These agents are created by the system designer and their knowledge is given by the system designer (e.g., the conceding techniques and skills are created by the system designer), therefore, their knowledge should reflect the designer's wishes.

- **What is the belief of an agent about other agents?**

  Negotiating agents are situated in a multiagent environment. Each agent is an expert for its own local problem solving. It can be assumed that (in Kwa's model) each agent knows its own local scheduling perfectly. However, it is not assumed to be fully informed about other agents' local problem solvings. We believe that a good negotiating strategy for an agent should be reasoned by it based on its own problem solving as well as problem solving of agents involved in the negotiation.

- **What does it means that "an agent is selfish"?**

  Terms like "selfishness", "benevolence", "altruism" are well understood by us in human activities. We can literally interpret them. However, we can not model them if we do not have a formal definition of them. So, bringing these terms into a scientific and technical discussion without being formally defined will only make the discussion more ambiguous and confusing. Although Kwa claimed that his model can display these characteristics of a negotiating agent, we did not find any clue from it.

5.5 Summary

In this chapter, we introduced Kwa's work on AGV movement planning, especially his work on AGVSP. We detailed his iterative negotiation model and illustrated an example of how conflicting agents cooperatively resolve a conflict by iterative concession. We briefly introduced our reimplementation which is designed as a building environment for further investigation into the AGVSP. Finally, we listed several important issues that have not been addressed or solved in Kwa's model.

Chapter 6

The Spring Model for Local AGV Scheduling

6.1 Introduction

This chapter details the domain specification of the AGVSP and proposes the idea of the spring model for AGV local scheduling. We view the AGVSP as a *Distributed Constraint Satisfaction Problem* (DCSP) which consists of several small *Constraint Satisfaction Problems* (CSP) problems each of which is called a local AGV scheduling problem, which concern the construction of a local schedule for an AGV. This domain specification will specify intra-agent constraints as relations between variables relating to each local schedule and inter-agent constraints as relations between variables relating two local schedules (see Section 6.3). Constraints can be divided into hard constraints which define the solution space for a CSP (or DCSP) problem and soft constraints which define the quality of a solution. Realising that soft constraints are relaxable during problem solving, we propose a spring-model for representing a local AGV schedule: soft constraints are represented by springs, and hard constraints are represented by other elements such as cylinders, walls, etc. The model is a representational model which has two functions: (1) it suggests a local AGV schedule given a limited amount of resources; (2) it suggests negotiation techniques when a local schedule is in conflict with others.

6.2 Prerequisites and Objectives

6.2.1 Prerequisites

The investigation which is described in the rest of the dissertation is carried out based on our belief that the following viewpoints presented by Kwa about solving the AGVSP problem are acceptable:

- Tolerant planning makes plans robust. So we will assume that agents' activities will be assigned redundant resources on top of their logical demands.
- Tolerant planning increases the possibility of resource contention and therefore conflict resolution is required. Multiagent approaches to conflict resolution are suitable for an AGVSP problem. So, we will view the AGVSP problem as a distributed problem consisting of several *local AGV scheduling problems* each of which is solved by a respective agent.
- Negotiation is a suitable mechanism for conflict resolution. We accept the main stream of Kwa's iterative negotiation model which is featured by (1) a global AGV schedule is built in an incremental way, (2) each time a new local schedule is built by an agent, the agent will make sure it is not in conflict with those having been scheduled before, and (3) negotiation between the agent and other agents with which it is in conflict is initialised by the agent.

6.2.2 Objectives

We have pointed out in Chapter 5 that there are several issues that were not addressed in Kwa's iterative model. Some of the issues are related to general DAI research on negotiation models and some of them are related only to the AGVSP. We have closely examined general negotiation issues in Chapter 3 and Chapter 4. After investigating cooperation between rational, benevolent agents, we proposed the idea of a common sense principle to solve a CIP problem in which one agent's gain means definitely at least one of the other agents' loss. We have defined formally the term of (metaphor-based) negotiation and proposed a computational negotiation model. As a specific application domain, we, motivated by some unaddressed important issues in Kwa's model, will develop a spring-based model for representing a negotiating agent.

6.3 Domain Specification for the AGVSP

6.3.1 Layout of an AGV-based Factory

The layout of an AGV-based factory provides the general geometrical information about the configuration of the area where AGVs move about, such as the location of stations, junctions, the distance from one place to another, etc. The layout set Γ contains the data we are concerned with

$$\Gamma =< \mathcal{X}, \Omega, \mathcal{A} >$$

- $\mathcal{X}$ is the station set consisting all stations in the factory layout. A station here is defined as a place on a path where loading/unloading is carried out or a junction.
- Ω is the arc set. Each member, arc, of Ω is a list described as follows:

  $$arc = ((x_l, x_k)\ v_{lk}^{max}\ d_{lk}\ t_{lk}^{min})\quad x_l, x_k \in \mathcal{X}$$

  where, (x_l, x_k) represents a traversable path from station x_l to station x_k. Between x_l and x_k there exist no other stations. d_{lk} is the distance between the two station. v_{lk}^{max} is the speed limit that each AGV is allowed to travel (we assume all AGVs have the same speed limit on the same path). t_{lk}^{min} is the shortest travel time for an AGV. We have,

  $$t_{lk}^{min} = \frac{d_{lk}}{v_{lk}^{max}}$$

Example 6.1 Layout: *Figure 6.1 shows the layout of an AGV-based Electronics Manufacture — Linn Product Ltd. Our implementation of the spring-based negotiation model is based on this layout. On this layout, a_i, b_i, ..., l_i are loading/unloading points near working desks, s_i are loading/unloading points at storerooms, and x_i are junctions. They are all taken as stations in this dissertation. There are many arcs. For example, (a_1, a_2) is an arc. But (a_2, a_1) is not an arc. Note, (a_1, a_3) is not an arc either, since there is a station a_2 between a_1 and x_3.*

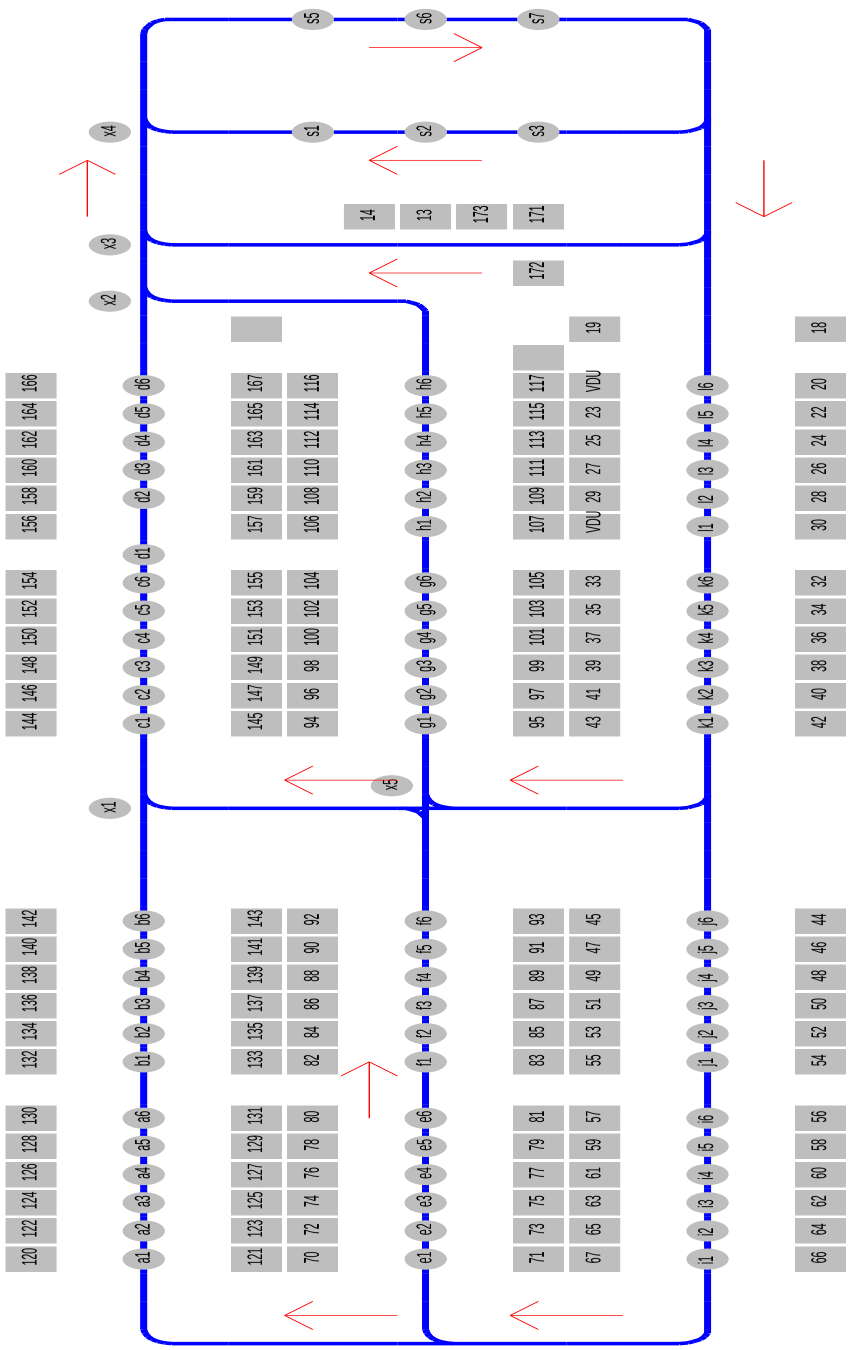


Figure 6.1: The Linn Product Ltd. Layout

6.3.2 A Local AGV schedule

A local AGV schedule can be represented by an ordered list of RTokens.

$$(rt_{j_1}, rt_{j_2}, ..., rt_{j_k}, ..., rt_{j_{N_j}})$$

where rt_{j_k} $(k = 1, 2, ..., N_j)$ is called an RToken, i.e., a reserved interval of space-time.

- j represents a particular job for an AGV j.
- j_k is indexed to a particular station x_{j_k} in $\mathcal{X}$.

We sometimes say that x_{j_k} or rt_{j_k} are the kth node of the local schedule of agent j. So, a local schedule means also an ordered list of stations

$$(x_{j_1}, x_{j_2}, ..., x_{j_k}, ..., x_{j_{N_j}})$$

which represents a traversable path, i.e,

$$(x_{j_k}, x_{j_{k+1}})\ (k = 1, 2, ..., (N_j - 1))$$

are arcs.

6.3.3 External Form and Internal Form of an RToken rt_{j_k}

External Form of an RToken rt_{j_k}

The external representation of rt_{j_k} is

$$(st_{j_k}, et_{j_k})$$

where st_{j_k} represents the starting time of this RToken and et_{j_k} represents the end time of the RToken. An external representation is useful when internal structure of the RToken is not of interest. Generally one AGV will not be interested in another AGV's RToken internal structures. It is interested in only their external expression.

Internal Form of an RToken rt_{j_k}

The internal structure of an RToken rt_{j_k} can be represented by

$$[tol^{\alpha}_{j_k}|\omega_{j_k}|tol^{\beta}_{j_k}]$$

The structure is illustrated in Figure 6.2.

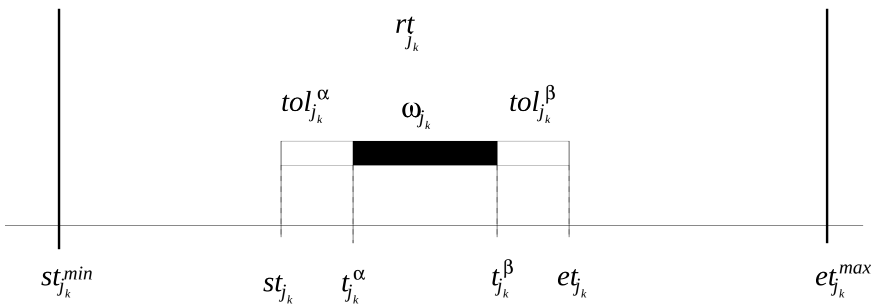


Figure 6.2: The Infra-Structure of Time Interval rt_{j_k}

- $t^{\alpha}_{j_k}$ is *the arrival time* scheduled.
- $tol^{\alpha}_{j_k}$ is *the front tolerant interval* set to allow the AGV arrive x_{j_k} earlier than $t^{\alpha}_{j_k}$.
- $tol^{\beta}_{j_k}$ is *the rear tolerant interval* set to allow the AGV arrive x_{j_k} later than $t^{\alpha}_{j_k}$.
- ω_{j_k} is the operational time required to complete an operation at x_{j_k}. Operation here means loading/unloading goods at or passing x_{j_k}. The width of ω_{j_k} is a fixed value given by the user.

We have

$$\begin{aligned} tol^{\alpha}_{j_k} &= (st_{j_k}, t^{\alpha}_{j_k}) \\ \omega_{j_k} &= (t^{\alpha}_{j_k}, t^{\beta}_{j_K}) \\ tol^{\beta}_{j_k} &= (t^{\beta}_{j_K}, t^{\beta}_{j_k}). \end{aligned}$$

The minimal number of variables for describing an RToken

By examining the internal structure of rt_{j_k}, we can see that 3 variables are enough to completely describe the RToken. Let us also use $tol^{\alpha}_{j_k}$, $tol^{\beta}_{j_k}$ and ω_{j_k} to denote their respective widths, then rt_{j_k} can be described by a set of variables

$$\{tol^{\alpha}_{j_k}, t^{\alpha}_{j_k}, tol^{\beta}_{j_k}\}$$

Other variables can be derived from these three variables:

- *the departure time*, $t^{\beta}_{j_k}$, at which, AGV j is planned to leave x_{j_k}:

$$t^{\beta}_{j_k} = t^{\alpha}_{j_k} + \omega_{j_k} \tag{6.1}$$

- *the starting time of the RToken*, st_{j_k}.

$$st_{j_k} = t^{\alpha}_{j_k} - tol^{\alpha}_{j_k} \tag{6.2}$$

- *the end time of the RToken*, et_{j_k}.

$$et_{j_k} = t^{\beta}_{j_k} + tol^{\beta}_{j_k} \tag{6.3}$$

- *the travel time*, $t^{\tau}_{j_k j_{k+1}}$, which is the time for AGV j to travel from x_{j_k} to $x_{j_{k+1}}$:

$$t^{\tau}_{j_k j_{k+1}} = t^{\alpha}_{j_{k+1}} - t^{\beta}_{j_k} = t^{\alpha}_{j_{k+1}} - t^{\alpha}_{j_k} - \omega_{j_k} \tag{6.4}$$

  $t^{\tau}_{j_k j_{k+1}}$ is an inter-node variable (see Fig 6.3).

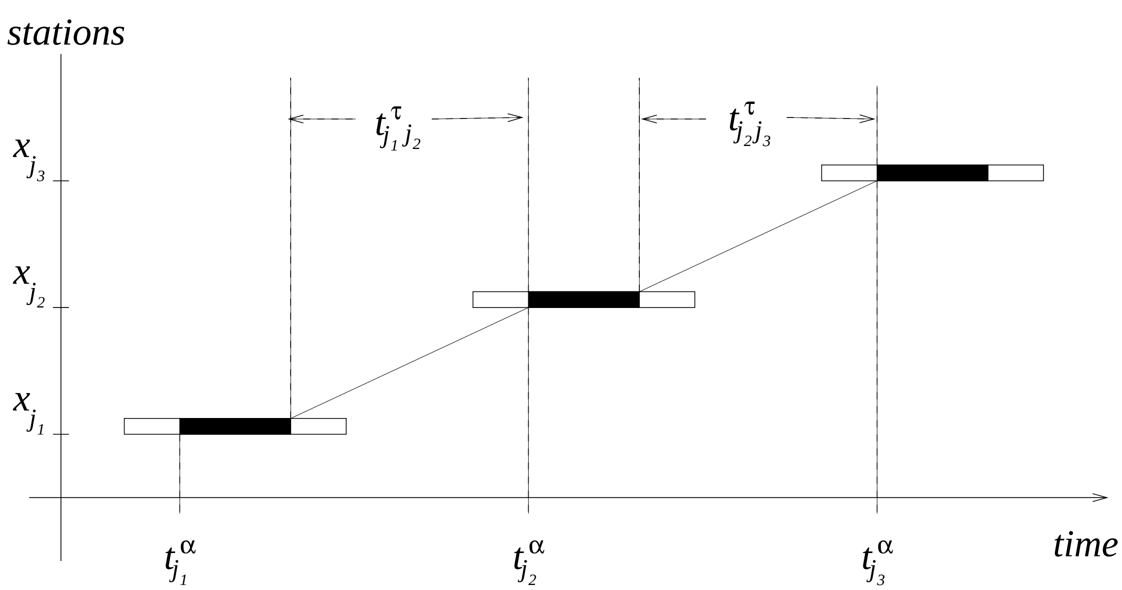


Figure 6.3: An Example of a Local Schedule for AGV j

Point-based vs Interval-based

Point-based scheduling does not allocate redundant resources for any AGV, formally,

$$\forall j, k \ tol^{\alpha}_{j_k} = tol^{\beta}_{j_k} = 0$$

Otherwise, we say schedules are *interval-based.*

6.3.4 Intra-agent Constraints

Notations

In this paragraph we define a set of notations for describing relations between two arbitrary RTokens, namely rt_a, rt_b (see Figure 6.4).

$$rt_a = (t^1_a, t^2_a) \ \ t^1_a \leq t^2_a$$

$$rt_b = (t^1_b, t^2_b) \ \ t^1_b \leq t^2_b$$

We say rt_a is BEFORE rt_b, if

$$t^2_a \geq t^1_b$$

and we denote this relation by $rt_a \| rt_b$. There are two possibilities for $rt_a \| rt_b$:

1. The first possibility is

   $$t^2_a = t^1_b$$

   we sometimes denote this by $rt_a | rt_b$ and say that rt_a TOUCHES rt_b.

2. The second possibility is

   $$t^2_a > t^1_b$$

   we sometimes denote this by $rt_a ||| rt_b$ and say that rt_a and rt_b SEPARATE.

We say rt_a is OVERLAPPING rt_b, if $rt_a \ \not\| rt_b$ and $rt_b \ \not\| rt_a$ and we denote this relation by $rt_a \rightleftharpoons rt_b$. So, (1) in Figure 6.4 (a) (b) $rt_a \| rt_b$, (2) in Figure 6.4 (a) $rt_a | rt_b$, and (3) in Figure 6.4 (c) (d) $rt_a \rightleftharpoons rt_b$ (or $rt_b \rightleftharpoons rt_a$). The relationships defined above are more simplistic and more suitable for describing the problems in the thesis than any others we know. For example, Allen [Allen & Hayes 87] defined a set

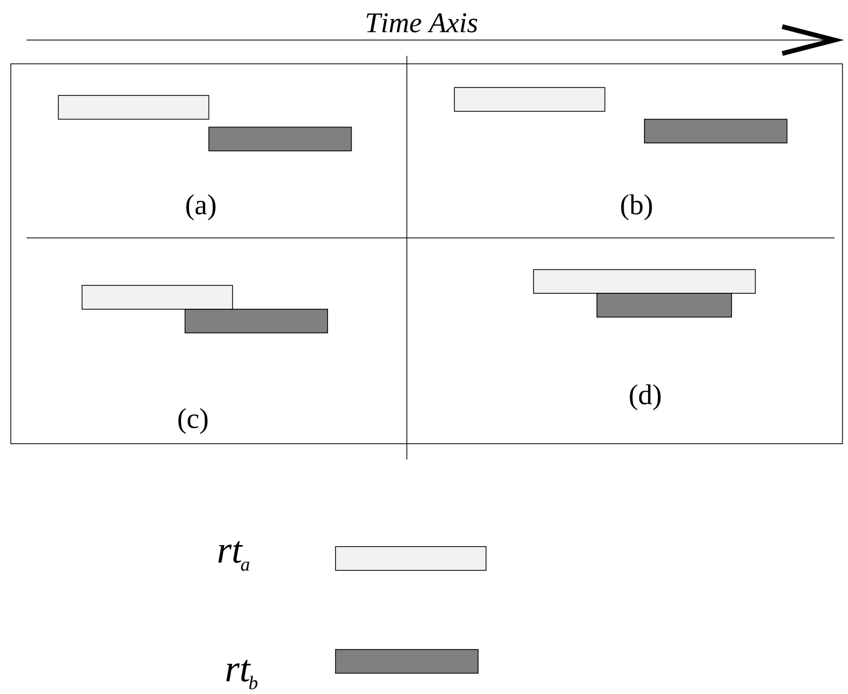


Figure 6.4: Relationships between Two Intervals

of interval relationships in logical formulae with universal quantifiers and existential quantifiers. The defined relationships include $MEETS$, $BEFORE$, $OVERLAPS$, $STARTS$, $FINISHES$, and $DURING$. We could prove that the three relationships can be used to derive all the relationships defined by Allen. So, they might be taken as the smallest and complete set of the temporal interval representation. The interested reader could do further investigation on this point. However, this is not the main issue of this thesis. We intuitively defined these relationships that, we think, are sufficient enough for our purpose. Similar description about the relationships between intervals can be also found in [Conway *et al.* 67]. Intra-agent constraints are those concerning relations between variables relating to one AGV.

Intra-agent Hard Constraints

Hard constraints are not relaxable. These constraints are listed as follow:

1. *No overlapping is allowed between any two RTokens reserved for the same AGV's activities*, i.e.,

$$rt_{j_k} \neq rt_{j_l} \tag{6.5}$$

2. *The starting time, st_{j_k}, of RToken rt_{j_k} must be no earlier than the earliest arrival time $st_{j_k}^{min}$*, i.e.,

$$st_{j_k} \geq st_{j_k}^{min} \tag{6.6}$$

$st_{j_k}^{min}$ is specified by the user;

3. *The end time, et_{j_k}, of RToken rt_{j_k} must be no later than the latest departure time, $et_{j_k}^{max}$*, i.e.,

$$et_{j_k} \leq et_{j_k}^{max} \tag{6.7}$$

$et_{j_k}^{max}$ is specified by the user;

4. *The travel time $t_{j_k j_{k+1}}^{\tau}$ between two stations x_{j_k} and $x_{j_{k+1}}$ must be no smaller than the mimimal travel time $t_{j_k j_{k+1}}^{min}$*

$$t_{j_k j_{k+1}}^{\tau} \geq t_{j_k j_{k+1}}^{min} \tag{6.8}$$

$(st_{j_k}^{min}, et_{j_k}^{max})$ is the bottom line of allocating resource to rt_{j_k}. The bottom line could determined by following way:

Suppose a local schedule is $(rt_{j_1}, rt_{j_2}, rt_{j_3})$, as a requirement, the user will often specify the earliest time $st_{j_1}^{min}$ from which the job could be started, and the latest time st_{j_3} before which the job must be finished. We then can derive the rest as follows:

$$\begin{aligned} st_{j_2}^{min} &= st_{j_1}^{min} + t_{j_1 j_2}^{min} \\ st_{j_3}^{min} &= st_{j_2}^{min} + t_{j_2 j_3}^{min} \\ et_{j_2}^{max} &= et_{j_3}^{max} - t_{j_2 j_3}^{min} \\ et_{j_1}^{max} &= et_{j_2}^{max} - t_{j_1 j_2}^{min} \end{aligned}$$

In other words, the left bottom line in Figure 6.5 is derived through forward propagation, and the right bottom line is derived through backward propagation.

6.3.5 Intra-agent Soft Constraints

A local schedule is an assignment of variables $t_{j_k}^{a}$, $tol_{j_k}^{\alpha}$ and $tol_{j_k}^{\beta}$ $(k = 1, 2, ...,)$ such that all *intra-agent* hard constraints are satisfied. Generally speaking, there exists

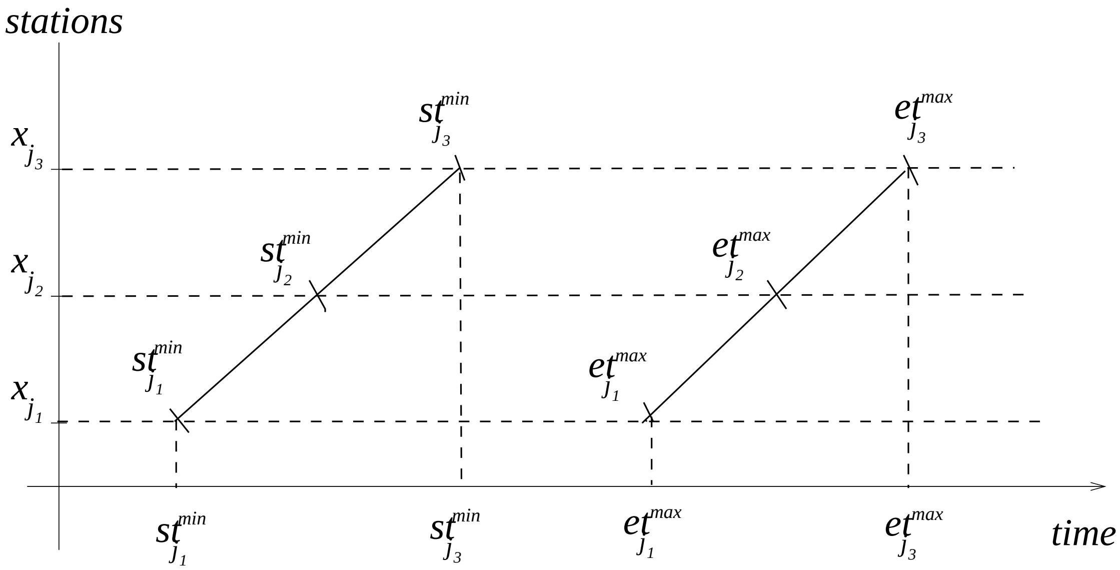


Figure 6.5: Bottom Lines for Resource Allocation

often more than one possible local schedule. The user often has his preferences over these local schedules. The preferences may include:

- The preferred arrival time $t^{\alpha *}_{j_k}$, i.e., the arrival time of $t^{\alpha}_{j_k}$ is as close to $t^{\alpha *}_{j_k}$ as possible.
- The preferred tolerance widths $tol^{\alpha *}_{j_k}$ and $tol^{\beta *}_{j_k}$. The preferred tolerance width will be decided by the user.
- The preferred travel time $t^{\tau *}_{j_k j_{k+1}}$. The preferred travel time could be decided by the optimal travel speed.

These preferences usually cannot be strictly satisfied, so we need have a method of measuring closeness between a real assignment and the preferences. The closeness can be represented as follow:

$$\left.\begin{array}{l} \Delta t^{\alpha}_{j_k} = t^{\alpha}_{j_k} - t^{\alpha *}_{j_k} \\ \Delta tol^{\alpha}_{j_K} = tol^{\alpha}_{j_K} - tol^{\alpha *}_{j_K} \\ \Delta tol^{\beta}_{j_K} = tol^{\beta}_{j_K} - tol^{\beta *}_{j_K} \\ \Delta t^{\tau}_{j_k j_{k+1}} = t^{\tau}_{j_k j_{k+1}} - t^{\tau *}_{j_k j_{k+1}} \end{array}\right\} \tag{6.9}$$

Hence, $\Delta t^{\alpha}_{j_k}, \Delta tol^{\alpha}_{j_K}, \Delta tol^{\beta}_{j_K}$, and $\Delta t^{\tau}_{j_k j_{k+1}}$ represent the respective errors between assignment and preferences. Consider Equation 6.4, we have

$$t^{\tau}_{j_k j_{k+1}} = t^{\alpha}_{j_{k+1}} - t^{\beta}_{j_k} = t_{j_{k+1}} - t^{\alpha}_{j_k} - \omega_{j_k} \tag{6.10}$$

therefore, we have

$$\Delta t^{\tau}_{j_k j_{k+1}} = \Delta t^{\alpha}_{j_{k+1}} - \Delta t^{\alpha}_{j_k} \tag{6.11}$$

So, $\Delta t^{\tau}_{j_k j_{k+1}}$ is a derived variable.

A good assignment will make these errors as small as possible. In this way of consideration, we can construct a local objective function e_j:

$$e_j = f_j(\Delta t^{\alpha}_{j_k}, \Delta tol^{\alpha}_{j_K}, \Delta tol^{\beta}_{j_K}, \Delta t^{\tau}_{j_k j_{k+1}}) \tag{6.12}$$

And $f_j(x_1, x_2, x_3, x_4)$ should have the following property:

$$\frac{df_j}{dx_i} \leq 0 (i = 1, 2., 3, 4) \tag{6.13}$$

That is to say, when x_i increases then f_j will decrease (or, strictly speaking, will not increase). So, to increase f_j, x_i must be as close to zero as possible. The following function can have this property:

$$\begin{gathered} e_j = -k^{\alpha}_{j_k}(\Delta t^{\alpha}_{j_k})^2 - k^{tol\alpha}_{j_k}(\Delta tol^{\alpha}_{j_K})^2 - k^{tol\beta}_{j_k}(\Delta tol^{\beta}_{j_K})^2 - k^{\tau}_{j_k j_{k+1}}(\Delta t^{\tau}_{j_k j_{k+1}})^2 \\ k^{\alpha}_{j_k}, k^{tol\alpha}_{j_k}, k^{tol\beta}_{j_k}, k^{\tau}_{j_k j_{k+1}} \geq 0 \end{gathered} \tag{6.14}$$

This objective function is usually called the energy function and especially suitable for evaluation of closeness of assignment.

6.3.6 Inter-agent Constraints

Inter-agent Hard Constraints

There are mainly two inter-agent hard constraints:

- *Overlapping is not allowed between two RTokens, rt_{a_k} and rt_{b_l}, reserved respectively for two agents*, formally,

$$\left.\begin{array}{l} rt_{a_k} \not\triangleq rt_{b_k} if \\ (1) a \neq b \\ (2) x_{a_k} = x_{b_l} \end{array}\right\}$$

- *Overtaking is not allowed.* In other words, if AGV a and AGV b will both travel along the same arc, then if agent a departs from the starting station of the arc first, then it is not allowed that b arrives the destination station of the arc first. Formally, we have

$$\left.\begin{array}{l} rt_{a_{k+1}} \| rt_{b_{k+1}} if \\ (1) a \neq b \\ (2) arc_{a_k a_{k+1}} = arc_{b_l b_{l+1}} \\ (2) rt_{a_k} \| rt_{b_k} \end{array}\right\}$$

A Global Objective

A global schedule is a collection of all AGVs' local schedules that satisfy inter-agent hard constraint. Having set the local objectives for each AGVs, we can expect that the global objective of AGVSP is that all local objectives are achieved as well as possible. The global objective could be:

$$E_{agvsp} = \Sigma_j e_j \tag{6.15}$$

6.3.7 Summary of Domain Specification

In this section, we systematically addressed the AGVSP problem by viewing it as a DCSP problem in which each subproblem is a local scheduling problem for an AGV. There are several constraints which are classified into hard constraints and soft constraints according to their relaxibility. Constraints are also divided into inter-agent constraints and intra-agent constraints. We also proposed the local objective function and global objective function for measuring the degree of satisfaction of soft constraints. This domain specification laid foundations for introducing the spring model in the rest of this Chapter.

6.4 Spring Model for AGV Local Scheduling problem

In this section, we describe a novel model — a *spring model* — for representing a local scheduling problem. The spring model will serve two purposes:

1. It, in any situation, provides an optimal solution to a local AGV scheduling problem in the sense that the local objective is e_j suggested by Equation 6.14.

2. It provides a negotiation skill for the respective agent during negotiation.

6.4.1 Intuitive Explanation of a Spring Model

In this paragraph, we explain why a spring model could be suitable for AGV local scheduling problems in an intuitive manner to help the reader to understand the appropriateness of the model. The explanation is given below:

- As we already know, the AGVSP problem can be described as a DCSP problem which consists of soft constraints and hard constraints. These constraints can also be classified into intra-agent constraints and inter-agent constraints. Hard constraints are not relaxable, so any solution must strictly satisfy them. Soft constraints are relaxable, and the objective function of the problem measures the quality of solutions in which some variable assignments have relaxed. Generally speaking, a solution that yields a higher utility (the value of the objective function) is better than one that yields a lower utility.

- In a negotiation process, agents negotiate when they are in conflict, i.e., some resources that are allocated by one agent to produce a high payoff utility are required by other agents. In this situation, some inter-agent constraints have been violated. In order to satisfy the inter-agent constraints violated, they have to discuss to reach an agreement on an alternative resource reallocation. It is obvious that hard constraints are not negotiable, therefore, whatever an agreement is, the resource reallocation must ensure that all hard constraints are strictly satisfied. So, reallocation actually means relaxing the degree of satisfaction of some soft constraints. In this sense, "relaxable" means "negotiable".

- Some soft constraints are more important than others. For example, an AGV which is loaded with goods urgently needed by an important customer should be scheduled to meet the customer's requirement concerning the delivery time. More important constraints often are less relaxable. Therefore we should have some way to describe the importance of a constraint.

- A soft constraint may be analogical to a spring. The relaxed state (when no external forces are acting upon it) of the spring is normally considered the best state of the spring (it keeps the lowest energy). The spring can be stretched (or compressed) by external forces. The more it is stretched (or compressed) the bigger the external forces are required. The force is proportional to the product of the length being stretched (or compressed) and Hooke's constant. If the force remains the same, then the larger Hooke's constant, the smaller the length change will be. This feature of the spring could be exploited to represent negotiating agents. Suppose, there are two RTokens, rt_{j_k} for agent j and rt_{i_l} for agent i. They overlap by T, then this overlap can be resolved by agent j conceding T_j and agent i conceding T_i ($T_j + T_i = T$). And T_j and T_i are decided by their relaxibility — or Hooke's constants: the more important RToken (e.g. a RToken for an activity that has an urgent deadline) will have a larger Hooke's constant (or smaller relaxibility).

6.5 The Basic Elements of a spring model

6.5.1 Hooke's Law and Newton's Law

A Spring and Hooke's Law

A body is said to be elastic if it suffers a deformation when a stretching or compressing force is applied to it [Hughes & Martin 77]. The force with which an elastic body resists deformation is called restoration. A linear spring is an elastic body which obeys a simple empirical law known as Hooke's Law: *the magnitude of the restoring force is directly and linearly proportional to the deformation* (see Figure 6.6):

$$\begin{aligned} F &= k\Delta l \\ \Delta l &= l - l_a \end{aligned} \tag{6.16}$$

where k is the Hooke's constant. l_a is the state of the spring when it is relaxed (Figure 6.6 (a)). Figure 6.6(b) shows the spring being compressed and Figure 6.6(c) shows the spring being stretched.

Newton's Third Law

We are interested in a static state of a spring. When a spring is in a static state, then the external force acting on the spring should be equal to the restoration force. This comes from Newton's Third Law [Ohanian 85] which gives the quantitative relationship between the action force (external force) and the reaction force (restoration force):

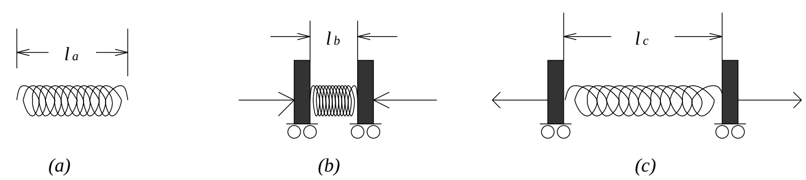


Figure 6.6: A Spring Being Stretched or Compressed

Whenever a body exerts a force on another body, the latter exerts a force of equal magnitude and opposite direction on the former.

Convention of describing forces

Since we are only interested in the static state of a spring, and Newton's third law states that the action force and reaction force should be equal, we will not state again whether a force we described is external force or internal force. To make the description simple, we will set a convention that all forces we describe are external and any force is directional: from left to right is the positive direction of a force. So, a force is denoted by $\vec{F}$. If the value of $\vec{F}$ is negative, then the force is from right to left; if the value of $\vec{F}$ is positive then the force is from left to right.

6.5.2 Basic elements for a spring model

A spring model is a proper connection of springs, movable walls, unmovable walls, hollow cylinders, and unbendable bars (see Fig 6.7). A movable wall (Figure 6.7(b)) will

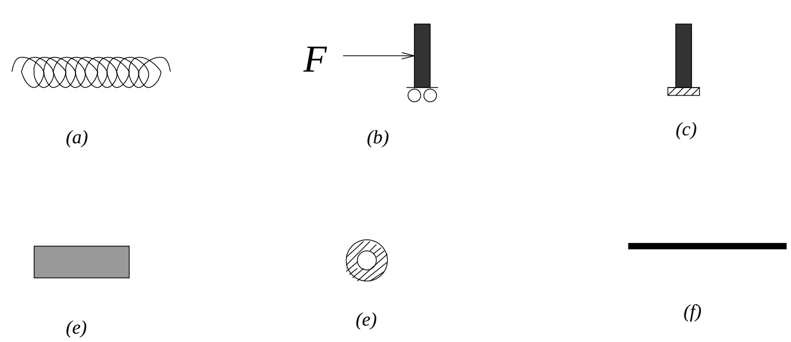


Figure 6.7: The Elements of Spring Models

be moved by an external force however small it is. An unmovable wall (Figure 6.7(c)) can never be moved by any force however large it is. A cylinder (Figure 6.7(d)) represents a constitutional part of an event that is unchangeable, such as the waiting time of an RToken, the minimal traverse time, etc. A cylinder may slide by external force along a dimension but can never be stretched or compressed by any force (A cylinder can be taken as a particular spring with $k = \infty$). Figure 6.7 (e) is the cross-section of the cylinder. Each cylinder is supported by a bar which passes through the hollow hole of the cylinder. The bar guarantees the cylinder to move only in two directions: move left and move right. An example can be seen in Figure 6.8 where ω_{j_k} can only move right or move left. An unbendable bar (Figure 6.7 (f)) is used to represent a connecting rod which connects two events.

6.5.3 A Spring unit for arrival time $t^{\alpha}_{j_k}$

The soft constraint for arrival time $t^{\alpha}_{j_k}$ is

$$\Delta t^{\alpha}_{j_k} = t^{\alpha}_{j_k} - t^{\alpha *}_{j_k} \tag{6.17}$$

Its spring model is shown in Figure 6.8. Where $st^{min}_{j_k}$ and $et^{max}_{j_k}$ are analogical to two unmovable walls which guarantees the departure time will not be earlier than $st^{min}_{j_k}$ and the arrival time will not later than $et^{max}_{j_k}$. The cylinder represents the operation time ω_{j_k}. When the spring is relaxed, $\Delta t^{\alpha}_{j_k} = 0$. This is the preferable state of $\Delta t^{\alpha}_{j_k}$.

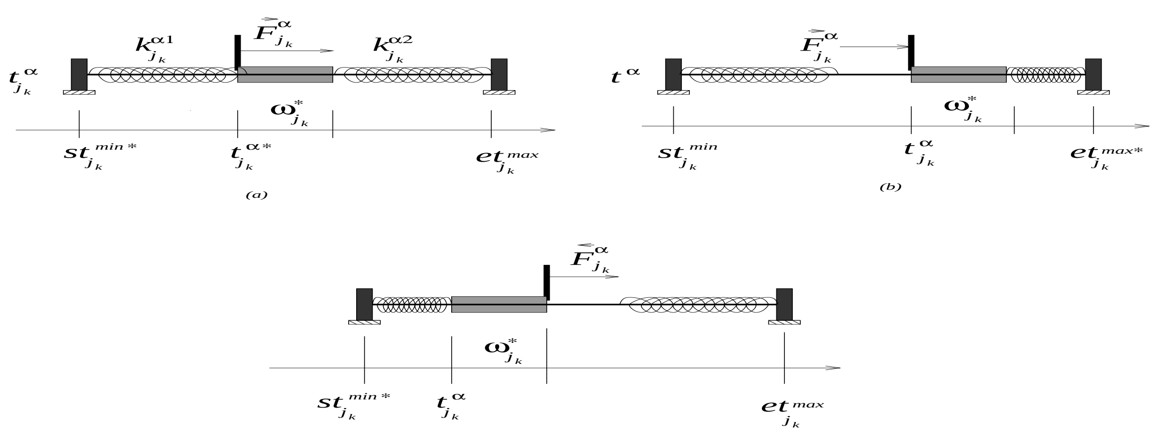


Figure 6.8: Spring Unit for $t^{\alpha}_{j_k}$

$k^{\alpha 1}_{j_k}$ and $k^{\alpha 2}_{j_k}$ represent the Hooke's Constants. In Figure 6.8 (b) when a force is applied, then the cylinder ω_{j_k} is pushed forwords (i.e., $\Delta t^{\alpha}_{j_k} > 0$) by a force ($\vec{F}^{a}_{j_k} > 0$, we have

$$\vec{F}^{a}_{j_k} = k^{\alpha 2}_{j_k} |\Delta t^{\alpha}_{j_k}| = k^{\alpha 2}_{j_k} \Delta t^{\alpha}_{j_k} \tag{6.18}$$

Likewise, when the cylinder is pushed backwards (i.e, $\Delta t^{\alpha}_{j_k} < 0$) by force $\vec{F}^{a}_{j_k} < 0$, then

$$-\vec{F}^{a}_{j_k} = k^{\alpha 1}_{j_k} |\Delta t^{\alpha}_{j_k}| = -k^{\alpha 1}_{j_k} \Delta t^{\alpha}_{j_k} \tag{6.19}$$

Then Equation 6.18 and Equation 6.18 can be unified as

$$\vec{F}^{a}_{j_k} = k^{\alpha}_{j_k} \Delta t^{\alpha}_{j_k} \tag{6.20}$$

where

$$k^{\alpha}_{j_k} = \begin{cases} k^{\alpha 1}_{j_k} & if \ \ \Delta t^{\alpha}_{j_k} < 0 \\ k^{\alpha 2}_{j_k} & if \ \ \Delta t^{\alpha}_{j_k} > 0 \end{cases} \tag{6.21}$$

$k^{\alpha 1}_{j_k}$ may be different from $k^{\alpha 2}_{j_k}$. For example,

$$k^{\alpha 1}_{j_k} < k^{\alpha 2}_{j_k}$$

means that lateness is worse than earliness if the AGV can not be scheduled to arrive at place at the required preferred arrival time.

6.5.4 A Spring unit for Traverse Time $t^{\tau}_{j_k j_{k+1}}$

$t_{j_k j_{k+1}}$ is the scheduled time for an AGV to travel from x_{j_k} to $x_{j_{k+1}}$. The hard constraint on it is

$$t^{\tau}_{j_k j_{k+1}} \leq t^{min}_{j_k j_{k+1}} \tag{6.22}$$

and the soft constraint on it is:

$$\Delta t^{\tau}_{j_k j_{k+1}} = t^{\tau}_{j_k j_{k+1}} - t^{\tau *}_{j_k j_{k+1}} = 0 \tag{6.23}$$

The spring model for $t^{\tau}_{j_k j_{k+1}}$ is one shown in Figure 6.9 (a). The cylinder represents the minimal traverse time $t^{min}_{j_k j_{k+1}}$ and two springs represent the rest parts of $t_{j_k j_{k+1}}$. When the spring is relaxed, $\Delta t_{j_k j_{k+1}} = 0$. This is the preferable state of the spring. Note that this spring is movable. When a force is applied to the spring, it will be deformed. In static state, we have

$$\begin{aligned} \vec{F}^{\tau 2}_{j_k} &= -k^{\tau}_{j_k j_{k+1}} \Delta t^{\tau}_{j_k j_{k+1}} \\ \vec{F}^{\tau 1}_{j_{k+1}} &= k^{\tau}_{j_k j_{k+1}} \Delta t^{\tau}_{j_k j_{k+1}} \end{aligned} \tag{6.24}$$

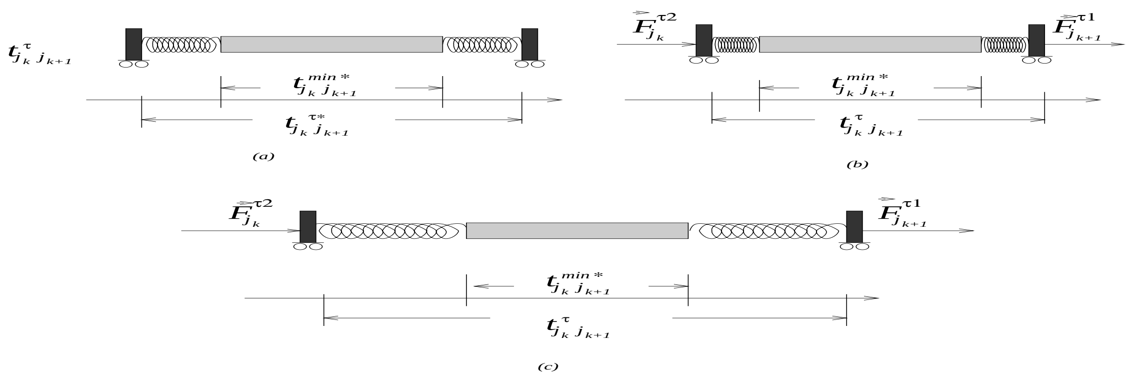


Figure 6.9: A Spring for $t^{\tau}_{j_k j_{k+1}}$

6.5.5 A spring Unit for RToken rt_{j_k}

Figure 6.10 (a) illustrates a spring unit for rt_{j_k}. The cylinder represents the ω_{j_k}; the spring on the left of the cylinder represents $tol^{\alpha}_{j_k}$; the spring on the right represents $tol^{\beta}_{j_k}$. The Hooke's constants for $tol^{\alpha}_{j_k}$ and $tol^{\beta}_{j_k}$ are respectively $k^{tol\alpha}_{j_k}$ and $k^{tol\beta}_{j_k}$. Both springs are attached to the cylinder as shown. rt_{j_k} are limited by two movable walls. The left wall represents the end time of an RToken (denoted by $rt^{L}_{j_k}$) which is reserved by an AGV at x_{j_k} and the RToken directly precedes rt_{j_k}. The right wall represents the starting time of an RToken (denoted by $rt^{R}_{j_k}$) which is reserved by an AGV at x_{j_k} and the RToken directly follows rt_{j_k}.

Note that, notations $rt^{L}_{j_k}$ and $rt^{R}_{j_k}$ will be used in situations where the agents which reserved them are not of our primary interest. Figure 6.10 (a) shows also the state of the spring model when it is relaxed. This is a situation where no other agents' resource allocations are considered. In this case,

$$\left.\begin{array}{l} \Delta tol^{\alpha}_{j_k} = tol^{\alpha}_{j_k} - tol^{\alpha *}_{j_k} = 0 \\ \Delta tol^{\beta}_{j_k} = tol^{\beta}_{j_k} - tol^{\beta *}_{j_k} = 0 \end{array}\right\} \tag{6.25}$$

In Figure 6.10 (b), $tol^{\beta}_{j_k}$ is compressed (e.g. consider a situation where the resource on right has been allocated to another agent). We have:

$$\begin{array}{l} \vec{F}^{tol\beta 1}_{j_k} = -k^{tol\beta}_{j_k} \Delta tol^{\beta}_{j_k} \\ \vec{F}^{tol\beta 2}_{j_k} = k^{tol\beta}_{j_k} \Delta tol^{\beta}_{j_k} \end{array} \tag{6.26}$$

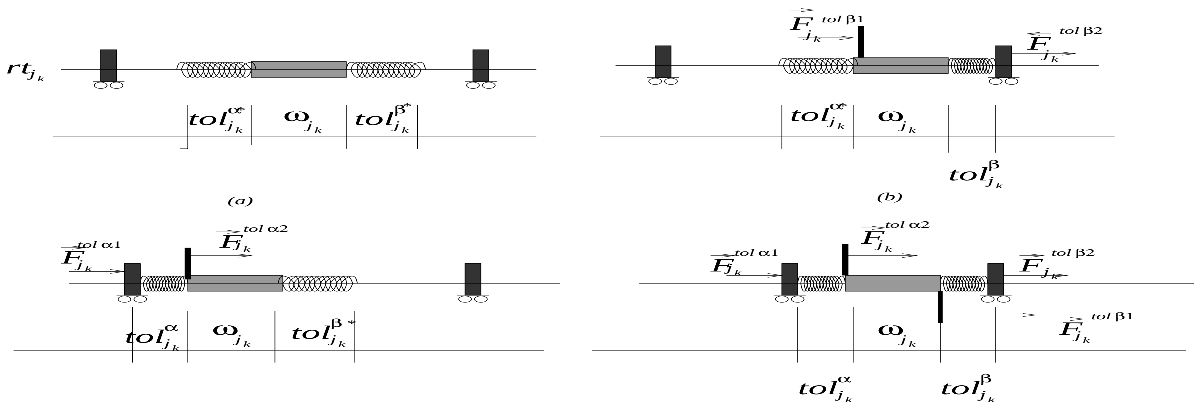


Figure 6.10: Spring Unit for rt_{j_k}

Remember, that $\Delta tol^{\beta}_{j_k}$ and $\Delta tol^{\alpha}_{j_k}$ are always negative. In Figure 6.10 (c), $tol^{\alpha}_{j_k}$ is compressed, we have:

$$\begin{aligned} \vec{F}^{tol\alpha 1}_{j_k} &= -k^{tol\alpha}_{j_k}\Delta tol^{\alpha}_{j_k} \\ \vec{F}^{tol\alpha 2}_{j_k} &= k^{tol\alpha}_{j_k}\Delta tol^{\alpha}_{j_k} \end{aligned} \tag{6.27}$$

Similarly, we have

$$\begin{aligned} \vec{F}^{tol\beta 1}_{j_k} &= -k^{tol\beta}_{j_k}\Delta tol^{\beta}_{j_k} \\ \vec{F}^{tol\beta 2}_{j_k} &= k^{tol\beta}_{j_k}\Delta tol^{\beta}_{j_k} \end{aligned} \tag{6.28}$$

6.6 A Spring-based Local Scheduling Model

Having defined basic spring units for rt_{j_k}, $t^{\tau}_{j_k j_{k+1}}$ and $t^{\alpha}_{j_k}$, we now propose a spring model for a local scheduling problem which is a spring network connecting these basic units (see Figure 6.11). In this figure, a three node schedule is presented though, it will be easily extended to an n node schedule. According to previous discussion, we

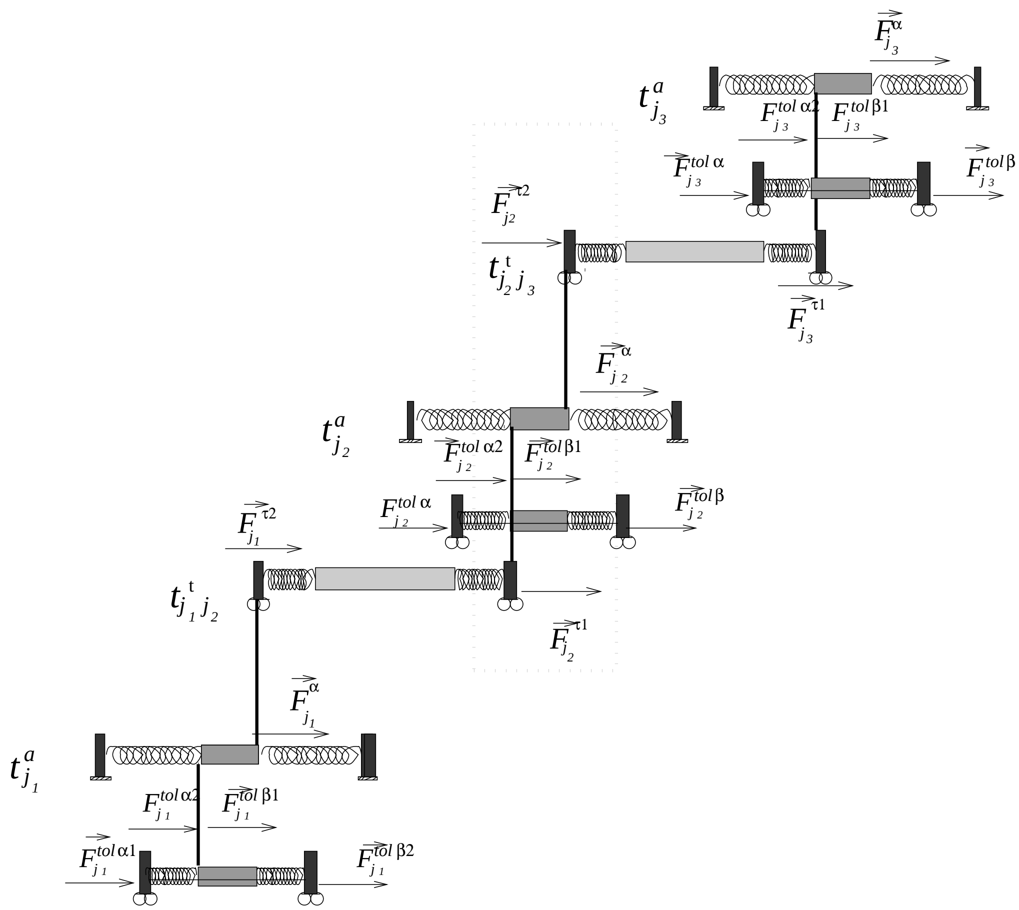


Figure 6.11: The Spring Model for Local Scheduling

summarise as the relationships between the components as follows:

$$\left.\begin{array}{l}
\overrightarrow{F}^{tol\alpha 1}_{j_k} = -k^{tol\alpha}_{j_k}\Delta tol {j_k}^{\alpha} \\
rforce^{tol\alpha 2}_{j_k} = k^{tol\alpha}_{j_k}\Delta tol {j_k}^{\alpha} \\
\overrightarrow{F}^{\tau 1}_{j_k} = k^{\tau}_{j_{k-1}j_k}\Delta t^{\tau}_{j_{k-1}j_k} \\
\overrightarrow{F}^{\tau 2}_{j_k j_{k+1}} = -k^{\tau}_{j_k j_{k+1}}\Delta t^{\tau}_{j_k j_{k+1}} \\
\overrightarrow{F}^{\alpha}_{j_k} = k^{\alpha}_{j_k}\Delta t^{\alpha}_{j_k} \\
\overrightarrow{F}^{tol\beta 1}_{j_k} = -k^{tol\beta}_{j_k}\Delta tol {j_k}^{\beta} \\
\overrightarrow{F}^{tol\beta 2}_{j_k} \quad k^{tol\beta}_{j_k}\Delta tol {j_k}^{\beta} \\
\Delta t^{\tau}_{j_k j_{k+1}} = \Delta t^{\alpha}_{j_{k+1}} - \Delta t^{\alpha}_{j_k}
\end{array}\right\} \tag{6.29}$$

6.7 Theoretical Justification for Local Scheduling Spring Model

We justify our spring local scheduling model by two steps. The first step is to find the equilibrium point of forces in the spring model according to Newton's Third Law. The second step is to find the optimal local scheduling solution when the objective is an energy function given by Equation 6.14.

6.7.1 Equilibrium of Forces

When a spring network is in static state the resultant of forces at any point in the network should be equal to zero. In Figure 6.11, the resultant of forces in the dotted rectangle should be equal to zero (see also Figure 6.12), i.e.,

$$\vec{F}^{\tau 1}_{j_k} + \vec{F}^{tol\alpha 2}_{j_k} + \vec{F}^{\alpha}_{j_k} + \vec{F}^{tol\beta 1}_{j_k} + \vec{F}^{\tau 2}_{j_k} = 0 \tag{6.30}$$

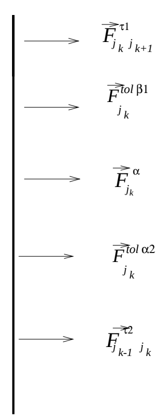


Figure 6.12: The Spring Model for Local Scheduling

6.8 Optimal local scheduling solution

When the objective function of a local AGV scheduling problem for agent j is given by Equation 6.14, then we can find its optimal solution by solving simultaneous equations:

$$\frac{de_j}{d\Delta t^{\alpha}_{j_k}} = 0 \quad (k = 1, 2, ..) \tag{6.31}$$

6.8.1 Optimal local scheduling when resources are fully available

When resources are fully available for a local scheduling, inter-agent constraints will not possibly be violated. In this case, tolerance times can always be fully guaranteed, i.e.,

$$\Delta tol^{\alpha}_{j_k} = \Delta tol^{\beta}_{j_k} = 0 \tag{6.32}$$

We only need to consider how to assign $\Delta t^{\alpha}_{j_k}$ and $\Delta t^{\tau}_{j_k j_{k+1}}$. Since we have

$$\Delta t^{\tau}_{j_k j_{k+1}} = \Delta t^{\alpha}_{j_{k+1}} - \Delta t^{\alpha}_{j_k} \tag{6.33}$$

consider, Equation 6.14, Equation 6.31 and Equation 6.33, we can derive the following simultaneous equations:

$$k^{\tau}_{j_{k-1} j_k} \Delta t^{\tau}_{j_{k-1} j_k} + k^{\alpha}_{j_k} \Delta t^{\alpha}_{j_k} - k^{\tau}_{j_k j_{k+1}} \Delta t^{\tau}_{j_k j_{k+1}} = 0 \quad (k = 0, 1, ..., N_j) \tag{6.34}$$

Compare this Equation 6.34 with Equation 6.29, we can see that, in Equation 6.34, $k^{\tau}_{j_{k-1} j_k} \Delta t^{\tau}_{j_{k-1} j_k}$ is in the same form of $\overrightarrow{F}^{\tau 1}_{j_k}$, $-k^{\tau}_{j_k j_{k+1}} \Delta t^{\tau}_{j_k j_{k+1}}$ is in the same form of $\overrightarrow{F}^{\tau 2}_{j_k}$ and $k^{\alpha}_{j_k} \Delta t^{\alpha}_{j_k}$ is in the same form of $\overrightarrow{F}^{\alpha}_{j_k}$, so we have

$$\overrightarrow{F}^{\tau 1}_{j_k} + \overrightarrow{F}^{\alpha}_{j_k} + \overrightarrow{F}^{\tau 2}_{j_k} = 0 \quad (k = 0, 1, ..., N_j) \tag{6.35}$$

Also, since Equation 6.32, we know that

$$\left.\begin{array}{l} \overrightarrow{F}^{tol\alpha 2}_{j_k} = k^{tol\alpha}_{j_k} \Delta tol^{\alpha}_{j_k} = 0 \\ \overrightarrow{F}^{tolbeta1}_{j_k} = -k^{tol\beta}_{j_k} \Delta tol^{\beta}_{j_k} = 0 \end{array}\right\} \tag{6.36}$$

Consider Equation 6.35 and Equation 6.36, we then have

$$\overrightarrow{F}^{\tau 1}_{j_k} + \overrightarrow{F}^{tol\alpha 2}_{j_k} + \overrightarrow{F}^{\alpha}_{j_k} + \overrightarrow{F}^{tol\beta 1}_{j_k} + \overrightarrow{F}^{\tau 2}_{j_k} = 0 \tag{6.37}$$

Compare Equation 6.37 and Equation 6.30, we can conclude:

Theorem 6.1 *The spring model proposed in Section 6.6 suggests an optimal local schedule in the sense of e_j given by Equation 6.14 when resources are fully available.*

That is to say, the spring model is suitable for local AGV scheduling when resources are fully available.

6.8.2 Optimal local scheduling when resources are constrained

One of major difficulties for any scheduling problem is that resources are tight. Figure 6.13 shows RToken rt_{j_k}'s resource assignment when there are a total amount, T, of resources in short:

$$T_{j_k} = T^1_{j_k} + T^2_{j_k} \tag{6.38}$$

$T^1_{j_k}$ and $T^2_{j_k}$ are two constants representing the unavailable amounts of resources respectively on the left hand side of rt_{j_k} and on the right hand side of rt_{j_k}. That is to say, on left side, rt_{j_k} must concede by $T^1_{j_k}$ and on right side rt_{j_k} must concede by $T^2_{j_k}$.

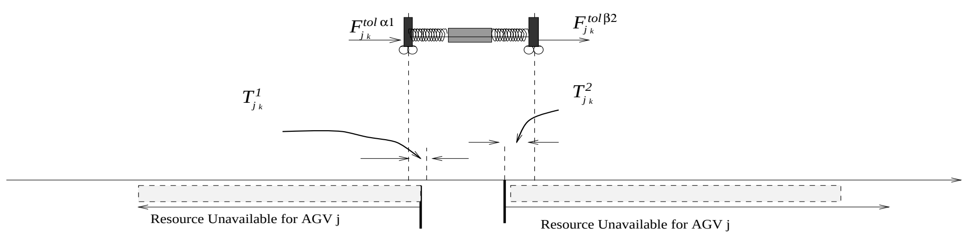


Figure 6.13: Resources Are Not Fully Available

That is to say, on left side, rt_{j_k} must concede by T_1 and on right side rt_{j_k} must concede by T_2. This problem could be overcome by yielding some tolerant times and/or shifting the RToken:

$$\left.\begin{array}{l} T_1 = -\Delta tol^{\alpha}_{j_k} + \Delta t^{\alpha}_{j_k} \\ T_2 = -\Delta tol^{\alpha}_{j_k} - \Delta t^{\alpha}_{j_k} \end{array}\right\} \tag{6.39}$$

From Equation 6.39, Equation 6.33 and the Equation 6.14, we can derive:

$$\left.\begin{array}{l} k^{tol\alpha}_{j_k} \Delta tol^{\alpha}_{j_k} + k^{\tau}_{j_{k-1}j_k} \Delta t^{\tau}_{j_{k-1}j_k} + k^{\alpha}_{j_k} \Delta t^{\alpha}_{j_k} - k^{\tau}_{j_k j_{k+1}} \Delta t^{\tau}_{j_k j_{k+1}} - k^{tol\beta}_{j_k} \Delta tol^{\beta}_{j_k} = 0 \\ (k = 1, 2, ..., N_j) \end{array}\right\} \tag{6.40}$$

If we do the similar replacements as in Section 6.8.1, then we have:

$$\overset{\rightarrow \tau 1}{F}_{j_k} + \overset{\rightarrow tol\alpha 2}{F}_{j_k} + \overset{\rightarrow \alpha}{F}_{j_k} + \overset{\rightarrow tol\beta 1}{F}_{j_k} + \overset{\rightarrow \tau 2}{F}_{j_k} = 0 \tag{6.41}$$

This equation is the same as Equation 6.30, therefore, we conclude:

Theorem 6.2 *The spring model given in Section 6.6 suggests an optimal local AGV schedule in the sense of e_j given by Equation 6.14 when resources are constrained.*

6.8.3 Solving the force equilibrium

Based on Theorem 6.2, then an optimal solution can be obtained by solving the force equilibrium equations, Equation 6.30. Considering Equation 6.29, Equation 6.39, we have:

$$\left.\begin{array}{l} \mathbf{\Delta t_j^\alpha = A_j^{-1} \Delta C_j} \\ \mathbf{\Delta tol_j^\alpha = \Delta t_j^\alpha - \Delta T_j^1} \\ \mathbf{\Delta tol_j^\beta = -\Delta t_j^\alpha - \Delta T_j^2} \\ \mathbf{\Delta t_j^\tau = A_j^\tau \Delta t_j^\alpha} \end{array}\right\} \tag{6.42}$$

where $\mathbf{A_j}$, $\mathbf{\Delta t_j^\alpha}$, $\mathbf{\Delta C_j}$, $\mathbf{\Delta t_j^\tau}$, $\mathbf{\Delta tol_j^\beta}$, $\mathbf{\Delta tol_j^\alpha}$, $\mathbf{A_j^\tau}$, $\mathbf{\Delta T_j^1}$ and $\mathbf{\Delta T_j^2}$ are all matrix:

$$\mathbf{\Delta t_j^\alpha} = \begin{bmatrix} \Delta t_{j_1}^\alpha \\ \Delta t_{j_2}^\alpha \\ \ldots \\ \Delta t_{j_k}^\alpha \\ \ldots \end{bmatrix}_{N_j \times\ 1} \tag{6.43}$$

$$\mathbf{\Delta tol_j^\alpha} = \begin{bmatrix} \Delta tol_{j_1}^\alpha \\ \Delta tol_{j_2}^\alpha \\ \ldots \\ \Delta tol_{j_k}^\alpha \\ \ldots \end{bmatrix}_{N\ \times\ 1} \tag{6.44}$$

$$\mathbf{\Delta tol_j^\beta} = \begin{bmatrix} \Delta tol_{j_1}^\beta \\ \Delta tol_{j_2}^\beta \\ \ldots \\ \Delta tol_{j_k}^\beta \\ \ldots \end{bmatrix}_{N_j \times\ 1} \tag{6.45}$$

$$\mathbf{T_j^i} = \begin{bmatrix} T_{j_1}^i \\ T_{j_2}^i \\ \ldots \\ T_{j_k}^i \\ \ldots \end{bmatrix}_{N_j \times\ 1} \qquad (i = 1, 2) \tag{6.46}$$

$$\mathbf{\Delta t_j^\tau} = \begin{bmatrix} \Delta t_{j_1 j_2}^\tau \\ \Delta t_{j_2 j_3}^\tau \\ \ldots \\ \Delta t_{j_k j_{k+1}}^\tau \\ \ldots \\ \Delta t_{j_{N_j - 1} j_{N_j}} \end{bmatrix}_{(N_j - 1)\ \times\ 1} \tag{6.47}$$

$$\mathbf{A_j} = \begin{bmatrix} a_{j_1} & -k^{\tau}_{j_1j_2} & 0 & 0 & 0 & ... \\ -k^{\tau}_{j_1j_2} & a_{j_2} & -k^{\tau}_{j_2j_3} & 0 & 0 & ... \\ 0 & -k^{\tau}_{j_2j_3} & a_{j_3} & -k^{\tau}_{j_3j_4} & 0 & ... \\ 0 & 0 & -k^{\tau}_{j_3j_4} & a_{j_4} & -k^{\tau}_{j_4j_5} & ... \\ 0 & 0 & ... & ... & ... & ... \end{bmatrix}_{N_j \times N_j} \tag{6.48}$$

$$\left. \begin{array}{l} a_{j_k} = k^{\tau}_{j_{k-1}j_k} + k^{tol\alpha}_{j_k} + k^{\alpha}_{j_k} + k^{tol\beta}_{j_k} + k^{\tau}_{j_kj_{k+1}} \\ (k = 1, 2, ..., N_j) \\ k^{\tau}_{j_0j_1} = k^{\tau}_{j_Nj_{N+1}} = 0 \end{array} \right\} \tag{6.49}$$

$$\mathbf{A^{\tau}_j} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & ... \\ 0 & -1 & 1 & 0 & 0 & ... \\ 0 & 0 & -1 & 1 & 0 & ... \\ 0 & 0 & ... & ... & ... & ... \end{bmatrix}_{(N_j-1) \times N_j} \tag{6.50}$$

$$\mathbf{C_j} = \mathbf{K^{tol\alpha}_j T^1_j} - \mathbf{K^{tol\beta}_j T^2_j} \tag{6.51}$$

$$\mathbf{K^{tol\alpha}_j} = \begin{bmatrix} k^{tol\alpha}_{j_1} & 0 & 0 & ... \\ 0 & k^{tol\alpha}_{j_2} & 0 & ... \\ .. & ... & ... & ... \end{bmatrix}_{N_j \times N_j} \tag{6.52}$$

$$\mathbf{K^{tol\beta}_j} = \begin{bmatrix} k^{tol\beta}_{j_1} & 0 & 0 & ... \\ 0 & k^{tol\beta}_{j_2} & 0 & ... \\ .. & ... & ... & ... \end{bmatrix}_{N_j \times N_j} \tag{6.53}$$

So, when we know how much resources are in short, i.e., what are $\mathbf{T^1_j}$ and $\mathbf{T^2_j}$, then we are able to get the optimal solution to a local scheduling problem through Equation 6.42. Care must be taken when some spring unit is not alive (or is dead) then a_{j_k} should be modified based on its basic form given in Equation 6.49. By " a spring unit is not alive", we mean the spring is either over stretched or over compressed such that it has lost elasticity. For example, when $\Delta tol^{\alpha}_{j_k}$ is smaller than $-tol^{\alpha *}_{j_k}$, then there is no tolerance time left and therefore the spring $tol^{\alpha}_{j_k}$ can no longer be compressed. In this case, we say $tol^{\alpha}_{j_k}$ is not alive. We will discuss this in Section 6.9.

6.8.4 Optimal Global Scheduling

When the global objective of an AGVSP is E_{agvsp} and

$$E_{agvsp} = \sum_j e_j \tag{6.54}$$

there is always a possibility of a resource conflict when two or more agents are making gains on achieving their individual goals. Suppose, two arbitrary RTokens, rt_{j_m} for agent j and rt_{i_n} for agent i, are reserved at the same station, and overlap T (see Figure 6.14), and also suppose that this conflict will be resolved by agent j conceding $T^2_{j_m}$ and agent i conceding $T^1_{i_n}$,

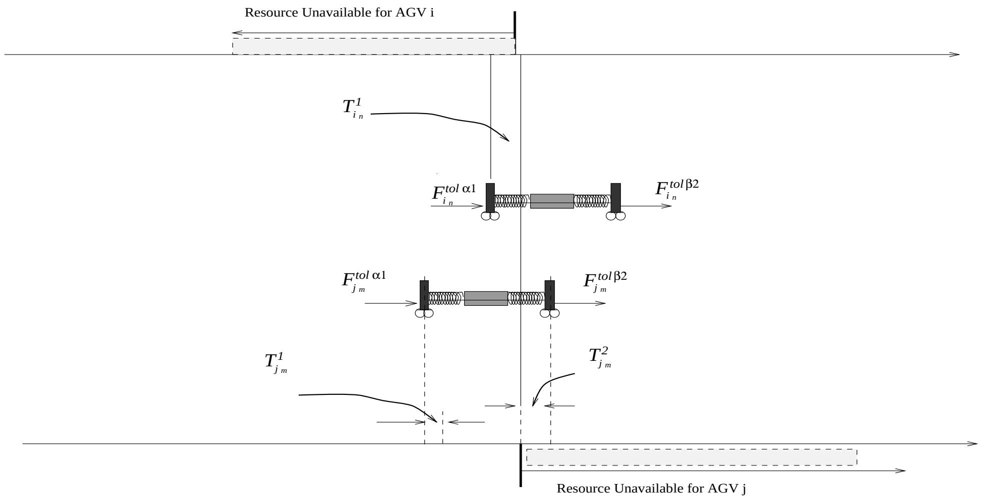


Figure 6.14: Two Agents' Resource Allocation Overlap

$$T^2_{j_m} + T^1_{i_n} = T \tag{6.55}$$

according to Newton's Third Law, then in the the spring model for agent j, the following equilibrium must be hold:

$$\vec{F}^{\tau 1}_{j_k} + \vec{F}^{tol\alpha 2}_{j_k} + \vec{F}^{\alpha}_{j_k} + \vec{F}^{tol\beta 1}_{j_k} + \vec{F}^{\tau 2}_{j_k} = 0 \text{ for all } j \tag{6.56}$$

$$\vec{F}^{tol\beta 2}_{j_m} + \vec{F}^{tol\alpha 1}_{i_n} = 0 \tag{6.57}$$

Now, we will prove that, suppose e_j is given by Equation 6.14, then, to maximise E_{agvsp}, Δt_{j_k}, $\Delta t_{j_k j_{k+1}}$, $\Delta t^{tol\alpha}_{j_k}$ and $\Delta t^{tol\beta}_{j_k}$ must also satisfy Equation 6.56 and Equation 6.57. The proof is delivered in the following steps:

step 1: Since E_{agvsp} is the sum of all local objective functions, therefore given a fixed amount of resources, each agent j must maximise e_j for the objective E_{agvsp} so that E_{agvsp} can be maximised. For two arbitrary agents, agent j (here j is a specific agent) and agent i (i is another specific agent), when they have an overlap as indicated in Figure 6.14, let us assume that that agent j will concede $T^2_{j_m}$ and agent i will concede $T^1_{i_n}$. In this case, we know from Theorem 6.2 that solutions to maximising e_j and e_i will be given by their respective spring models. That is to say, the solutions are decided according to the force equilibrium, Equation 6.56. Let us denote the optimal assignments for agent j are $\Delta t^{\alpha **}_{i_k}$, $\Delta t^{\tau **}_{j_k j_{k+1}}$, $\Delta tol^{\alpha **}_{j_k}$ and $\Delta tol^{\beta **}_{j_k}$ $(k = 1, 2, ..., N_j)$, and the optimal assignments for agent i are $\Delta t^{\alpha **}_{i_l}$, $\Delta t^{\tau **}_{i_l j_{l+1}}$, $\Delta tol^{\alpha **}_{i_l}$ and $\Delta tol^{\beta **}_{i_l}$ $(l = 1, 2, ..., N_i)$.

step 2: Let us rearrange E_{agvsp} according to Equation 6.54

$$E_{agvsp} = E^{'}_{agvsp} + E^{''}_{agvsp} \tag{6.58}$$

where $E^{'}_{agvsp}$ consists of items relating only to $\Delta tol^{\beta}_{j_m}$ and $\Delta tol^{\alpha}_{i_n}$, and $E^{'}_{agvsp}$ consists of all the rest of E_{agvsp}. Since e_j and e_i are given according to Equation 6.14, so

$$E^{'}_{agvsp} = k^{tol\beta}_{j_m}(\Delta tol^{\beta **}_{j_m})^2 + k^{tol\alpha}_{i_n}(\Delta tol^{tol\alpha **}_{i_n})^2 \tag{6.59}$$

From Figure 6.14, we can see that, $\Delta tol^{\beta}_{j_m}$ and $\Delta tol^{\alpha}_{i_n}$ can be reassigned values without demanding any further resource and influencing other variables' assignments, if

$$\Delta tol^{\beta}_{j_m} + \Delta tol^{\alpha}_{i_n} = \Delta tol^{\beta **}_{j_m} + \Delta tol^{\alpha **}_{i_n} \tag{6.60}$$

or

$$\Delta tol^{\alpha}_{i_n} = tol^{\beta **}_{j_m} + \Delta tol^{\alpha **}_{i_n} - tol^{\beta}_{j_m} \tag{6.61}$$

can be satisfied. Since such reassignment will not influence other variables' assignment, therefore, $E^{''}_{agvsp}$ will not be affected. Now, let us find an assignment that maximises $E^{'}_{agvsp}$ from the following equation:

$$\frac{dE^{'}_{agvsp}}{d\Delta tol^{\beta}_{j_m}} = 2k^{tol\beta}_{j_m}\Delta tol^{\beta}_{j_m} - 2k^{tol\alpha}_{i_n}\Delta tol^{\alpha}_{i_n} = 0 \tag{6.62}$$

or

$$k^{tol\beta}_{j_m}\Delta tol^{\beta}_{j_m} - k^{tol\alpha}_{i_n}\Delta tol^{\alpha}_{i_n} = 0 \tag{6.63}$$

We know from Equation 6.29 that this equation is equivalent to:

$$\vec{F}^{tol\beta 2}_{j_m} + \vec{F}^{tol\alpha 1}_{i_n} = 0 \tag{6.64}$$

That is to say, optimal global assignments must also satisfy Equation 6.57.

From the discussion in **step 1** and **step 2**, we have the following conclusion:

Theorem 6.3 *In a global AGVSP problem, when* **the precedence order of all RTokens is fixed**, *the spring network which connects all local spring models according to the order suggests an optimal solution to the AGVSP in the sense of E_{agvsp} where e_j for all agent j is given according to Equation 6.14.*

The phrase "the precedence order of all RTokens is fixed" must be emphasised, since the spring model network does not explicitly tell how those RTokens should be ordered.

6.9 Negotiation Equations

6.9.1 About Spring Constants

Spring constant for $t^{\alpha}_{j_k}$

In previous discussion, we implicitly assume that the restoration force of a spring is linearly proportional to the deformation of the spring. However, this assumption must be used very carefully. Let us first examine the Hooke's constant $k^{\alpha}_{j_k}$ of spring $t^{\alpha}_{j_k}$ (see Figure 6.15). We normally expect $t^{\alpha}_{j_k}$ to be within the normal region (t_3, t_4) (or

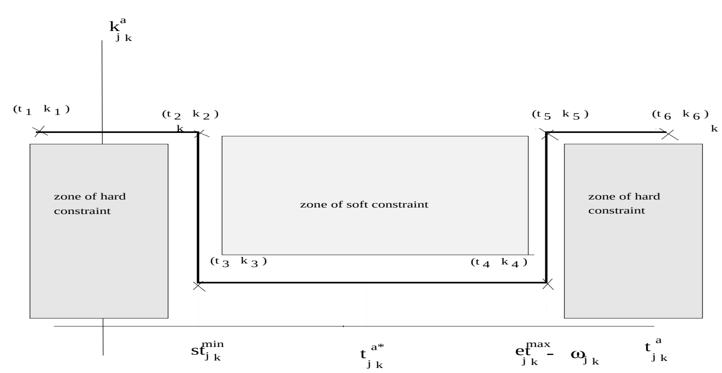


Figure 6.15: Hooke's Constant of Spring $t^{\alpha}_{j_k}$

$(st^{min}_{j_k}, et^{max}_{j_k} - \omega_{j_k})$. In this region, the spring has normal elasticity. Suppose the

Hooke's constant is $k^{\alpha}_{j_k}$, i.e.,

$$k_3 = k_4 = k^{\alpha}_{j_k}$$

Now, suppose that the spring is over-compressed such that $t^{\alpha}_{j_k}$ goes to the region of (t_1, t_2) (or $(t_1, st^{min}_{j_k})$), then spring should no longer be able to be compressed since (t_1, t_2) is region of hard constraint for $t^{\alpha}_{j_k}$. In this case, we say the spring is dead or not alive. Theoretically, we should assign assume that the spring constant in this region is ∞, i.e.,

$$k_1 = k_2 = \infty$$

However, practically, we can assume the spring constant is a finite number relatively much larger than normal $k^{\alpha}_{j_k}$, $k^{\tau}_{j_k j_{k+1}}$, $k^{tol\alpha}_{j_k}$ and $k^{tol\beta}_{j_k}$. Likewise, if the spring is over-stretched such that $t^{\alpha}_{j_k}$ goes into region (t_5, t_6) (or $(et^{max}_{j_k} - \omega_{j_k}, t_6)$), then the spring is dead. Similarly, instead of giving the spring an infinite spring constant, we assign the spring a relatively large spring constant but not infinite. In this way of consideration, the spring constant $k^{\alpha}_{j_k}$ should be expressed in the following way:

$$k^{\alpha}_{j_k} = \begin{cases} k^{\infty} & \text{if } \ t^{\alpha}_{j_k} < st^{min}_{j_k} \ \text{ or } \ t^{\alpha}_{j_k} > et^{max}_{j_k} - \omega_{j_k} \\ k_{j_k} & \text{otherwise} \end{cases} \tag{6.65}$$

where, k^{∞} is a very large number. So, even when the spring is dead, if there is an external force acting upon it, it can still be deformed, but the degree of deformation should be very small.

The reason of assigning a spring in dead regions (the zones of hard constraints as shown in Figure 6.15) is that we want the spring to be alive all the time such that it could automatically recover to its relaxed state should the external force be removed. We know that a spring has function of remembering its history of being deformed.

Spring constant for $tol^{\alpha}_{j_k}$ and $tol^{\beta}_{j_k}$

Since $tol^{\alpha}_{j_k}$ and $tol^{\beta}_{j_k}$ are the similar variables, we only discuss about one of their spring unit, i.e., $tol^{\alpha}_{j_k}$. Figure 6.16 illustrates the spring constant $k^{tol\alpha}_{j_k}$ for spring $tol^{\alpha}_{j_k}$. There are three regions:

- The first region is "zone of soft constraint" where $tol^{\alpha}_{j_k}$ has normal elasticity (the spring is alive). In this case, the spring constant is $k^{tol\alpha}_{j_k}$, i.e.,

$$k_3 = k_4 = k^{tol\alpha}_{j_k}$$

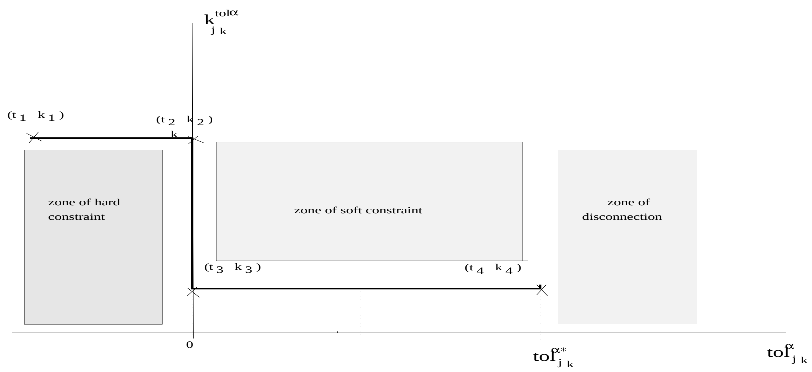


Figure 6.16: Spring Constant for $k^{tol\alpha}_{j_k}$

- The second region is "zone of hard constraint". For the same reason given in Section 6.9.1, the spring is dead, but we assign the spring a relative large spring constant.

- The third region is a "zone of disconnection". In this region, external forces will not be able to act upon it. More specifically, in this case, RToken rt_{j_k} does not TOUCH any RToken $rt^L_{j_k}$ (an arbitrary RToken on the left of rt_{j_k}). Therefore, the spring is virtually disconnected from any other spring. Since no external force is acting upon it, it will not be deformed, therefore, there is no possibility for $tol^{\alpha}_{j_k}$ going into this region. In this way of consideration, the spring constant in this region is not necessarily to be considered.

To conclude the above explanation, we have:

$$k^{tol\alpha}_{j_k} = \begin{cases} k^{\infty} & \text{if } \; tol^{\alpha}_{j_k} < 0 \\ k^{tol\alpha}_{j_k} & \text{if } \; 0 \leq tol^{\alpha}_{j_k} \leq tol^{\alpha*}_{j_k} \\ \phi & \text{otherwise} \end{cases} \tag{6.66}$$

Spring Constant for $t^{\tau}_{j_k j_{k+1}}$

The spring constant $k^{\tau}_{j_k j_{k+1}}$ for spring $t^{\tau}_{j_k j_{k+1}}$ is shown in Figure 6.17. It has two regions, "zone of hard constraint" where the spring is dead, and "zone of soft constraint" where the spring is alive. For the same reasons as we presented in the previous two paragraph,

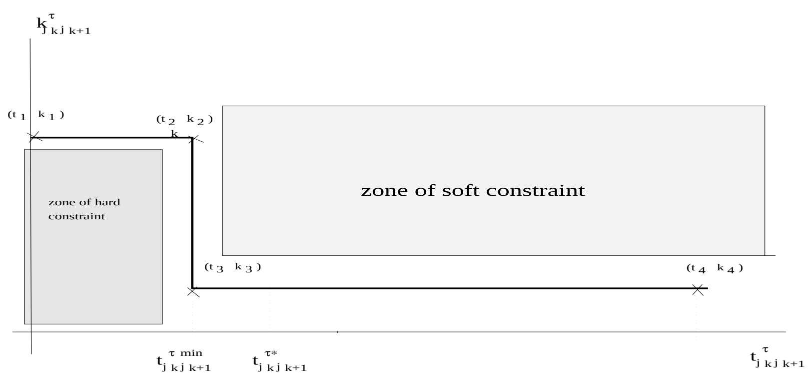


Figure 6.17: Spring Constant for Spring $t^{\tau}_{j_k j_{k+1}}$

we have:

$$k^{\tau}_{j_k j_{k+1}} = \begin{cases} k^{\infty} & \text{if } \; t^{\tau}_{j_k j_{k+1}} < t^{\tau min}_{j_k j_{k+1}} \\ k^{\tau}_{j_k j_{k+1}} & \text{otherwise} \end{cases} \tag{6.67}$$

6.9.2 Revisiting Kwa's Negotiation Procedure

In Kwa's negotiation model, the whole scheduling problem is solved in an incremental manner: a new local schedule will be brought into consideration only after the previous local schedules that have already been brought into consideration have been made conflict-free. RTokens reserved for these conflict-free schedules are stored in *RT-DB* (see Appendix B). *RT-DB* stores RTokens in a well-sorted manner: RTokens reserved at one station are stored in the same list and RTokens in the same list are in an temporal order. In this way, retrieving RTokens from *RT-DB* can be very efficient. Suppose, agent j has an initial resource allocation (which is based on for its local schedule as shown in Figure 6.18. Let us call RTokens in *RT-DB* *old* RTokens, and RTokens for agent j *new* RTokens. These old RTokens' relationship have already been built in the past. For each old RToken, namely rt_{i_l}, it generally has a right hand side RToken, $rt^R_{i_l}$, and a left hand side RToken, $rt^L_{i_l}$. They have the following relations:

$$rt^L_{i_l} \| rt_{i_l} \| rt^R_{i_l}$$

So, they do not overlap each other — they are conflict-free. Conflicts, if initially existing, have already been resolved through negotiation between agents that reserved the

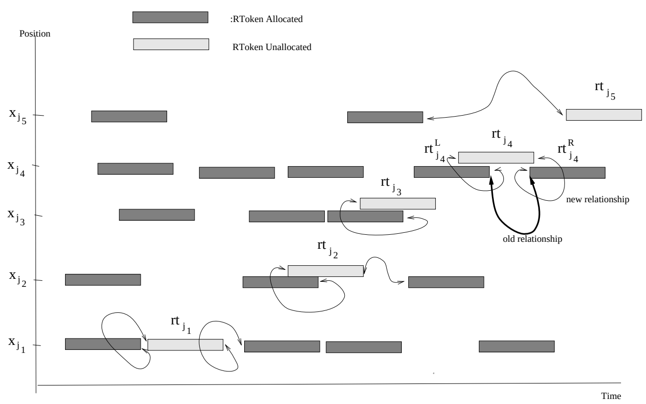


Figure 6.18: Allocating Resources

RTokens. So, all old RTokens have been connected as a conflict-free network. We can visualise that there is also a spring network in which all spring units for RTokens are connected in the same order as this conflict-free network.

Some of these old relationships will be broken when the schedule for agent j is considered. For example, in Figure 6.18, rt_{j_4} is initially in conflict with two old RTokens, rt^L_{j4} and $rt^R_{j_k}$. Suppose, agent j wishes to allocate rt^L_{j4} between the two old RTokens, then the following procedure must be taken:

1. it has to inform the two agents, namely agent a and b, which reserved respectively rt^L_{j4} and $rt^R_{j_k}$, of that that the old relationship:

$$rt^L_{j_k} \| rt^R_{j_k} \quad (k = 4)$$

   is disconnected,

2. it will then try to build a new relationship with agent a,

$$rt^L_{j_k} \| rt_{j_k} \quad (k = 4)$$

   by resolving their conflict through negotiation,

3. if the overlap between rt^L_{j4} and rt_{j_4} has been successfully resolved, then it will try to build a new relationship with agent b,

$$rt_{j_k} \| rt^R_{j_k} \quad (k = 4)$$

by resolving their conflict through negotiation,

4. if the (3) and (4) are all successfully gone through, then rt_{j_4} is successfully allocated, and it now becomes an old RToken.

Based on these considerations, during the process in which agent j is trying connecting its RTokens with these old RTokens in RT-DB, each old RToken rt_{i_l} will generally have the following relationship:

$$rt_{i_l}^L \| rt_{i_l} \| rt_{i_l}^R$$

However, here it is possible that $rt_{i_l}^L$ or $rt_{i_l}^R$ may be a RToken for agent j newly successfully connected into RT-DB.
RTokens for agent j will fall into the following two situations:

1. RTokens that have been successfully connected into RT-DB will can be treated as old RTokens and hence
$$rt_{j_k}^L \| rt_{j_k} \| rt_{j_k}^R$$
will hold for them.

2. RTokens that have not been considered could be treated as free RTokens, i.e., there do not exist $rt_{j_k}^L$ and $rt_{j_k}^R$ constraining rt_{j_k}'s shifting. In this case, we can denote $rt_{j_k}^L = \phi$ and $rt_{j_k}^R = \phi$, and then
$$rt_{j_k}^L \| rt_{j_k} \| rt_{j_k}^R$$
will still hold.

So, all in all, for any RToken rt_{i_l}, no matter whether it is old or new, it generally have

$$rt_{i_l}^L \| rt_{i_l} \| rt_{i_l}^R$$

In this way, we now consider, how agent j is to connect a specific RToken, for example rt_{j_4} in Figure 6.18 into the RT-DB. Suppose some of its RTokens have already been connected into RT-DB, and also suppose that agent j will first resolve conflict between rt_{j_4} and $rt_{j_4}^L$ then between rt_{j_4} and $rt_{j_4}^R$ by:

1. disconnecting rt_{j4}^{L} and rt_{j4}^{R} by informing agent b (which reserves rt_{j4}^{R}) of that it will temporally have no left hand side RToken relationship, i.e.

$$\phi \| rt_{j_k}^{R} \quad (k = 4)$$

For agent j, rt_{j_4} will have right hand RToken relationship, i.e.,

$$rt_{j_k} \| \phi \quad (k = 4)$$

$rt_{j_4}^{L}$ and rt_{j_4} are currently overlap, i.e.,

$$rt_{j_4}^{L} \rightleftharpoons rt_{j_4}$$

suppose, they overlap by T, then a new relationship

$$rt_{j_4}^{L} \| rt_{j_k} \quad (k = 4)$$

can be built if differential T can be resolved. This conflict can be resolved by agent a taking up T_{a_u} and and agent j taking up T_{j_k} where

$$T_{a_u} + T_{j_k} = T \quad (k = 4)$$

2. Any other RTokens' relationships will not be affected.

In this way, we can see that each time the connecting process focuses on only one conflict between agent j and another agent (e.g., agent a), and all the rest of RTokens are treated as if they were all conflict-free even though some of them for agent j are having conflicts with those in RT-DB. The above discussions help us to understand the following negotiation modules which are equations for calculating negotiation strategies.

6.9.3 Negotiation Modules

Viewing Resource Conflict as CIP Problem

Suppose rt_{a_u} (an old RToken) and rt_{j_k} (a new RToken) has an overlap T, then this conflict can be resolved by agent a taking up T_{a_u} and agent j taking up T_{j_k}, so we will have a strategy set:

$$\Sigma = \{(T_{a_u}, T_{j_k}) \mid T_{a_u} + T_{j_k} = T\} \tag{6.68}$$

the common sense principle for the negotiation, according to Theorem 6.3, is that (suppose the negotiation objective is $rt_{a_u} \| rt_{j_k}$),

$$\overrightarrow{F}^{tol\beta 2}_{a_u} + \overrightarrow{F}^{tol\alpha 1}_{j_k} = 0 \tag{6.69}$$

or

$$\overrightarrow{F}^{tol\beta 2}_{a_u} = \overleftarrow{F}^{tol\alpha 1}_{j_k} \tag{6.70}$$

That is the common sense principle for negotiation is that *the restoration forces,* $\overrightarrow{F}^{tol\beta 2}_{a_u}$ *and* $\overleftarrow{F}^{tol\alpha 1}_{j_k}$ *must be equal.* This principle is illustrated in Figure 6.19. where the

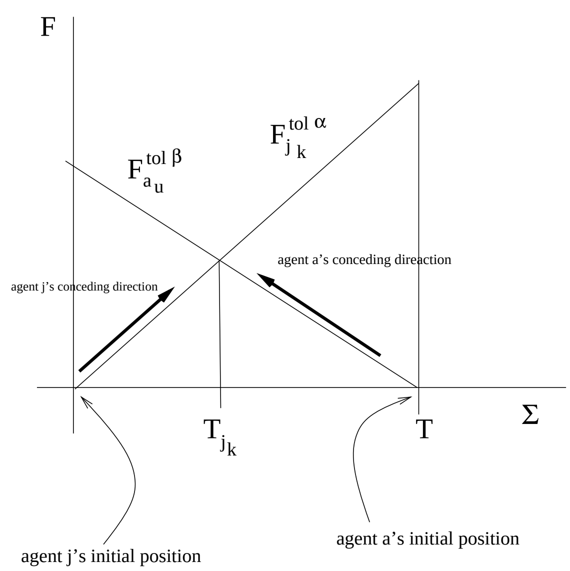


Figure 6.19: The Common Sense Principle

horizontal axis is for Σ which is represented by a parameter T_{j_k} which is the amount of resources agent j concedes (hence, agent a concedes $T - T_{j_k}$. When the two lines intersect each other, then an agreement should be made.

We now can construct a CIP situation:

$$< \overbrace{\{a, j\}}^{\mathcal{A}}, \Sigma, \overbrace{< e_a, e_j >}^{\pi}, \overbrace{< \overrightarrow{F}^{tol\beta 2}_{a_u}, \overleftarrow{F}^{tol\alpha 1}_{j_k} >}^{L}, \overbrace{\overrightarrow{F}^{tol\beta 2}_{a_u} = \overleftarrow{F}^{tol\alpha 1}_{j_k}}^{\text{PRINCIPLE}} > \tag{6.71}$$

Negotiation Modules

How much will agent j concede? What to concede? This paragraph will solve this problem. Suppose agent j now has already conceded $T_{j_k}^n$ and the force is $\overleftarrow{F}_{j_k}^{tol\alpha n}$, but still can not meet the common sense principle, then it try to concede further by $\Delta T_{j_k}^{n+1}$. Then, through Equation 6.42, we can calculate $\Delta t_{j_l}^{\alpha(n+1)}$, $\Delta tol_{j_l}^{\alpha(n+1)}$, $\Delta tol_{j_l}^{\beta(n+1)}$, $\Delta t_{j_l j_{l+1}}^{\tau(n+1)}$ for $l = 1, 2, ..., k, ..., N_j$. When then have

$$\left.\begin{array}{l} \overleftarrow{F}_{j_k}^{tol\alpha(n+1)} = \overleftarrow{F}_{j_k}^{tol\alpha n} - k_{j_k}^{tol\alpha} \Delta tol_{j_k}^{\alpha(n+1)} \\ T_{j_k}^{n+1} = T_{j_k}^n + \Delta T_{j_k}^{n+1} \end{array}\right\} \tag{6.72}$$

Note that $\mathbf{T_j}^1$ and $\mathbf{T_j}^2$ is given by:

$$\left.\begin{array}{l} \mathbf{T_j^1} = \begin{bmatrix} 0 \\ 0 \\ ... \\ 0 \\ \Delta T_{j_k} \\ 0 \\ ... \\ 0 \end{bmatrix}_{N_j \times 1} \\ \mathbf{T_j^2} = \begin{bmatrix} 0 \\ 0 \\ ... \\ 0 \end{bmatrix}_{N_j \times 1} \end{array}\right\} \tag{6.73}$$

Since only rt_{j_k} will concede some resources on its left hand side. a_{j_l} is given by Equation 6.49, i.e,

$$\left.\begin{array}{l} a_{j_l} = k_{j_{l-1} j_l}^{\tau} + k_{j_l}^{tol\alpha} + k_{j_l}^{\alpha} + k_{j_l}^{tol\beta} + k_{j_l j_{l+1}}^{\tau} \\ (l = 1, 2, ..., N_j) \\ k_{j_0 j_1}^{\tau} = k_{j_N j_{N+1}}^{\tau} = 0 \end{array}\right\} \tag{6.74}$$

where

$$k_{j_l}^{tol\alpha} = \begin{cases} 0 & \text{if } \phi \| rt_{j_l} \text{ or } rt_{j_l}^L ||| rt_{j_l} \\ k_{j_l}^{tol\alpha} & \textit{(see Equation 6.66)} \text{ otherwise} \end{cases} \tag{6.75}$$

$k_{j_l}^{tol\alpha}$ is $zero$ (i.e., $k_{j_l}^{tol\alpha}$ is omitted from a_{j_l}) when there is no left hand RToken (i.e., $rt_{j_l}^L = \phi$), or when the left hand RToken separates from $k_{j_l}^{tol\alpha}$ (i.e., $rt_{j_l}^L ||| rt_{j_l}$). This is because, in these cases, there is no external force acting upon $k_{j_l}^{tol\alpha}$ on its left side. So, $\overrightarrow{F}_{j_k}^{tol\alpha 2}$ $(= 0)$ can be omitted from Equation 6.30. $k_{j_l}^{tol\beta}$ can be similarly delivered (But we will not list it here for space reasons). To summarise the discussion in this paragraph: agent j can use the recurrence formula, Equation 6.72, to calculate T_{j_k}

and $\overleftarrow{F}_{j_k}$ until $\overleftarrow{F}_{j_k}$ reaches to the point where $\overleftarrow{F}_{j_k}$ intersects $\overrightarrow{F}_{a_u}$. At this point, the calculated T_{j_k} is the amount to be conceded by agent i.

Internally, how does agent j modify its previous assignments? This is easily answered by the following recurrence formula:

$$\left.\begin{array}{l} t_{j_l}^{\alpha(n+1)} = t_{j_l}^{\alpha(n)} + \Delta t_{j_l}^{\alpha(n+1)} \\ tol_{j_l}^{\alpha(n+1)} = tol_{j_l}^{\alpha(n)} + \Delta tol_{j_l}^{\alpha(n+1)} \\ tol_{j_l}^{\beta(n+1)} = tol_{j_l}^{\beta(n)} + \Delta tol_{j_l}^{\beta(n+1)} \\ t_{j_l j_{l+1}}^{\tau(n+1)} = t_{j_l j_{l+1}}^{\tau(n)} + \Delta t_{j_l j_{l+1}}^{\tau(n+1)} \\ (l = 1, 2, ..., k, ..., N_j) \end{array}\right\} \tag{6.76}$$

Nested Negotiation Modules

A nested negotiation process will be applied when an agent, during its negotiating with another agent, monitored a new conflict with a third agent. The nested negotiation process will resolve the conflict with that the third agent. In our spring model, the new conflict is represented by a spring, (i.e, $tol_{j_l}^{\alpha}$, or $tol_{j_l}^{\beta}$) being over-compressed, or say that $\overrightarrow{F}_{j_l}^{tol\alpha}$ or $\overrightarrow{F}_{j_l}^{tol\beta}$ becomes very large. The negotiation is very similar to the main negotiation discussed in last paragraph, except that (assume that the negotiation is between agent j and agent a — two arbitrary agents).

$$T_{j_l} + T_{a_u} = 0 \tag{6.77}$$

That is to say that agent j and agent a do not have resource conflicts prior to negotiation. However, since agent j has been over-constrained, so it will ask for some resources from agent a. The amount of resources T_{j_l} agent j asked for is equivalent to $|T_{a_u}|$ — the amount of resources conceded from agent a. Figure 6.20 illustrates in nested negotiation agent i and agent j's concession.

6.10 Related Work

6.10.1 Operations Research

Scheduling or resource allocation is a problem that has been examined in Operation Research literature since the early Fifties [Conway *et al.* 67]. Methods and theories

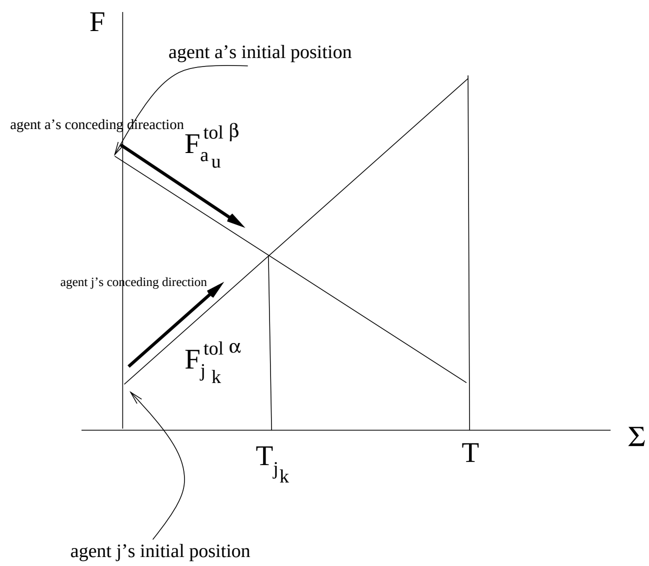


Figure 6.20: Conceding in Nested Negotiation

developed are based on strong mathematical theories. For a given objective, a typical scheduling task falls into the mathematical class of "NP-hard" problems. For this reason, the main focus of Operations Research has been on finding fast algorithms (e.g., the Hungarian Method, the Branch and Bound Method, the Dynamic Programming Method, etc.) for solving mathematical formulas. Our spring model, to some extent, is similar to nonlinear programming methods [Eiselt & vonFrajer 77] which concerns finding fast algorithms for solving problems with non-linear objective function. For example, the Lagrange Method formulates the Lagrange-function as the sum of objective function and the weighted constraints. Each partial derivative of the function is set equal to zero resulting in a unique system of equations (simultaneous equations as we interpreted). In this way, all variables' elasticity has been transformed into the simultaneous equations. However, Operations Research methods generally have the following problem [Dorn & Slany 94]: the algorithms are too complex for real world applications. This problem is solved in our model by dividing the whole problem into many subproblems which are assigned to a set of cooperative agents.

6.10.2 Constraint-directed Scheduling

The ISIS job-shop scheduling system was the first attempt to formulate and operationalize the view of scheduling as a heuristic, constraint-directed activity [Fox 87] [Fox 94]. ISIS emphasised a complete representation framework that recognised the conflicting and negotiable nature of many of these constraints. A representation of preference (i.e., relaxable) constraints was defined to encode knowledge relating to various factory objectives and operating preferences, including their relative importance, possible relaxations of preferred choices, the utility of each alternative, and the types of decisions that the constraint impacts. This knowledge about preferences was embedded in a larger relational framework for modelling the entities and physical constraints of the production environment. Constraints were classified into non-relaxable (hard constraints) and relaxable (soft constraints). What is of our most interest is his concept of the elasticity of the relaxation of a constraint which describes the difference between alternative relaxations. The elasticity is modelled in the system in production rules. Importance of a constraint was introduced to emphasize the relative influence on other constraints' relaxation.

Our spring model has some common features with ISIS:

- They are both representational model or system for representing constraints — hard constraints and soft constraints.
- They both emphasize the elasticity of a soft constraint. We represent the elasticity by a spring unit, while Fox represented the elasticity through production rules.
- We use the representational model to guide negotiation. Fox used his representational model to direct the search for solution.

6.10.3 Train Scheduling

Fukumori's work on train scheduling [Fukumori 80] is very relevent to our spring model. He proposed an algorithm that uses a range-constriction search technique to schedule the timing and pass-through relations of trains smoothly and efficiently. The basic ideas of the algorithm are:

- Train schedules are represented by a "belt" (which is similar to Kwa's interval-based scheduling) instead of a "line" (which is similar to point-based schedulinng).
- This representation potentially allows a train's schedule to be "bent": if the train's (e.g. an express's) schedule is in conflict with other trains' schedules, the conflict could be solved by letting the express slow down to avoid the conflict. This is similar to compressing the $t^{\tau}_{j_k j_{k+1}}$ in our spring model.
- A train's belt could be shrunk when other trains' belts overlap with it. This is similar to reducing tolerant time in our spring model.

Fukumori's algorithm is based on human experience. It was not systematically developed.

6.10.4 Neural Network: The Elastic Net Approach to TSP

Our spring model is an analogical approach. We may find some similarity to the elastic net approach to Travelling Salesman Problem [Durbin *et al.* 89] [Simmen 91]. A elastic net can be visualised as a rule for deforming an imaginary elastic band placed in the city plane by attractive, distance-dependent forces from the cities and by elastic forces within the band itself. A scale parameter controls the effective range of the city forces. It is initially set high, then gradually reduced. In practice, the net is modeled by a finite number of points ("beads") and the algorithm reduces to an iterative procedure for updating the bead positions. Our model and the elastic model have some similarities. For example, the two models are anological to elastic properties. They yield solutions when the elastic entities (springs or elatic net) preserve the lowest energies. Solutions are gained under the effect of forces from the elatic entities.

6.10.5 Various Spring Models

There were various spring models used in other fields. Here we just give some examples. In [Arbib 72], Arbib proposed a spring model for representing muscles of human hand.

In [Liritzis *et al.* 95], a spring model — spring-block model — is applied for the cross-correlation analysis of seismic time-series to clarify the complicated space-time pattern of the world wide mosaic of tectonic plates interaction. In seismicity study, similar model to the spring-block model is the spring-mass model proposed in [**?**].

Closely related to our spring model we found is the combination of a certain number of spring models in the finite element methods to represent some properties interested by researchers and engineers. For example, [Yue *et al.* 97] presents a new flexible rotor beam element to study the dynamic behaviour of manipulators with flexible links and joints. Both link and joint flexibility are incorporated together by using the element model which is the combination of a finite element model for links and a torsional spring model for joints. The coupling terms of link and joint flexibility are considered in the dynamic equations of the minipulator.

In [Okamoto *et al.* 95], a new algorithm for particle tracking, called the spring model technique, is proposed to anlayze image data. The algorithm is based on pattern matching of particle clusters between the first and second image. A particle cluster is composed of particles which are assumed to be connected by invisible elastic springs. Depending on the deformation of the cluster patern (i.e., the particle positions), the invisible springs have some forces. The smallest force pattern in the second image is the most probable patten match to the correspondent original pattern in the first image.

The idea of text spacing technique used in [Knuth 86] can be expressed as a spring model as Knuth described: TeX makes complicated pages by starting with simple individual characters and putting them tegother in larger units, and putting these together in still larger units, and so on. The terms used to describe such page construction are boxes and glue. During spacing these boxes, they should be stretched or compressed. The proporties of these boxes can be expressed by springs.

All applications of spring models have a common feature: they use spring models to represent some properties that can be described by forces, elasticity, flexibility, etc. In this aspect, our spring model serves the same purposes as other spring models. However, there is a major distinction between our model and other spring models we know so far: when constructing a spring network by connecting basic spring elements, much work is involved in computing how they should be connected (e.g, while connecting two RTokens, we must first compute which RToken should be on the right and which should be on the left), but this does not seems to be a problem in these other models where how spring elements should be connected is known.

6.10.6 Job Shop Scheduling Problem (JSSP)

Taking stations and junctions as machines, we can see that the AGVSP is related to the JSSP which involves allocating start-times to activities (or operations). A JSSP can be described by four aspects [Conway *et al.* 67]: (1) m machines, (2) N jobs with each of them being a set of operations with each of them being specified by a machine where the operation is processed and a processing-time needed for the operation being processed; (3) constraints that restrict the manner in which allocations can be made and (4) the criteria by which a schedule will be evaluated. Efforts have been made by researchers in the last several decades to seek efficient searching algorithms of solving complex JSSPs with large solution spaces. Some algorithms are based on enumerative search (e.g., Bound and Branch Search [Balas 69] [Carlier & Pinson 89], etc.) — they aim to obtain the optimal solution by dropping non-optimal solutions. However, they often prove to be ineffective when seaching spaces are huge. Some algorithms are based heuristic search (e.g., Tabu Search [Glover & Laguna 93], Simulated Annealing Search [Shen *et al.* 94], genetic algorithms [Fang *et al.* 93] [Corne *et al.* 93], etc.). Heuristic search algorithms aim to obtain sub-optimal solutions. Solutions are achieved by gradual improvement on previous solutions obtained and the final solutions will be given when there is no improvement possible or when searching time is due. For more detailed review of JSSP algorithms, the reader is recommended to refer [Fang 94].
Although the AGVSP has many features similar to the JSSP, it has its own unique features as well. These unique features are listed below:

- *Strictly ordered activities* — for each AGV, its planned route prior to the AGVSP determines the order in which the AGV moves from one station (or junction) to another. So, the order of activities are fixed and cannot be changed during problem solving. This is not the case in many JSSPs where the order of activities may be changed.

- *Strong links between activities* — for each AGV, two consecutive activities are linked by the travelling time between a station for the first activity and another for the second. The constraints on the travelling time are critical: the shortest travelling time (e.g., $t^{\tau min}_{j_k j_{k+1}}$) is limited by the AGV's speed, and longer travelling times will affect the feasibility of the overall system. By feasibility we mean whether a planning system can organise all the movements of the AGVs required by a factory. Obviously, the longer the travelling times, the fewer movements can be organised.

- *Strong preferences over solutions* — the parts should be delivered to each work station at the time when workers (or robots) need them, arriving earlier may cause local inventory problems, while arriving later will make less efficient use of resources — workers, machines, robots, etc; ready products should be delivered to custermers in time to satisfy them. In other words, Just-IN-Time (JIT) requirement is important. Classical approaches to the JSSP are aimed at shortening the makespan — the time period between the time when the first job is started and the time when the last job is finished. The makespan in the case of JIT as in the AGVSP is not as strongly desired as would be in the case of other JSSPs.

- *Tolerance allocations* — to reduce the need for frequent rescheduling, in the AGVSP we are concerned, AGVs' activities are allocated some redundant resources so that they can be made more robust. Classical JSSP algorithms do not consider in this dimension.

Due to these distinguishing requirements, there are difficulties in directly applying the existing JSSP approaches to the AGVSP.

6.11 Summary

In this chapter, we systematically developed a novel spring model to represent an agent's local scheduling. The spring model has several purposes:

1. It is a representational model — it represents various constraints.

2. It is an optimal local scheduling model — it can suggest an optimal solution to a local AGV scheduling problem given a certain amount of resources.

3. It is a negotiating model — it can suggest a negotiation strategy (i.e., how much to concede) for an agent, and suggest how to concede so that a local scheduling problem can have an optimal solution. The negotiation is based on the common sense principle that the force and reacting force between two RTokens should be equal.

Our model solves the important issues that Kwa's model was unable to solve in the following way:

- **What negotiating technique does the conceding agent use?**
  Negotiating techniques are generated based on the spring model of a local schedule. Each time, the conceding agent will use combined techniques which is similar to Kwa's negotiating skill. However, instead of shifting and yielding resources allocated to one node (i.e. rt_{j_k} for a particular k), our model is to shift and yield resources allocated to all nodes (i.e. $k = 1, 2, ...$). In other words, each time agent j takes up some differential of overlap, its internal resource allocations are all modified. This modification is made under a prerequisite that there is no third party being affected.

- **How much differential does the conceding agent take up?**
  This isssue is solved by the common sense principle in our negotiation model: the conceding agent must take up an amount of differential by which the two negotiating agents' restoration forces are equal (see Figure 6.19).

- **What do negotiating agents communicate with each other**
  They communicate proposals during negotiation. A proposal comprises the amount of differential the conceding agent wishes to take up and the state of it provided the proposal is accepted by its opponent. The state is represented by the restoration force of its spring-based local scheduling model.

- **What is the belief of a conceding agent about its negotiating opponent?**
  The belief of the conceding agent about its negotiating opponent is the predicted

restoration force of that opponent provided that the proposal from the conceding agent is accepted. So, our model allows the conceding agent to reason about negotiating strategies based on its belief about its opponent's state.

- **What does it mean that "an agent is selfish"?** As we argued in Chapter 5, terms, such as "selfishness", "benevolence", "altruism", etc., are well understood by us in human activities. However, without formal definitions of them, we are unable to model them into computational models. In our model, the altruistic agent assumption is not made, nor is the selfish agent assumption. We assume that agents are rational and benevolent and rationality and benevolence are formally defined. The rational agent assumption requires each agent to seek the optimal solution to its local problem and the benevolent agent assumption requires each agent to help other agents under the condition that its own interest is not affected during the help.

In conclusion, our spring-based negotiation model possesses properties that are required for solving the AGVSP in a multi-agent system.

Chapter 7

Implementation and Experiments

7.1 Introduction

In this chapter, we will first introduce our implementation of the spring-based negotiation model. Then we will give some experimental results to compare our model with Kwa's model to show whether our model is better than Kwa's.

7.2 Implementation of Spring-based Negotiation Model

7.2.1 System Structure: A Small Multiagent System

The system architecture which outlined our implementation of the spring-based negotiation is characterised as a small multiagent system (Figure 7.1). We view an agent here as an entity consisting of databases plus a set of procedures which manipulate these databases.

Two Types of Agents

The system is a multi-agent system consisting of two types of agents. The first type consists of only one agent called the manager which is responsible for distributing jobs to proper agents which then undertake the assigned jobs. These agents being assigned jobs to undertake fall into the second type. We still call each of them an agent or an AGV. A job here is a specification of requirements (soft constraints and hard constraints) for a local AGV schedule. Each agent's task is then to assign resources

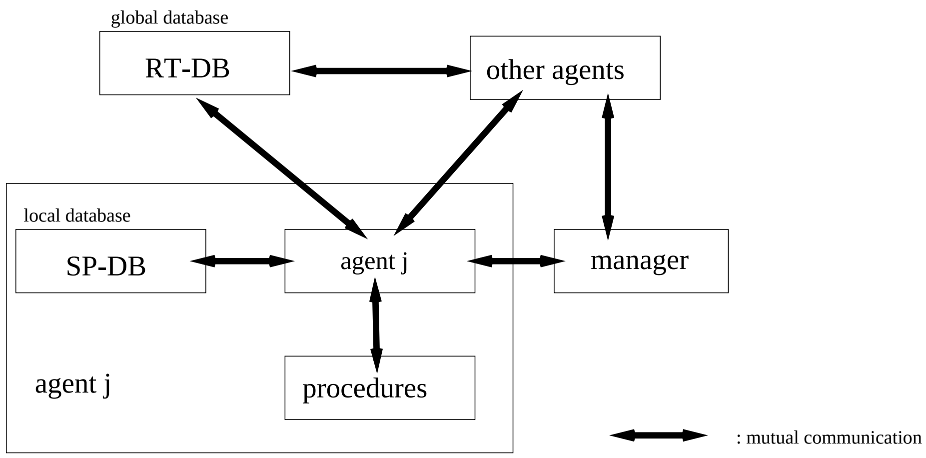


Figure 7.1: System Structure

for the schedule according to the requirements. It initially assigns resources to the schedule based only on its intra-agent constraints (see Chapter 6), and then it checks whether the initial assignments are in conflict with those of other agents' schedules which have been made before. If there are conflicts, then the agent will initialise a negotiation process to resolve the conflicts. The initiator is also responsible for controlling the whole negotiation process, e.g., when the negotiation will terminate, whether the negotiation is successful or not. If the negotiation is successful, then it informs all other agents involved in the negotiation that the results of the negotiation should be honoured; these agents therefore modify their schedules according to the results. If the negotiation is not successful, then the initiator will also inform those agents involved that the negotiation is not successful and that they should keep their schedules as they were before the negotiation. So, it is important for each agent to keep all its schedule made before each round of negotiation so that backtracking is possible.

Databases

There are two types of databases: RT-DB and SP-DB. The RT-DB is a database mainly for recording RTokens reserved by each agents. This database can be used by every agent: (1) to check whether its own RTokens are in conflict with those of other agents, (2) to insert a RToken into the RT-DB if it is conflict-free with all RTokens in the RT-DB. The structure of the RT-DB is based on the one given in Appendix B. The SP-DB is a database for storing a spring model: the initial state of the spring model, the intermediate state of the spring model, and state of the spring model during a negotiation. We can visualise it as in the following structure:

$$(AGV\ SM^2\ SM^1\ SM^0)$$

where SM^0 is the initial state of a local schedule made by the respective agent based only on its intra-agent requirement. It is then the optimal local schedule. SM^1 is the intermediate state of the schedule. SM^2 is the state of the spring model during a negotiation. SM^1 and SM^2 have the following relationship:

- At the starting time of a negotiation process initialised by the agent itself or by any other agent, $SM^2 = SM^1$;
- During negotiation, SM^1 remains unchanged. But SM^2 will be modified according to conceding proposals it may wish to make.
- At the terminating time of the negotiation, if the negotiation is successful then SM^1 will be changed to the new state the same as SM^2, i.e., $SM^1 \leftarrow SM^2$. If the negotiation is not successful, then SM^2 should be undone and changed back to its state prior to the negotiation, i.e., $SM^2 \leftarrow SM^1$.

This data structure makes it possible for an agent to do backtracking search should a negotiation fail.

Communication Between Agents

When agents negotiate over resource allocations, they exchange information. The content of information, according to Definition 4.1 is:

$$(\sigma, l)$$

where σ is a cooperative strategy proposed by an agent, and l is the state of the agent when the strategy proposed is accepted. In our spring-based negotiation, σ is an interval (T) which is conceded willingly by the proposing agent and l is the force of the agent under this proposal. So, a proposal is:

$$(T, F)$$

In order to let the opponent to predict the proposing agent's state F, we modify the form of the proposal to:

$$(T, F, \frac{dF}{dT})$$

where $\frac{dF}{dT}$ represents the slope of F with respect to T. With the additional information, the opponent is able to predict the proposing agent's state and make a counter-proposal more accurately (i.e., closer to the requirement given by a common sense principle) (See Section 7.2.2 and Section 7.2.2 for details). This could make negotiation more efficient.

7.2.2 Predictive Concession vs Qualitative Concession

When two agents, namely agent i and agent j, have a conflict that their rt_{i_l} and rt_{j_k} (they are two RTokens at the same stations reserved by agent i and agent j respectively) overlap by T, then a negotiation procedure

$$\textbf{negotiate}(j, i, T)$$

will be called to resolved the conflict. The basic framework of the procedure is based on Algorithm 4.3. However, in spring-based negotiation where each time there are only two negotiators, regarding how the two negotiators make proposals, we will have the following simplied version of **negotiate**(j, i, T). We call this version the *predictive version* (see Algorithm 7.2) to distinguish the *qualitative version* (see Algorithm 7.1). The two versions differ in ways of proposing offers. In the predictive version, each proposing agent will make a proposal based on its knowledge about itself (what is its state if the proposal is accepted) and its prediction about its opponent's state under that proposal. In the qualitative version, each proposing agent makes proposals only according to its current state and its opponent's current state.

Qualitative Concession

The first style of concession is what we called "Qualitative concession". Algorithm 7.1 shows a qualitative concession procedure.

Algorithm 7.1

procedure negotiate(j, i, T)

1. $T_j = \Delta T$;; *agent* j *concedes* T_j

2. $T_i = \Delta T$;; *agent* i *concedes* T_i

3. *Computing* F_j *and* F_i

4. if $T_j + T_i \leq T$ **then exit return** (T_j, T_i) **endif**

;; *the conflict is resolved*

5. if $F_j = F_i = \infty$ **then exit return** ϕ **endif**

;; *none of the agents is able to concede anymore.*

6. if $F_j \leq F_i$ **then**

$T_j = T_j + \Delta T$

$F_j = F_j(T_j)$

goto step *4*

endif

7. $T_i = T_i + \Delta T$

$F_i = F_i(T_i)$

8. goto step *4*

Predictive Concession

In the qualitative concession algorithm, a conceding agent is one of the two agents which has smaller force. But how much the force is smaller than the other's is not important. Each time, each conceding agent concedes a fixed amount of resource. Although qualitative concession is very straight forward and easy to apply, it still suffers some serious problems. For example, the fixed amount ΔT is very difficult to determine: (1) A large ΔT will speed up the negotiation process but sometimes will cause a conceding agent to over-concede and hence affect the agent's rationality assumption; and (2) a small can limit the damage to agents' rationality, but will increase negotiation time. A "predictive" concession algorithm, Algorithm 7.2 and Algorithm 7.3 can solve this problem.

Algorithm 7.2

procedure negotiate(j, i, T)

1. $p_j \leftarrow$ **initial-concession**(j, T)

2. $\sigma_j = car(p_j)$; $F_j =$ **cadr**(p_j)

3. $p_i \leftarrow$ **p-concession**(i, σ_j, F_j, T)

4. $\sigma_i = \mathbf{car}(p_i)$

5. $F_i = \mathbf{cadr}(p_i)$

6. if $\sigma_i = \sigma_j$ **exit return** σ_j **endif**

;; since both agents agree on the same strategy, it then becomes the agreement.

7. if $F_j = F_i = \infty$ **then exit return** ϕ

8. $p_j \leftarrow$ **p-concession**(j, σ_i, F_i, T)

;; negotiation continues and now is agent j 's turn of concession.

9. $\sigma_j = \mathbf{car}(p_j)$

10. $F_j = \mathbf{cadr}(p_j)$

11. if $\sigma_j = \sigma_i$ **exit return** σ_i **endif**

12. if $F_j = F_i = \infty$ **then exit return** ϕ **endif**

;; since none of the agents is able to concede, the negotiation terminates with no agreement.

13. goto step 3 *;; negotiation continues*

Algorithm 7.3

procedure p-concession(j, σ_i, F_i, T)

1. $F_j = F_j(\sigma_i)$

2. if $F_j \leq F_i$ **then exit return** $(\sigma_i,\ F_j)$ **endif**

;; agent j agrees on σ_i proposed by agent i.

3. *Predict* $F_i(\sigma)$

4. *Find a* σ_j *such that* $F_j(\sigma_j) = F_i(\sigma_j)$

5. return (σ_j, F_j)

;; agent j makes a new proposal.

In the predictive algorithm, each conceding agent chooses a cooperative strategy based on its prediction about the force of the other agent. It will select a strategy that satisfies, according to its prediction, the common sense principle, i.e., the forces of the two agents will be equal. The process is iterative because the prediction can not be 100% precise due to that the predicting agent does not have complete knowledge about its opponent. Despite this, this predictive model is expected to be much faster than the qualitative model. In our spring-based negotiation, we will use the qualitative version for nested negotiation procedures and predictive version for main negotiation procedure. This is because (1) nested negotiation normally requires each conflicting agent to make small modifications to their resource allocations (one agent is gaining some resources while the other is losing the same amount of resources), therefore negotiation efficiency is not essential and qualitative concession is simple and acceptable, and (2) in the main negotiation procedure, two conflicting agents often have a large amount of overlapping resource allocation, therefore negotiation speed is very important, and so we use predictive concession.

Figure 7.2, Figure 7.3 and Figure 7.4 illustrated how the predictive concession algorthim works.

Initially, agent j and agent i reserve RTokens rt_j and rt_i respectively. They overlap by T as shown in Figure. Heuristically, the conflict can be resolved by agent j shifting leftwards and/or yielding rear tolerance, and agent i shifting rightwards and/or yielding front tolerance. As we already know that the optimal state of global scheduling is that

$$|F_{j_k}^{tol\beta}(\sigma)| = |F_{i_l}^{tol\alpha}(\sigma)| \quad (\sigma \in \Sigma) \tag{7.1}$$

That is $\sigma(= (T_1, T_2))$ is one that satisfies Equation 7.1. So, we can use this equation as a common sense principle to guide negotiation. And force F is the common language used to communicate during negotiation.

So, each agent (e.g., agent j) will select a negotiation strategy based on the following knowledge:

1. Its knowledge about $F_{j_k}^{tol\beta}(\sigma)$. Since agent j knows its local scheduling spring model completely, therefore it knows $F_{j_k}^{tol\beta}(\sigma)$ completely (see below).

2. Its belief about $F_{i_l}^{tol\alpha}(\sigma)$. Agent j does not have complete knowledge about $F_{i_l}^{tol\alpha}(\sigma)$, it can only predict it based on its incomplete knowledge about agent i. The incomplete knowledge comes from communication between the two agents during negotiation. Assuming $F_{i_l}^{tol\alpha}(\sigma)$ is a linear function will make the prediction easy.

7.2.3 Examples of Spring-based Negotiation

In this section, we give two examples of how resource conflicts in AGV scheduling are resolved by our spring-based negotiation model. The two examples are based on two scenarios used by Kwa (see Example 5.1 for the first scenarios and [Kwa 88a] (pp181-183) for the second scenario).

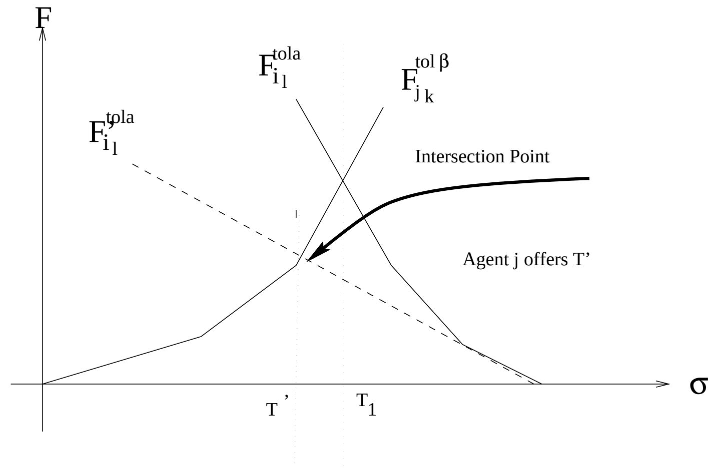


Figure 7.2: The Elements of Spring Models

A Simple Example

The simple example is based on the scenario given in Example 5.1. Below is the trace of negotiation between two agents AGV1 — which is to schedule R1, R2 and R3 — and AGV2, which is to schedule R7, R8 and R9.

```
Scheduling for AGV1.....
The provisional  local schedule for AGV1 is as follow:
(X9 (197.18 257.18))
(X10 (212.50 284.00))
(X11 (250.93 313.93))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; The provisional local schedule is made by the respective;;;
;;; AGV  when  other agents'  resource  allocations  are not;;;
;;; considered.                                             ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
The conflict-free local schedule for AGV1 is as follow:
(X9 (197.18 257.18))
(X10 (212.50 284.00))
(X11 (250.93 313.93))


Scheduling for AGV2.....
The provisional  local schedule for AGV2 is as follow:
(X12 (241.60 312.10))
(X10 (264.50 337.50))
(X13 (290.06 364.56))
Initial state of negotiation between AGV2 and AGV1:
```

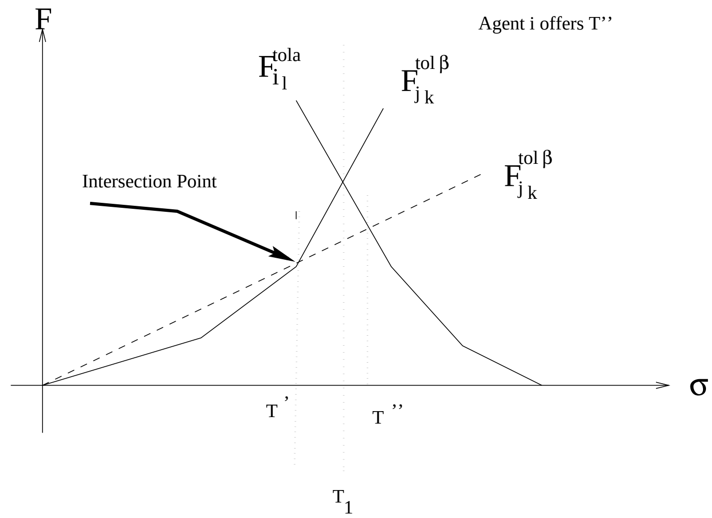


Figure 7.3: The Elements of Spring Models

```
(1) They have reserved RTokens at station X10;
(2) The two Rtokens overlap with a differential of 19.50;
(3) AGV2 will concede on TOLA side;
(4) AGV1 will concede on TOLB side;
(5) AGV2's RToken is (264.50 337.50);
(6) AGV1's RToken is (212.50 284.00);
Now the negotiation starts ....
AGV2 conceded on TOLA by 8

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; "AGV2 conceded on TOLA by 8" means that concession is;;;
;;; made on the front end of the RToken.                 ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AGV2's force changes from 0. to 68.02

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; The force of the conceding agent is calculated by the;;;
;;; conceding agent according to its own spring    model,;;;
;;; hence it is the true force of the agent.             ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

the oppenent's force changes from 102.92 to 60.70

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; The force of the opponent is predicted by the conceding;;;
;;; agent according to its current knowledge about that    ;;;
;;; opponent, therefore it may not be the same as the true ;;;
;;; force of the opponent.                                 ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

AGV2's internal modification is as
follow:
(0. 1.15 0.)
```

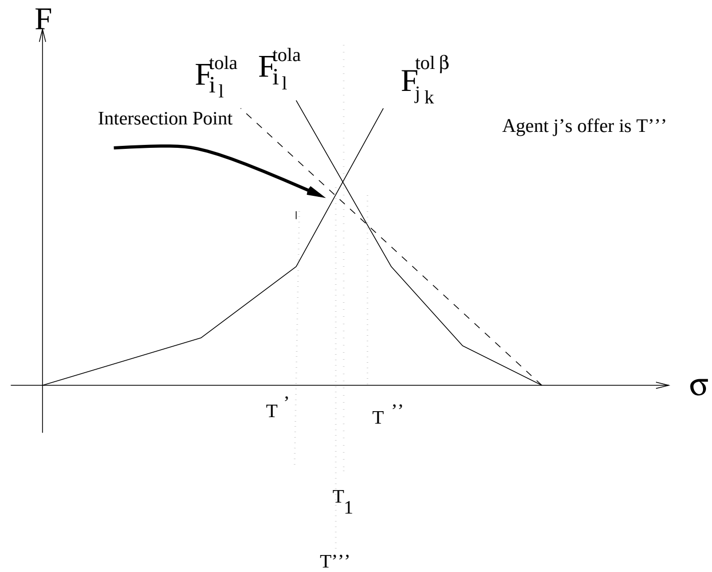


Figure 7.4: The Elements of Spring Models

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; TOLA of the first node of AGV2 remains the same        ;;;
;;; Arrival Time of the first node increased by 1.15       ;;;
;;; TOLB of the first node remains the same                ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(-3.7 4.22 0.)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; The width of TOLA of the second node decreased by 3.7  ;;;
;;; The Arrival Time of the  second node increased by 4.22 ;;;
;;; TOLB of the second node remains the same.               ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(0. 4.18 0.)

AGV2's current RTokens are as follows:
(242.75 313.25)
(272.50 341.72)
(294.25 368.75)

AGV1 conceded on TOLB by 12
AGV1's force changes from 0. to 63.32
the oppenent's force changes from 165.84 to 63.80
AGV1's internal modification is as
follow:
(0. -5.63 0.)
(0. -5.66 -6.33)
(0. -2.26 0.)
AGV1's current RTokens are as follows:
(191.54 251.54)
(206.84 272.00)
(248.66 311.66)

The conflict-free local schedule for AGV2 is as follow:
```

```
(X12 (242.75 313.25))
(X10 (272.50 341.72))
(X13 (294.25 368.75))
```

The comments interspersed in the trace are made to help the reader to understand it.

A Non-trivial Example

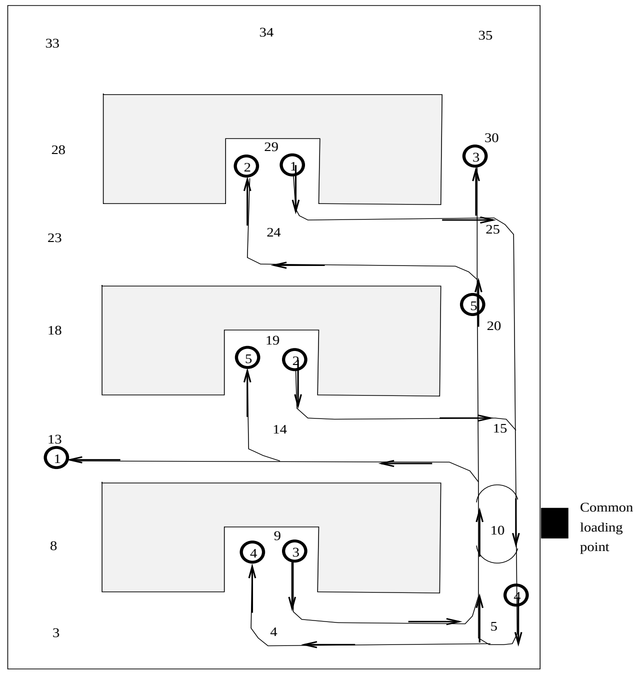
The second example of resource conflict resolution by the spring-based negotiation model is based on a scenario (see Figure 7.5) given by Kwa [Kwa 88a]. There are five AGVs each of which has been planned a route as shown in the figure. Each agent's provisional schedule and final schedule (i.e., a schedule when the whole negotiation process is completed) is illustrated in Appendix D and the negotiation process is illustrated in Appendix C After negotiation, all conflicting allocations are resolved. In the whole negotiation process, there are altogether 25 rounds of communication between negotiators to exchange proposals. Compared with 56 rounds of communication required in a negotiation process to solve the same problem by Kwa's model (in Appendix D in [Kwa 88a]), our model saves significant negotiation times. By examining our negotiation process and Kwa's negotiation process, there are two main distinctions between them that makes our model faster than Kwa's:

1. Our model enables agents to predict their negotiating opponent's state so that they can make proposals closer to the requirements given by the common sense principle and speed up negotiation process. Kwa's model does not have this ability.
2. Our model allows agents to apply combined negotiation techniques while Kwa's model allows agents to apply one technique a time.

7.3 Experiments

Although the spring-based negotiation model we developed has strong theoretical supports, we keep in mind the following three questions all the time:

1. Can our model outperform Kwa's model?



Uncircled numbers are the nodes defining the route map
Bold circled numbers denote the AGVs

Paths of AGVs

AGV1 ... (29 24 25 20 15 10 15 14 13)
AGV2 ... (19 14 15 10 15 20 25 24 29)
AGV3 ... (9 4 5 10 15 20 25 24 29)
AGV4 ... (5 10 5 4 9)
AGV5 ... (20 15 10 15 14 19)

Figure 7.5: Trajectories of Five AGVs

2. Can our model be compared with other existing scheduling approaches?

3. Can our model be used to solve other scheduling problems?

In this section, we report some experimental results we achieved in comparing our model with other models.

7.3.1 Comparing Our Model with Kwa's Model

The first objective of our experiments is to compare our negotiation model with Kwa's negotiation model to see whether our model can solve an AGVSP problem better than

Kwa's does.

Measuring Performance of a Scheduling Method

There are two main aspects to consider for measuring a scheduling method — efficiency, i.e., how fast a complete schedule can be achieved by the method, and quality, i.e., how good the schedule is.
About efficiency, we mainly consider how many communication times are needed during the whole process of negotiation. Obviously, the more communication times are needed, the less efficient the method is.
There are various objective functions to measure the quality of a schedule in the literature of JSSP. The followings are some typical objective function [Conway *et al.* 67]:

1. An objective function for measuring makespan of a schedule, i.e., the maximum completion time. An optimal solution is one that minimises the makespan.

2. Objective functions for measuring flow time. Flow time of a job is defined as the interval between the time the job is ready and the time the job is completed. For example, the weighted flow time objective function and the weighted mean flow time objective function are for measuring flow time. Optimal solutions are those that minimise these flow time objective functions.

3. Objective functions for measuring closeness of the completion time of a job and the due-date of the job. For example, the weighted lateness objective function and the weighted tardiness objective function, etc. are in this category. Optimal solutions are those that minimise these objective functions.

The makespan objective function, which is the most common objective function in the literature of JSSP, is not considered in our experiment. Makespan is the capacity measurement which indicates the shortest time period for finishing a given set of jobs. Our model mainly deals with satisfying various preferences over how each job is done. For example, a job, which could be allocated to be finished by 3:00pm with the current available resource, would be scheduled by our model (and Kwa's model) to finish by 4:00pm if that is the job's preferred finish time. In this case, the makespan of the whole schedule for the AGVSP will certainly be affected. Therefore, the makespan is not of our primary consideration.

About flow time, we consider the travel time of a job, i.e., the interval from the starting time of a job and the finishing time of the job:

$$\tau_j = t^{\beta}_{j_{N_j}} - t^{\alpha}_{j_1} \tag{7.2}$$

where τ_j is the travel time of job j; $t^{\alpha}_{j_1}$ is the starting time (arrival time) at the first station and $t^{\beta}_{j_{N_j}}$ is the completion time (departure time) at the last station. We use the following weighted travel time objective function:

$$\tau_{weight} = \sum_{j}^{N} w_j \tau_j \tag{7.3}$$

where w_j is the weight of job j. The spring constants in our model for a job are directly proportional to the weight of the respective job.

Travel time is closely related to flow time in that they both focus on how much time is needed for completing a job. However, flow time uses the ready time (the earliest start time in our model) as the starting time. So, minimising flow time means scheduling a job to be finished as close to the ready time as possible. This again is not acceptable in the AGVSP. In the AGVSP, a job means that an AGV travels along a route with various operations such as loading, unloading, etc. These operations are normally required to be done at preferred times. So, the job is not necessarily to be started and finished as early as possible. We use the following weighted deviation objective function:

$$ET_{weight} = \sum_{j}^{N} \sum_{k} w_j |et^{o}_{j_k} - et^{o*}_{j_k}| \tag{7.4}$$

Here $et^{o}_{j_k}$ is the departure time of AGV j (job j) at its kth station. Superscript o indicates that at this station there are loading/unloading operations required. This objective function is different from classical JSSP objective functions relating to finishing time which are concerned either with tardiness (how late a job is scheduled to be finished than the due-date) or earliness (how much earlier a job is scheduled to be finished than the due-date). ET_{weight} concerns Just-In-Time (JIT) requirements which, in the AGVSP, are more important than requirements of tardiness or earliness.

The last objective function we considered is related to the robustness of an AGV schedule:

$$R_{weight} = \sum_{j,k} w_j(|\Delta tol^{\alpha}_{j_k}| + |\Delta tol^{\beta}_{j_k}|) \tag{7.5}$$

This objective function measures how much tolerant time has been conceded after negotiation. Obviously, the more the tolerant times are lost, the larger value the objective function will yield. Therefore, a better schedule will return a smaller value from the objective function.

Outline of the Experiments

Our experiments were carried out according to the following procedure:

1. Generating a set of N random jobs;
2. Generating a set of N provisional local schedules according to the job set;
3. Applying the spring-based negotiation model to the set of provisional local schedules until the process is completed.
4. Applying the iterative negotiation model of Kwa's to the same set of provisional schedules until the process is completed.
5. Analysing the results of the both models and comparing them.

About how a local schedule is randomly generated, the interested reader can read Appendix E.

About Setting Negotiation Skill in Kwa's Model

In Kwa's model, a negotiation skill is a list of negotiation techniques and a negotiation technique indicates an amount of resource is conceded and a method (yielding or shifting) for realising the concession. It is fundamentally important that negotiation skills be properly defined. Unfortunately, there are no clear instructions from Kwa's work about what negotiation skill to use in a given situation. For this reason, we did various experiments on choosing approriate negotiation skills. Generally, these experiments tell us that:

1. If a negotiation skill consists of only yielding techniques, then the robustness of the whole AGV schedule will be seriously affected, since most of the redundant resources are conceded during negotiation. Another observation is that the number of local schedules that can not be made conflict-free after negotiation with the yielding only techniques becomes very large when the number of total local schedules increases. In other words, $p_r(n)$, which is the probability of resolving a local schedule which is in conflict with one or more of n local schedules in $RT - DB$ (they are conflict-free), decreases more sharply when n increases than when other negotiation techniques are used (about $p_r(n)$, the interested reader can refer to Appendix E).

2. If a negotiation skill consists of only shifting techniques, more local schedules can be made conflict-free compared with using yielding only techniques. However, although the robustness is not affected (since no redundant resources are yielded), other qualities of the whole AGV schedule become worse: (1) τ_{weight} becomes larger and this indicates that the travel time for each AGV to complete its journey generally becomes larger; and (2) ET_{weight} becomes larger and this indicates that the deviation of the finishing time for each job from its preferred due-date generally becomes larger.

3. So, a good negotiation skill may be one that combines yielding and shifting techniques. The question that remains to be unanswered is, if combined techniques are used, how much should be yielded and how much should be shifted. Kwa's model does not give any guidance about how to combine them. In order to find a combined skill that can produce the best result of conflict resolution, we did the following experiment:

   (1) A negotiation skill is constructed as:

$$(\text{Yield(p)}, \text{Shift(1-p)}) \quad 0 \leq p \leq 1$$

   where p is the percentage of resource conceded by a yielding technique. In other words, if an agent is considering to concede T to the opponent for resolving a conflict, then it will first concede $T * p$ by yielding techniques, and then concede the rest (i.e., $T * (1 - p)$) by shifting techniques.

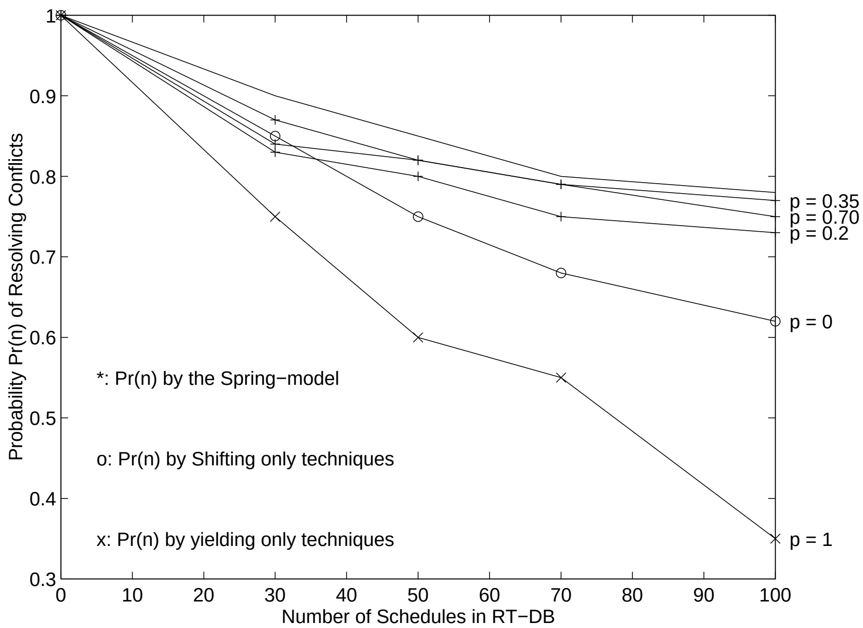


Figure 7.6: The Probability of Resolving Conflicting Schedules

(2) For each given p, we will have a negotiation skill. Apply this skill for solving the AGVSP with Kwa's model, then analyse the results to see how well this skill can resolve conflicts during problem solving.

Figure 7.6 shows the probability of resolving conflicting schedules with parameter p. The X axis is the number of conflict-free schedules in RT-DB, and the Y axis is the probability of resolving a conflicting schedule. From this experiment, we can see that when $p = 0.35$, the ability of Kwa's model to resolve conflicts reaches the best point (which is very close to that — the up-most line — by the spring-based negotiation model). Therefore, we used this negotiation skill for the rest of our experiments.

Results of Experiments

The experimental results are listed in Tabel 7.1. These data are achieved according to the following procedure: (1) generate 100 local schedules for 100 AGVs, (2) use Kwa's model to build the 100 local schedules into a complete conflict-free global schedule and calculate the values of (a) ComTimes — How many times of communication are needed

Model	ComTimes	ET_{weight}	τ_{weight}	R_{weight}
Kwa's Model	956	1674.5	9064.34	1323
Spring Model	254	1299.03	6045.44	950.2

Table 7.1: Experimental Results from Kwa's Model and the Spring-based Model

during negotiatiation, (b) ET_{weight} (see Equation 7.4) — the weighted deviation objective function for measuring the error between the scheduled finishing time of an operation and the preferred finishing time of the operation, (c) τ_{weight} (see Equation 7.3) — the weighted travel time objective function, and (d) R_{weight} (see Equation 7.5) — the weighted deviation objective function for measuring the robustness of the complete schedule, and finally (3) repeat (1) and (2) for 30 times and calculate the averages for $ComTimes$, ET_{weight}, τ_{weight} and R_{weight} respectively. Data listed in Tabel 7.1 are these averages. These data have shown that the spring-based negotiation model is better than Kwa's model in many aspects. To help the reader to understand these data, we clarify them further as follows:

1. Our negotiation model needs many fewer communication times during negotiation time than Kwa's model does. This implies that our model is more efficient than Kwa's. Figure 7.7 roughly shows the communication times needed during a negotiation process by the two model (the lines are based on three points when the number, n, of schedules in the RT-DB are 0, 50 and 100 respectively).

2. Our model meets the JIT requirements better than Kwa's does, since it yields a much smaller number of ET_{weight} than Kwa's does. That is to say, finishing times of operations are scheduled by our model closer to their respective preferred finishing times than by Kwa's model.

3. Our model schedules AGV's travels with shorter travel times than Kwa's does, i.e., τ_{weight} by our model is smaller than by Kwa's. According to our analysis, the main reason is that in our model, there are linking springs, i.e, $t^{\tau}_{j_k j_{k+1}}$, which keep travel times as close to the preferred travel times as possible. Kwa's model does not have such links.

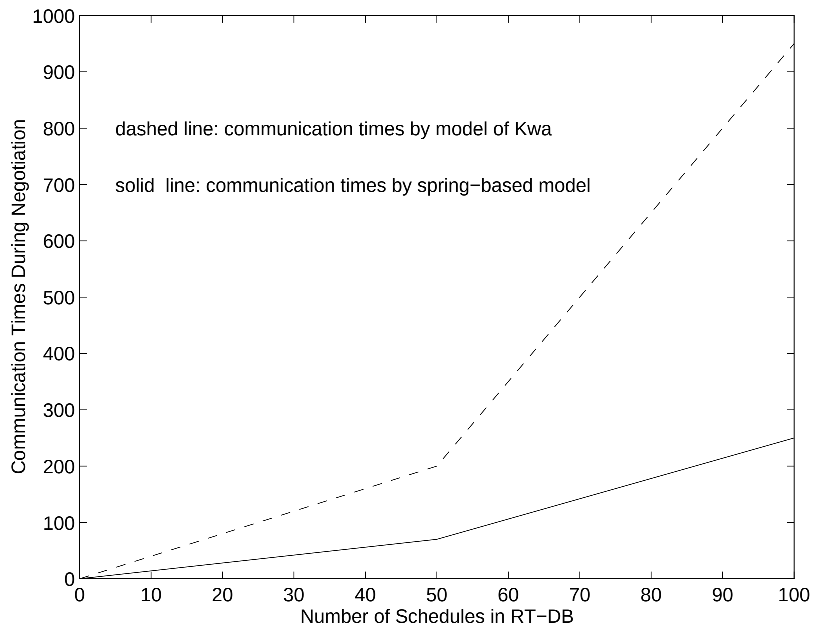


Figure 7.7: The Communication Times

4. Our model produces schedules more robust than Kwa's model does, in terms of R_{weight}. Schedules by our model yields smaller values of R_{weight}. This can be further clarified by the following Figure 7.8 and Figure 7.9. Figure 7.8 illustrates the probability (Y-axis: scale = 1:100) of percentage of remaining tolerant time after negotiation by the spring-based model. 0.5021 in this Figure is the mean percentage of the remaining tolerance resulted from by the spring-based model. Figure 7.9 shows the probability (Y-axis: scale = 1:100) of percentage of remaining tolerant time after negotiation by Kwa's model. 0.3444 is the mean percentage of the remaining tolerance from Kwa's model. From these two figures, our model produces schedules that keep more initially allocated tolerant times than Kwa's does. Since, the major concern of allocating tolerant times is to make schedules more robust, losing initially allocated tolerant times means the robustness is affected negatively. In this sense, our model performs much better than Kwa's. Our experiments also shows that, in the schedule built up by the spring-based model, more important operation nodes (with larger spring constants) will on average keep more tolerance than those with smaller constants do, while in the schedule built up by Kwa's model, this property is not so evident

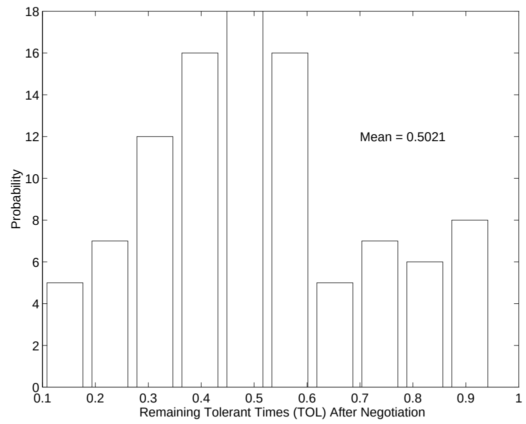


Figure 7.8: The Probability of Percentage of Remaining Tolerant Time After Negotiation By the Spring Model

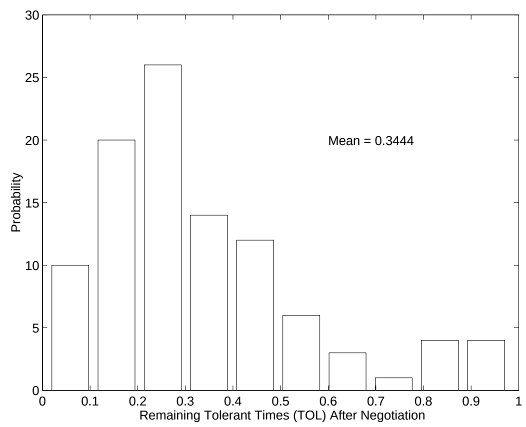


Figure 7.9: The Probability of Percentage of Remaining Tolerant Time After Negotiation By Kwa's Model

— in other words — how much tolerance a node keeps after negotiation is not as closely related to the importance of the node as it is in our spring-based model. (We did not illustrate the experiment results due to space reasons).

7.3.2 Comparing Our Model with a GA Scheduling Method

We also made efforts on comparing our spring model with other scheduling approaches. We tried to run Fang's GA scheduling model ([Fang *et al.* 93] [Corne *et al.* 93] [Corne *et al.* 94]) on the AGVSP problem. However, the results achieved by that model are generally not satisfactory. They are not comparable with the results achieved by our spring model.

Afterwards, we analysed the reasons why Fang's GA scheduling model, which was reported to perform remarkably well for solving JSSP problems, could not solve the AGVSP problems. They are:

1. In general JSSP problems, tolerant planning is not considered. But the AGVSP problem we considered is based on the assumption that tolerant planning is allowed.

2. As is explained in [Fang *et al.* 93], in general JSSP, few heuristics rules can satisfy JIT requirements. However, in our spring model, the spring unit, $t^{\alpha}_{j_k}$, which attracts finishing time of an operation to its preferred finishing time, provides a very good heuristic to satisfy JIT requirements.

3. His model is mainly aimed at reducing the makespan of a complete schedule for all operations. It is a permutation problem, i.e., once the order of starting operations is decided then the scheduling problem is virtually solved. For example, if there are three operations, A, B and C, to be scheduled (they may use the same machine or not, they may have some precedence constraints, e.g., A must be done before B ...), no matter what constraints are, if we know the order in which these operations are done, then the remaining problem of scheduling is to let starting time of each operation be as early as possible as long as these constraints can be satisfied. This means that negotiation between operations' resource allocations is not possible. In our model, we assume that even if the order of any two operations are decided, their starting times are still negotiable.

7.4 Summary

In this chapter, we outlined our implementation of the spring-based negotiation model. We gave two examples to show that this model can solve the AGVSP problem. Some results of experiments are reported to show that the spring-based negotiation model performs better than Kwa's. Although we could not compare with our model with other scheduling models, we presented some reasons to explain this.

Chapter 8

Contributions and Future Work

8.1 Introduction

This chapter concludes the dissertation by summarizing the major contributions we achieved in this dissertation and presenting some possible work that could be done in future.

8.2 Major Contributions

This dissertation focuses mainly on two closely related issues: (1) in general, we investigated how agents, which operate in a situation of the cooperative interaction problem (CIP) where they have to jointly choose a cooperative strategy, can unanimously focus on a single strategy, and (2) specifically, based on the achievements we made in the first part, we developed the spring-based negotiation model to solve the AGVSP problem in a multi-agent system where each agent's local schedule is represented by a spring model.

8.3 Contributions Relating to the CIP problem

1. We constructed the CIP problem and showed that many DAI problems such as the resource allocation problem, the task allocation problem, etc., can be classified as CIP problems.

2. We proved that, in general, a CIP, when agents are assumed to be rational and

benevolent, will become a difficult cooperative problem in which none of these agents is able to improve its current state (payoff) without negatively affecting that of others. So, merely assuming agents are rational and benevolent can only solve the CIP problem in part.

3. We proposed a concept of a common sense principle to allow agents to continue their cooperation when such a difficult situation is met. The common sense principle played the same role as common sense in a social community. We argued that the concept is especially important when we are dealing with a non-open system where the system designer has some implicit or explicit requirements about how the system should perform. The common sense principle, to some extant, reflects these requirements from the system designer. We can also view a common sense principle as a law made by a government — although the government may not be able to totally control how each individuals should behave, the law will constrain these individuals' behaviour. The society is still a multi-agent system, but the law could guide individuals' behaviour such the society develops according to the government's wishes.

4. Based on the concept of a common sense principle, we formally defined metaphor-based negotiation. This definition differs from other definitions of the term negotiation found in the DAI literature in that (1) it formally defines what is a proposal, (2) it formally provides conditions for an agent's accepting an proposal by another agent. With this definition, we are able to derive a negotiation algorithm for solving an application problem, provided that we are able to equip agents with common sense principles.

8.4 Contributions to Solving the AGVSP Problem

We continued the study of the CIP by developing the spring-based negotiation model for solving an application problem: the AGVSP problem. Our major contributions can be summarised as follows:

1. The spring model is a novel representational model for representing hard constraints and soft constraints in the AGVSP. So far, in the scheduling literature,

as far as we learnt, there does not exist such a representational model that can represent hard constraints and soft constraints synthetically.

2. The spring model is a scheduling model which provides a solution space for the AGVSP: as long as the spring model can be pushed, there exist legal solutions to the AGVSP.

3. The spring model is a negotiation model. Two agents each of which has a local spring network, if they are in conflict in resource allocations, will negotiate to resolve the conflict based on their local spring network under a common sense principle that their restoration forces towards each other must be equal.

4. We implemented a spring-based negotiation model for the AGVSP.

8.5 The Spring Model vs Kwa's Negotiation Model

We compared our spring-based model with Kwa's model and the comparison tells us that our model performs better than Kwa's in various aspects: lesser rounds of negotiation — which could imply less computational time, better robustness, short travel times, etc.

In Kwa's model, the procedure of negotiation is very simple: when agents' resource allocations are in conflict, there is an initiator (one which detected the conflict) invoking a negotiation procedure. What each agent will concede totally depends on a skill set which is set by the user prior to the negotiation. In each negotiation round, each agent may work out a negotiation technique (retract the first negotiation technique in the skill set) in a faster speed than it will do in our spring model. However, this also means less autonomy for each agent in deciding what to concede in a particular circumstance. Our model on the other hand allows each agent to work out its negotiation technique based on the situation where the agent is located: more powerful agent (with stronger restoration forces) will apply a technique that concedes lesser resource to other agents. Another advantage of the spring model over Kwa's negotiation model is that our model has a theoretical background while Kwa's does not. In Kwa's model, how a skill set should be set has not been explained, therefore it will be very difficult for us to apply it to a practical problem. In our model, how much an agent should concede and how to concede can be caculated based on the spring model.

Through experiments, we also found that our model needs much lesser negotiation rounds to resolve a conflict than Kwa's does. This may imply that our model needs lesser computational time than Kwa's does. However, further experiments must be done in future to support this argument.

8.6 Future Research

There are several issues that could be raised for future research relating to the work done in this dissertation.

First of all, about the general CIP research, we proposed the concept of a common sense principle. Although, we have shown how to use it in the spring-based negotiation model, we are not able to study further on whether there exists some common principles for generating a common sense principle.

Secondly, about the Spring model, we assume that springs are linear, that is the restoration force of a spring is linearly proportional to the deviation of the assignment of a variable from its preferred assignment. This may not be universally true in the AGVSP. For example, an activity which, due to resource constraints, can not be allocated resources to satisfy its preferred due-date, may be allocated with other resources so that its finishing time is as close to the preferred time as possible. Currently, we use a linear spring to attract the assignment to the preferred due-date. However, it might be the case sometimes that if the activity can not be allocated resources to guarantee its finishing to be very near the preferred due-date (e.g., within two hours), then it might be less important when the activity should be allocated. In this case, we need a non-linear spring unit to represent the activity. This issue has not been dealt with in this dissertation.

Finally, our model is currently demonstrated only on the AGVSP problem. Although we have showed that it can solve the AGVSP better than Kwa's model, it will be more interesting if we can extend this model to other application domains. For example, the spring-based negotiation model may be applied to solve the train scheduling problem [Fukumori 80].

8.7 Conclusion

In conclusion, this dissertation is concerned with solving a cooperative interaction problem. We reported our achievements both in general investigation and in specific application. The experimental results reported have shown that our spring-based model is a promising model in solving the AGVSP problem. It could possibly be extended to other application domains of scheduling in future.

Bibliography

[Aarts & Korst 89] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons Ltd., 1989.

[Adler *et al.* 89] M. Adler, A. Davis, R. Weihmayer, and R. Worrest. Conflict-resolution strategies for non-hierarchical distributed agents. In *Distributed Artificial Intelligence*, volume 2. Pitman Publishing, London, 1989.

[Agha & Hewitt 85] G. Agha and C. E. Hewitt. Concurrent programming using actors: Exploiting large-scale paralleism. Ai memo 865, mit, 1985.

[Agha & Hewitt 86] G. Agha and C. E. Hewitt. *A Model of Concurrent Computation in Distributed Systems.* MIT Press, Cambridge, MA, 1986.

[Allen & Hayes 87] J. F. Allen and P. J. Hayes. Moments and points in an interval-based temporal logic. Department of computer science, university of rochester, new york, December 1987.

[Arbib 72] M. A. Arbib. *The Metaphorical Brain: An Introduction to Cybernetics as Artificial Intelligence and Brain Theory.* New York: Wiley Interscience, 1972.

[Axelrod 80] R. Axelrod. Effective choice in the prisoner's dilema. In *Journal of Conflict Resolution*, volume 24. 1980.

[Axelrod 87] R. Axelrod. The evolution of strategies in the iterative prisoner's dilema. In *Genetic Algorithm and Simulated Annealing.* Pitman Publishing, 1987.

[Balas 69] E. Balas. Machine sequencing via disjunctive graphs: An implicit enumerative algorithm. In *Operations Research.* 1969.

[Bendifallah & Scacchi 87] S. Bendifallah and W. S. Scacchi. Understanding software maintenance work. In *IEEE Transactions on Sortware Engineering.* 1987.

[Bohlander 87] R. Bohlander. New developments in automated guided vehicle system guidance controls. In *Technical Papers Ms87-470, AGVS Research Coordinator, Georgia Institute of Technology*. 1987.

[Bond & Gasser 88] Alan H. Bond and Les Gasser. An analysis of problems and research in dai. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1988.

[Brams 90] S. J. Brams. *Negotiation Games — applying game theory to bargaining and arbitration*. Routledge, Chapman and Hall, Inc., 1990.

[Bratman 87] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Massachusetts, 1987.

[Brussel *et al.* 88] H. Brussel, C. Helsdinge, and K. Machiels. Frog-free ranging on grid: New perspectives in automated transport. In *Proceedings of 6th International Automated Guided Vehicle Systems*, pages 223 – 232. 1988.

[Burmester & Sundermeyer 92] B. Burmester and K. Sundermeyer. Cooperative problem-solving guided by intentions and perception. In *Decentralized A.I.-3*. Elsevier Science Publisher B. V., 1992.

[Carlier & Pinson 89] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. In *Management Science*. 1989.

[Conry *et al.* 91] S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer. Multistage negotiation for distributed constraint satisfaction. In *IEEE Transactoins on Systems, Man and Cybernetics*, volume 21(6). 1991.

[Conway *et al.* 67] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison Wesley Publishing Company, 1967.

[Corkill 82] D. D. Corkill. *A Framework for Organizational Self-Design in Distributed Problem solving Networks*. Unpublished PhD thesis, Department of Computer and Information Science, University of Massachusettes, Amherst, MA, 1982.

[Corne *et al.* 93] D. Corne, H. Fang, and C. Mellish. Solving module exam scheduling problem with genetic algorithms. In *Proceedings of the Sixth International Conference* *on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. 1993.

[Corne *et al.* 94] D. Corne, P. Ross, and H. Fang. Evolutionary timetabling practice, prospects and work in progress. In *The 23th UK Planning and Scheduling Special Interest Group*. 1994.

[Davis & Smith 83] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63 – 109, 1983.

[Dorn & Slany 94] J. Dorn and W. Slany. A flow shop with compatibility constraints in a steelmaking plant. In *Intelligent Scheduling*. Morgan Kaufmann Publishers, INC., 1994.

[Durbin *et al.* 89] R. Durbin, R. Szeliski, and A. Yuille. An analysis of the elastic net approach to the travelling salesman problem. In *Neural Computation*, volume 1, pages 345–358. MIT Press, 1989.

[Durfee & Lesser 87] E. H. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint conference on Artificial Intelligence*. 1987.

[Durfee & Lesser 89] E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In *Distributed Artificial Intelligence*. Pitman Publishing Ltd., 1989.

[Durfee *et al.* 87] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Cooperation through communnication in a distributed problem solving network. In *Distributed Artificial Intelligence*, pages 29 – 58. Morgan Kaufmann Publishers, Inc., 1987.

[Eiselt & vonFrajer 77] H. A. Eiselt and H. von Frajer. *Operations Research Handbook*. The Macmillan Press Ltd., 1977.

[Elbracht & Plum 88] D. Elbracht and W. Plum. Wire grided flexible manufacturing line systems in europe, usa and japan. *Proceedings of 2nd International Automated Guided Vehicle Systems*, pages 129–146, 1988.

[Elbrancht & Plum 83] D. Elbrancht and W. Plum. Wire grided flexible manufacturing line systems in europe, usa and japan. In *Proceedings of 2nd International Automated Guided Vehicle Systems*, pages 129 – 146. 1983.

[Ephrati & Rosenschein 91] E. Ephrati and J. S. Rosenschein. The clark tax as a consensus mechanism among automated agents. *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1991.

[Ephrati & Rosenschein 94] E. Ephrati and J. S. Rosenschein. Divide and conquer in multi-agent planning. In *AAAI*. 1994.

[Fang 94] H. Fang. *Genetic Algorithms in Timetabling and Scheduling*. Unpublished PhD thesis, University of Edinburgh, 1994.

[Fang *et al.* 93] H. Fang, P. Ross, and D. Corne. A promising genetic algorithm approach to job-shop scheduling problems. In *Proceedings of the Fifth International Conference on Genetic Algorithms*. 1993.

[Fenster *et al.* 95] M. Fenster, S. Kraus, and J. S. Rosenschein. Coordination without communication: Experimental validation of focal point techniques. In *Proceedings of The First International Conference on Multi-agent Systems*. AAAI Press / The MIT Press, 1995.

[Fox 87] Mark Fox. *Constraint-directed Search: A Case Study of Job-Shop Scheduling*. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1987.

[Fox 94] M. S. Fox. Isis: A retrospective. In *Intelligent Scheduling*. Morgan Kaufmann Publishers, INC., 1994.

[Fukumori 80] K. Fukumori. Fundamental scheme for train scheduling: Application of range-constriction search. A.i.memo no.596, mit, 1980.

[Gasser 84] L. Gasser. The social dynamics of routine computer use in complex organasations. 1984.

[Gasser 86] L. Gasser. The integration of computing and routine work. In *ACM Transactions on Office Information Systems*. 1986.

[Gasser 90] L. Gasser. Social concepts of knowledge and action: Dai foundations and open systems semantics. In *Proceedings of the Tenth International Workshop on Distributed Artificial Intelligence*. October 1990.

[Genesereth *et al.* 86] M. Genesereth, M. Ginsberg, and J.S. Rosenschein. Cooperation without communication. *Proceedings of the Fifth National Conference on Artificial Intelligence*, August 1986.

[Ginsberg 87] M. L. Ginsberg. Decision procedures. In *Distributed Artificial Intelligence*, volume 1, pages 3 –28. 1987.

[Glover & Laguna 93] F. Glover and M. Laguna. Chapter 3: Tabu search. In *Modern Heuristic Techniques for Combinatorial Problems*. 1993.

[Harsanyi 77] J. C. Harsanyi. *Rational Behaviour and Bargaining Equilibrium in Games and Social Situations*. Cambridge University Press, 1977.

[Hewitt 86] C. E. Hewitt. Offices are open systems. In *ACM Transactions on Office Information Systems*, pages 271 – 287. 1986.

[Huang 95] X. Huang. Defining metaphor-based negotiation. In *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, pages 383 — 388. 1995.

[Hughes & Martin 77] J. H. Hughes and K. F. Martin. *Basic Engineering Mechanics*. The Macmillan Press Ltd., 1977.

[Jennings 94] N. R. Jennings. *Cooperation in Industrial Multi-agent Systems*. World Scientific Publishing Co. Pte. Ltd, 1994.

[Knuth 86] D. E. Knuth. *Computing and Typesetting*. Addison Wesley Publishing Company, 1986.

[Kraus & Rosenschein 92] S. Kraus and J. S. Rosenschein. *The role of Representation in Interaction: Discovering Focal Points among Alternative Solutions*, volume 3. Amsterdam: Elsevier Science Publishers, 1992.

[Kraus & Wilkenfeld 90] S. Kraus and J. Wilkenfeld. The function of time in cooperative negotiations: Extended abstract. In *Proceedings of the Tenth Workshop on Distributed Artificial Intelligence*. 1990.

[Kraus & Wilkenfeld 91a] A. Kraus and J. Wilkenfeld. Function of time in cooperative negotiations. In *Proceedings of AAAI*. 1991.

[Kraus & Wilkenfeld 91b] A. Kraus and J. Wilkenfeld. Negotiations over time in a multiagent environment: Preliminary report. In *Proceedings of the International Joint Conference on Artifical Intelligence*. 1991.

[Kraus *et al.* 95] S. Kraus, J. Wilkenfeld, and G. Zlotkin. Multiagent negotiation under time constraints. In *Artificial Intelligence*, volume 75, pages 297–345. 1995.

[Kwa 88a] J. B. H. Kwa. *Planning Automated Guided Vehicle Movements in a Factory.* Unpublished PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1988.

[Kwa 88b] James Boon Hwee Kwa. Tolerant planning and negotiation in multiagent environments. *Applied Artificial Intelligence*, 1988.

[Laasri *et al.* 92] B. Laasri, H. Laasri, S. E. Lander, and V. Lesser. A generic model for intelligent negotiating agents. *Intelligent Information Systems*, 1(2), 1992.

[Lander & Lesser 91] S. E. Lander and V. R. Lesser. A framework for cooperative design. Department of computer and information science, university of massachusetts, amherst, massachusetts 01003, 1991.

[Lander & Lesser 92] Susan E. Lander and Victor R. Lesser. Negotiated search: Organizing cooperative search among heterogeneous expert agents. *Proceedings of the Fifth International Symposium on Artificial Intelligence, Applications in Manufacturing and Robotics*, 1992.

[Lander & Lesser 93] Susan E. Lander and Victor R. Lesser. Understanding the role of negotiation in distributed search among heterogeneous agents. *Proceedings of International Joint Conference on Artificial Intelligence*, 1993.

[Lander 94] Susan E. Lander. *Distributed Search and conflict Management among Reusable Heterogeneours Agents.* Unpublished PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 1994.

[Lander *et al.* 91] S. E. Lander, V. R. Lesser, and M.E. Connell. *Knowledge-based Conflict Resolution for Cooperation among Expert Agents.* Springer-verlag, 1991.

[Liritzis *et al.* 95] I. Liritzis, D. Diagourtas, and C. Markropoulos. Earth moon and planets, 1995.

[Luce & Raiffa 57] R. D. Luce and H. Raiffa. *Games and Decisions.* New York: Wiley, 1957.

[Matwin *et al.* 89] S. Matwin, S. Szpakowicz, and Z. Koperczak. Negoplan: An expert system shell for negotiation support system. *IEEE Expert*, 1989.

[Matwin *et al.* 91] S. Matwin, T. Szapiro, and K. Haigh. Genetic algorithms approach to a negotiation support system.

In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 21(1). 1991.

[Moehlman *et al.* 92] T. A. Moehlman, Victor R. Lesser, and Brand L. Buteau. Decentralized negotiation: an approach to the distributed planning problem. *Group Decision and Negotiation*, 1992.

[Myerson 91] Roger B. Myerson. *Game Theory — Analysis of Conflict*. President and Fellows of Harvard College, 1991.

[Nash 51] J. F. Nash. Noncooperative games. *Annals of Mathematics*, pages 54:289–295, 1951.

[Ohanian 85] H. C. Ohanian. *Physics*. W. W. Norton & Company, Inc., 1985.

[Okamoto *et al.* 95] K. Okamoto, Y. A. Hassan, and W. D. Schmidl. New tracking algorithm for particle image velocimetry, 1995.

[Owen 82] Guillermo Owen. *Game Theory*. Academic Press, Inc, 1982.

[Page & Thomas 89] J. Page and W. Thomas. An object-oriented modeling environment. Los angeles, calif.: Unversity of california, computer science department, 1989.

[Pruitt 81] D. G. Pruitt. *Negotiation Behaviour*. Academic Press, New York, 1981.

[Rosenschein & Genesereth 88] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1988.

[Rosenschein 85] J. S. Rosenschein. *Rational Interation: Cooperation among Intelligent Agents*. Unpublished PhD thesis, Stanford University, 1985.

[Rubinstein 82] A. Rubinstein. Perfect equilibrium in a bargaining model. In *Econometria*, volume 50(1), pages 97 – 109. 1982.

[Sandholm 93] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost caculations. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1993.

[Sathi & Fox 89] A. Sathi and M. S. Fox. Constraint-directed negotiation of resource allocations. In *Distributed Artificial Intelligence*, volume 2, pages 163 – 193. Pitman Publishing, London, 1989.

[Schelling 60] T. C. Schelling. *The Stategy of Conflict.* Harvard University Press, reprint, 1975 edition, 1960.

[Shapiro 89] S. C. Shapiro. *Encyclopedia of Artificial Intelligence*, volume 2. John Wiley and Sons, Inc, 1989.

[Shen *et al.* 94] C. Shen, Y. Pao, and P. Yip. Scheduling multiple job problems with guided evolutionary simulated annealing approach. In *Proceedings of the First IEEE Conference on Evolutionary computation.* 1994.

[Simmen 91] M. W. Simmen. Parameter sensitivity of the elastic net approach to the travelling salesman problem. In *Neural Computation*, volume 3, pages 363–374. MIT Press, 1991.

[Smith & Davis 88] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1988.

[Smith 88] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., 1988.

[Steele 78] G. L. Steele. The revised report on scheme, a dialect of lisp. Mit artificial intelligence memo 452, January 1978.

[Stefik & Bobrow 86] M. Stefik and D. G. Bobrow. Object-oriented programming: Themes and variations. *The AI Magazine*, 6, 1986.

[Sycara 88] K. Sycara. Resolving goal conflicts via negotiation. *Artificial Intelligence*, 1988.

[Sycara *et al.* 90] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Decentralized factory scheduling: Coordinating resource allocation using contrained directed search. In *Proceedings of Tenth Workshop on Artificial Intelligence.* 1990.

[Thomas 84] L. C. Thomas. *Games, Theory and Applications.* Ellis Horwood Limited, 1984.

[Valery 87] N. Valery. Factory of the future. *The Economist*, 30 May - 5 June 1987.

[vonNeumann & Morgenstern 44] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, 1944.

[Yokoo *et al.* 90] M. Yokoo, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for dai problems. In *Proceedings of Tenth Workshop on Artificial Intelligence*. 1990.

[Yue *et al.* 97] S. G. Yue, Y. G. Yu, and S. X. Bai. Flexible rotor beam element for the manipulators with joint and link flexibility, 1997.

[Zlotkin & Rosenschein 91] G. Zlotkin and J. S. Rosenschein. Negotiation and goal relaxation. In *Decentralized AI 2*. Elsevier Science Publishers, B.V, 1991.

[Zlotkin & Rosenschein 93] G. Zlotkin and J. S. Rosenschein. A domain theory for task oriented negotiation. In *Proceedings of the 13th Internation Joint Conference on Artificial Intelligence*, pages 417 — 422. 1993.

[Zlotkin & Rosensein 89] G. Zlotkin and J. S. Rosensein. Negotiation and task sharing among agents in cooperative domains. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 912–917, 1989.

[Zlotkin & Rosensein 90] G. Zlotkin and J. S. Rosensein. Negotiation and conflict resolution in non-cooperative domains. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 100–105, 1990.

[Zlotkin & Rosensein 91] G. Zlotkin and J. S. Rosensein. Cooperation and conflict resolution via negotiation among autonomous agents in non-cooperative domains. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 20(1), 1991.

[Zlotkin & Rosensein 96] G. Zlotkin and J. S. Rosensein. Mechanisms for automated negotiation in state oriented domains. In *Artificial Intelligence Research*, volume 5, 1996.

Appendix A

Iterative Negotiation Algorithms

Algorithm A.1

```
=============================
procedure:  plan(Agent, Task)
=========================
/* Construct a conflict-free Plan */
vars: P, pi, MainConflictSet,Residue1, pi' P';

1. Generate Plan P independently for Agent to achieve Task.
      /* P is feasible from the viewpoint of Agent, but may involve steps which conflict with
      other agents. In general P is a partially ordered graph of primitive plan steps */

2. until all plan steps in P have been considered do

3. Choose a plan step pi not considered before.

4. Identify MainConflictSet, the set of agents in conflict with Pi.

5. if MainConflictSet is empty then goto step 13 endif

6. Residuel ← ResolveConflict(Agent, pi, MainConflictS

7. if Residue1 = 0 then goto step 13 endif
      /* pi cannot be made conflict-free */

8. Find an alternative plan step pi' for pi.

9. if pi' exists then pi ← pi'; go to step 4 endif
      /* P is infeasible */

10. Find an alternative plan P' for Agent to achieve Task.

11. if P' exists then P ← P'; repeat step 2
      afresh endif

12. signal failure and exit
      /* Task cannot be achieved by Agent */

13. enduntil

endprocedure
```

Algorithm A.2

===

procedure: ResolveConflict(Initiator, Action, ConflictSet)

===

/* Attempt to eliminate all conflicts associated with Action; returns the residue */

var: Residue2, Respondent, C;

1. **Residue2 $\leftarrow$ 0**
2. **until Residue2 $>$ 0 or all members of ConflictSet have been considered before.**
3. **Choose from ConflictSet, an agent Respondent not consi- dered before.**
4. **Compute conflict C between Initiator and Respondent with respect to Action.**
5. **Residue2 $\leftarrow$ Negotiate(Initiator, Respondent, C) enduntil**
6. **Return Residue2.**

endprocedure

Algorithm A.3

procedure: Negotiation(Initiator, Respondent, Differential)

/* Iteratively negotiate to eliminate Differential; returns the residue */

1. **Differential $\leftarrow$ ApplyTechnique(Initiator, Differential).**
2. **if Differential $=$ 0 then exit, return 0 endif**
3. **Differential $\leftarrow$ ApplyTechnique(Respondent, Differential)**
4. **if Differential $=$ 0 then exit, returning 0 endif**
5. **if no new techniques can be applied to reduce Differential**
6. **then exit, returning Differential /* Negotiation impasse occurred */ endif**
7. **goto step 1**

endprocedure

Algorithm A.4

====================================

procedure: ApplyTechnique(Agent, Differential)

====================================

/* Respond by applying a technique in the current negotiation round; returns the residue */

vars: T, SubAgents, Residue3;

1. **Find technique T appropriate for Agent to reduce Differential**
2. **if prior to T's application, negotiation with Agent's subsidiary agents is required.**
3. **then identify Subagents, the set of Agent's subsidi- ary agents involved.**
4. **Residue3 $\leftarrow$ ResolveConflict(Agent, T, SubAgents) endif**
5. **Apply T to reduce Differential as far as Residue3 would allow.**
6. **return Differential**

endprocedure

Appendix B

Reimplementation of the Iterative Negotiation Model

In order to be able to perform backtracking search, every negotiator must keep the history of negotiation with all the information undestroyed.

In this section, we present our reimplementation of Kwa's iterative negotiation model. We are aiming to provide the top planner with the necessary information, so that backtracking can be performed if the top level planner wishes to do so. To implement the model, we must consider the following requirements that have to be met by the negotiation programme :

- The *ability* to find a resolution for the conflict. By the ability is meant that a resolution can be found if there exists any. For a conflict there may exist several resolutions, but it is of principal interest whether the negotiation can find one of them.
- The *minimum influence* on other agents' plans. If two agents are in conflict, it is better to resolve the conflict between them only without involving a third agent for yielding resources. Otherwise there are problems which might result from the chain reaction of nested negotiation. This does not mean that calling a third agent is strictly prohibited. Over-emphasizing the spirit of altruism will result in another problem of unbalanced overall robustness of the multi-agent system, that is, the agents involved in the negotiation will severely compromise their robustness for the sake of not asking a third agent to yield its resources.
- The *fast speed* of completing the negotiation. The speed must be fast enough because negotiation is a dynamic procedure from the viewpoint of the whole AGV planning system. If the speed is slow the negotiation will not serve the purpose of overcoming the problem of frequent replanning.

To decrease the influence on other agents' plans, the negotiation skills can be combined by using yield skills first and using shift skills after that. The fastest speed of completing the whole negotiation process depends on several factors:

- The *search strategy* we choose.
- The *computer programming language* we use.
- The *design* of the negotiation program.

In this appendix we implement Kwa's interactive negotiation model, aiming to preserve all the necessary information during negotiation so that it can be utilized to perform backtracking in future work.

B.1 The Programming Language Choice

The computer programming language used for implementing the iterative negotiation models must satisfy with the requirement of the fast execution of the negotiation procedure.

We have noticed that the iterative negotiation models are all procedural so it is better to use a kind of procedural computer programming language to implement them. For this reason, we choose the **SCHEME** programming language [Steele 78]. **SCHEME** is a statically scoped and properly tail-recursive dialect of the **LISP** programming language. It was designed to have an exceptionally clear and simple semantics and few different ways to form expressions. Using a *Scheme* $\longrightarrow$ *C* interpreter, **SCHEME** can be compiled to **C** and thus the **SCHEME** language can be combined with **C**, fairly efficiently. **C** is generally considered as one of the fastest high level language used for control purposes. Simply speaking, we choose the **SCHEME** language for implementing the iterative negotiation models mainly because the **SCHEME** language is a kind of procedural language and can be compiled to **C**.

There is another reason why we choose the **SCHEME**. The **SCHEME** programming can be run in **EZD**, an easy drawing program for the X-window environment. All drawing commands are written in the form of **SCHEME** lists. We can use this feature to design a display system for our AGV planning system.

B.2 The Appearance of a Node in a Plan for an AGV

A plan for an AGV can be examined from various perspectives.

Firstly, we can regard a plan as a string of nodes which are the names of places that the AGV will arrive at and depart from in due time:

$$PLAN = (PLC_1, PLC_2, ..., PLC_n) \tag{B.1}$$

$$NODE = PLC_i \ (i = 1, 2, ..., n) \tag{B.2}$$

Secondly, if we are interested in the *INTERV* s (or RTokens) that an AGV reserved then we can regard a plan as a list of RTs:

$$PLAN = (RT_1, RT_2, ..., RT_n) \tag{B.3}$$

$$NODE = RT_i \ (i = 1, 2, ..., n) \tag{B.4}$$

Finally, if we concentrate on how each *INTERV* is composed then we can define a plan as:

$$PLAN = (INTV_1, INTV_2, ..., INTV_N) \tag{B.5}$$

$$NODE = INTV_i \ (i = 1, 2, ..., n) \tag{B.6}$$

The three types of definitions of a node are consistent but they express nodes to a different degree. A node in form of an INTV will contain more information than it does in the form of an RT. Similarly, a node in the form of an RT contains more information than it does in the form of a *PLC*. We do not pay much attention to the form *PLC*, because it does not contain as much information as is needed during the negotiation.

A node in the form of an RT contains the maximum limit of information that one *AGV* is willing to offer to another. As stated in chapter 2, every *AGV* is selfish, each one is trying to concede as little resource as possible during negotiation. This means every *AGV* must keep its secrets from other *AGV*s, i.e, to keep their internal compositional parts of the INTVs as well as other information such as *bottomline* , TECH , etc. concealed. Therefore it is better to distinguish the internal look of a node in the form of an INTV (or in more complicated form) from the external look of a node in the form of an RT. Every *AGV* does not want other *AGV*s to spy into its own internal affairs, but the *AGV* itself knows the details of the compositional parts of its every node. This is similar to the case of inviting tenders when a company is trying to invite a contractor to accept an engineering business. If the bottom line of the engineering business is revealed to one contractor then the contractor may possibly to accept the business in its best favour, i.e, the company offers very high price. In the case of negotiation, if one *AGV* 's negotiation bottom line is open to its opponent, the opponent could possibly not yield any resource to the *AGV* before the *AGV* exhausted all its redundant resources.

B.2.1 Layered Negotiation Process

In the iterative negotiation model, we can see that the process of conflict resolution through negotiation is an iterative process, the negotiator and the respondent give in resources in turn and it needs several rounds to finish. In every round, the initiator and the respondent may possibly concede some of their resources and their internal structures of nodes are therefore being modified, moreover, there are probably nested negotiations involved and thus other *AGV* s' nodes are probably modified too. Those modified nodes will be used in the further negotiation rounds and such modifications will be continued until the whole negotiation process finishes. In order to be able to do backtracking search, every modification should be stored in some way such that when the negotiation fails in one round, it can still continue by going back one round and seeking for other alternative conflict resolutions. Figure B.1 illustrates such a layered negotiation process. Every state in the tree presents the states of all the nodes in *AGV* plans after a round negotiation is finished. Applying different negotiation strategies, such as, the order of using negotiation techniques, etc., by the AGV s involved in the negotiation will lead to different states. Backtracking search allows the negotiation to continue by shifting it from one level at which the negotiation fails to its previous level, e.g. from S111 to S11, and searching in an alternative direction, say, to S112, from S11 and thus saves the searching time from S0 to S11.

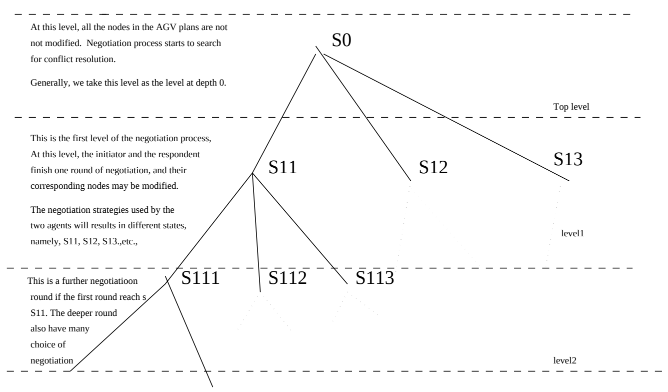


Figure B.1: Layered Negotiation Process

In our reimplementation, the layered process is represented in terms of negotiation depth. A unit of depth indicates one round. So all the nodes which are modified at this depth will be stored with the depth associated.

B.2.2 The design

Our reimplementation is designed in three main modules: the negotiator, the data bases with their handlers and the top level planner. We assume that the top level planner exists and will do the final decision about whether the negotiation results can be committed or not. So the main tasks of the reimplementation are to write the negotiation program and data base handler. Figure B.2 shows the AGV planning system modules.

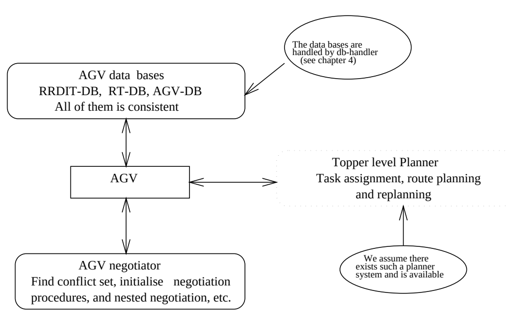


Figure B.2: The AGV Planning System Modules

The negotiator for an AGV will find the conflict set by consulting the data bases through the data base handler and then choose one respondent to negotiate with and record the negotiation history by storing all changes of its nodes along with the negotiation depth while leaving the information about the past of those nodes unaltered. The negotiator does not make the final decision, leaving that right to the top planner.

B.2.3 Example

Suppose AGV_1 has a plan ($Node_{11}$ $Node_{12}$ $Node_{13}$) which is executable and the only plan that has been inserted into the data bases sometime before, and AGV_2 has a plan, ($Node_{21}$ $Node_{22}$ $Node_{23}$), which is newly created by the planner and has not been put into the data base yet.

The nodes for the two plan in terms of RTs are:

$Node_{11} = (AGV_1\ PLC_a\ 100\ 118)$
$Node_{12} = (AGV_1\ PLC_b\ 200\ 230)$
$Node_{13} = (AGV_1\ PLC_c\ 300\ 348)$
$Node_{21} = (AGV_2\ PLC_d\ 80\ 99.8)$
$Node_{22} = (AGV_2\ PLC_b\ 190\ 213.4)$
$Node_{23} = (AGV_2\ PLC_b\ 300\ 354))$

When we run the negotiation program, $Node_{21}$ is inserted into the data base where the existing plan nodes (e.g. for AGV_2) are stored, for there is no conflict node found. After that $Node_{22}$ whose RToken , (190 213.4), is reserved at a place called PLC_b, is found a conflict node, $Node_{12}$ which reserved an RToken, (200 230), at the same place, PLC_b. The negotiation procedures are then invoked to resolve conflict and the conflict is resolved by $Node_{22}$ yielding its RToken to (184.4 206) and $Node_{12}$ to (206 232). The updated $Node_{22}$ is then inserted into the data base. $Node_{23}$ has no conflict node in the data base and is inserted into the data base. The negotiation process is finished in three rounds. Checking the data base after the whole process is finished we can find the history of the negotiation stored in the data base.

The whole process is presented below:

```
RT : (AGV2 PLCD 80 99.8)
RT : (AGV2 PLCB 190 213.4)

==================================================
RT for Newplan node     : (AGV2 PLCB 190 213.4)
Conflict node on RIGHT  : (AGV1 PLCB 200 230)
Differential            : 13.40000000000001
====================================================

           ----------Concession Disp---------
           Type         : Positive Concession
           Depth        : 200
           Technique    : SHIFTLEFT
           Differential : 13.40000000000001
           Conceded     : 2
           Residue      : 11.40000000000001
           rt           : (AGV2 PLCB 188 211.4)
           ---------------------------------------

           ----------Concession Disp---------
           Type         : Positive Concession
           Depth        : 200
           Technique    : SHIFTRIGHT
```

```
  Differential : 11.40000000000001
  Conceded     : 2
  Residue      : 9.40000000000001
  rt           : (AGV1 PLCB 202 232)
  ---------------------------------------
  ----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 201
  Technique    : SHIFTLEFT
  Differential : 9.40000000000001
  Conceded     : 2
  Residue      : 7.40000000000001
  rt           : (AGV2 PLCB 186 209.4)
  ---------------------------------------
  ----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 201
  Technique    : SHIFTRIGHT
  Differential : 7.40000000000001
  Conceded     : 0
  Residue      : 7.40000000000001
  rt           : (AGV1 PLCB 202 232)
  ---------------------------------------
----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 201
  Technique    : YIELDFRONTHEDGE
  Differential : 7.40000000000001
  Conceded     : 2
  Residue      : 5.40000000000001
  rt           : (AGV1 PLCB 204 232)
  ---------------------------------------

  ----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 202
  Technique    : SHIFTLEFT
  Differential : 5.40000000000001
  Conceded     : 1.599999999999994
  Residue      : 3.800000000000016
  rt           : (AGV2 PLCB 184.4 207.8)
  ---------------------------------------
  ----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 202
  Technique    : YIELDFRONTHEDGE&SHIFT
  Differential : 3.800000000000016
  Conceded     : 0.
  Residue      : 3.800000000000016
  rt           : (AGV2 PLCB 184.4 207.8)
  ---------------------------------------
  ----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 202
  Technique    : YIELDENDHEDGE
  Differential : 3.800000000000016
  Conceded     : 0.400000000000006
  Residue      : 3.400000000000010
  rt           : (AGV2 PLCB 184.4 207.4)
  ---------------------------------------
  ----------Concession Disp---------
  Type         : Positive Concession
  Depth        : 202
  Technique    : SHIFTRIGHT
  Differential : 3.400000000000010
  Conceded     : 0
  Residue      : 3.400000000000010
  rt           : (AGV1 PLCB 204 232)
```

```
--------------------------------------
----------Concession Disp---------
Type         : Positive Concession
Depth        : 202
Technique    : YIELDFRONTHEDGE
Differential : 3.400000000000010
Conceded     : 2
Residue      : 1.400000000000010
rt           : (AGV1 PLCB 206 232)
--------------------------------------
----------Concession Disp---------
Type         : Positive Concession
Depth        : 203
Technique    : SHIFTLEFT
Differential : 1.400000000000010
Conceded     : 2
Residue      : 1.400000000000010
rt           : (AGV2 PLCB 184.4 207.4)
--------------------------------------

----------Concession Disp---------
Type         : Positive Concession
Depth        : 203
Technique    : YIELDFRONTHEDGE&SHIFT
Differential : 1.400000000000010
Conceded     : 0.
Residue      : 1.400000000000010
rt           : (AGV2 PLCB 184.4 207.4)
--------------------------------------
----------Concession Disp---------
Type         : Positive Concession
Depth        : 203
Technique    : YIELDENDHEDGE
Differential : 1.400000000000010
Conceded     : 1.400000000000010
Residue      : 0.
rt           : (AGV2 PLCB 184.4 206.)
--------------------------------------

RT : (AGV2 PLCE 300 354) Has been Inserted
```

From the example, we can see that: the negotiation process is completed in several rounds (here is three rounds) and every AGV 's negotiator may apply several negotiation techniques to concede its planned amount which it wish to give up in each round.

Here is a more complex example:

Suppose the plan nodes for AGV_1 and AGV_2 are in the data base and they are the results of the negotiation in the example above:

$$Node_{11} = (AGV_1\ PLC_a\ 100\ 118)$$
$$Node_{12} = (AGV_1\ PLC_b\ 206\ 232)$$
$$Node_{13} = (AGV_1\ PLC_c\ 300\ 348)$$
$$Node_{21} = (AGV_2\ PLC_d\ 80\ 99.8)$$
$$Node_{22} = (AGV_2\ PLC_b\ 184.4\ 206)$$
$$Node_{23} = (AGV_2\ PLC_b\ 300\ 354))$$

Now, there is a new plan to confirm whose nodes are:

$$Node_{31} = (AGV_3 \; PLC_f \; 60 \; 79.8)$$
$$Node_{32} = (AGV_3 \; PLC_b \; 175 \; 199)$$
$$Node_{33} = (AGV_3 \; PLC_g \; 330 \; 384)$$

Since $Node_{32}$ is in conflict with $Node_{22}$, the negotiation procedures are called. For the Rtoken corresponding to $Node_{22}$ is so close to that corresponding to $Node_{12}$, the nested conflict occurs when $Node_{22}$ concedes resource by *SHIFT* techniques and nested negotiation is called. After the negotiation, $Node_{22}$ and $Node_{32}$ become respectively:

$$Node_{22} = (AGV_2 \; PLC_b \; 191.5 \; 206)$$
$$Node_{32} = (AGV_3 \; PLC_b \; 171 \; 191.5)$$

The record of the process is listed below:

```
RT : (AGV3 PLCF 60 79.8)
RT : (AGV3 PLCB 175 199)
================================================
RT for Newplan node     : (AGV3 PLCB 175 199)
Conflict node on RIGHT  : (AGV2 PLCB 184.4 206.)
Differential            : 14.59999999999999
================================================

----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 200
Technique    : SHIFTLEFT
Differential : 14.59999999999999
Conceded     : 1.5
Residue      : 13.09999999999999
rt           : (AGV3 PLCB 173.5 197.5)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 200
Technique    : SHIFTRIGHT
Differential : 13.09999999999999
Conceded     : 1.5
Residue      : 11.59999999999999
rt           : (AGV2 PLCB 185.9 207.5)
---------------------------------------
           -----Start Nested Nego-----
           Depth: 200
           Dir  : RIGHT
           RT1  : (AGV2 PLCB 185.9 207.5)
           RT2  : (AGV1 PLCB 206 232)
           DF   : 1.5
```

```
          -----------------------------
          ----------Conceded by One Tech--------
          Type         : Possitive Concesion
          Depth        : 200
          Technique    : YIELDENDHEDGE
          Differential : 1.5
          Conceded     : 1.5
          Residue      : 0.
          rt           : (AGV2 PLCB 185.9 206.)
          --------------------------------------

----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 201
Technique    : SHIFTLEFT
Differential : 11.59999999999999
Conceded     : 1.5
Residue      : 10.09999999999999
rt           : (AGV3 PLCB 172. 196.)
-----------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 201
Technique    : SHIFTRIGHT
Differential : 1.5
Conceded     : 1.5
Residue      : 0.
rt           : (AGV2 PLCB 187.4 207.5)
---------------------------------------
           -----Start Nested Nego-----
           Depth: 201
           Dir  : RIGHT
           RT1  : (AGV2 PLCB 187.4 207.5)
           RT2  : (AGV1 PLCB 206 232)
           DF   : 1.5
          -----------------------------
          ----------Conceded by One Tech----------
          Type         : Possitive Concesion
          Depth        : 201
          Technique    : YIELDENDHEDGE
          Differential : 1.5
          Conceded     : 0.5999999999999885
          Residue      : 0.9000000000000115
          rt           : (AGV2 PLCB 187.4 206.9)
          ---------------------------------------
           ----------Conceded by One Tech----------
          Type         : Possitive Concesion
          Depth        : 201
          Technique    : YIELDFRONTHEDGE&SHIFT
          Differential : 0.9000000000000115
          Conceded     : 0.9000000000000115
          Residue      : 0.
          rt           : (AGV2 PLCB 187.4 206.)
          ---------------------------------------
```

```
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 202
Technique    : SHIFTLEFT
Differential : 8.599999999999994
Conceded     : 1.
Residue      : 7.599999999999994
rt           : (AGV3 PLCB 171. 195.)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 202
Technique    : YIELDFRONTHEDGE&SHIFT
Differential : 7.599999999999994
Conceded     : 0.
Residue      : 7.599999999999994
rt           : (AGV3 PLCB 171. 195.)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 202
Technique    : YIELDENDHEDGE
Differential : 7.599999999999994
Conceded     : 0.5
Residue      : 7.099999999999994
rt           : (AGV3 PLCB 171. 194.5)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 202
Technique    : SHIFTRIGHT
Differential : 7.099999999999994
Conceded     : 1.5
Residue      : 5.599999999999994
rt           : (AGV2 PLCB 188.9 207.5)
---------------------------------------
           -----Start Nested Nego-----
           Depth: 202
           Dir  : RIGHT
           RT1  : (AGV2 PLCB 188.9 207.5)
           RT2  : (AGV1 PLCB 206 232)
           DF   : 1.5
          -----------------------------
          ----------Conceded by One Tech----------
          Type         : Possitive Concesion
          Depth        : 202
          Technique    : YIELDENDHEDGE
          Differential : 1.5
          Conceded     : 0.
          Residue      : 1.5
          rt           : (AGV2 PLCB 188.9 207.5)
          ---------------------------------------
          ----------Conceded by One Tech----------
          Type         : Possitive Concesion
```

```
          Depth        : 202
          Technique    : YIELDFRONTHEDGE&SHIFT
          Differential : 1.5
          Conceded     : 1.5
          Residue      : 0.
          rt           : (AGV2 PLCB 188.9 206.)
          ---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 203
Technique    : SHIFTLEFT
Differential : 5.599999999999994
Conceded     : 0.
Residue      : 5.599999999999994
rt           : (AGV3 PLCB 171. 194.5)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 203
Technique    : YIELDFRONTHEDGE&SHIFT
Differential : 5.599999999999994
Conceded     : 0.
Residue      : 5.599999999999994
rt           : (AGV3 PLCB 171. 194.5)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 203
Technique    : YIELDENDHEDGE
Differential : 5.599999999999994
Conceded     : 1.5
Residue      : 4.099999999999994
rt           : (AGV3 PLCB 171. 193.)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 203
Technique    : SHIFTRIGHT
Differential : 4.099999999999994
Conceded     : 1.5
Residue      : 2.599999999999994
rt           : (AGV2 PLCB 190.4 207.5)
---------------------------------------
           -----Start Nested Nego-----
           Depth: 203
           Dir  : RIGHT
           RT1  : (AGV2 PLCB 190.4 207.5)
           RT2  : (AGV1 PLCB 206 232)
           DF   : 1.5
          -----------------------------
          ----------Conceded by One Tech----------
          Type         : Possitive Concesion
          Depth        : 203
          Technique    : YIELDENDHEDGE
```

```
          Differential : 1.5
          Conceded     : 0.
          Residue      : 1.5
          rt           : (AGV2 PLCB 190.4 207.5)
          ---------------------------------------
          ----------Conceded by One Tech----------
          Type         : Possitive Concesion
          Depth        : 203
          Technique    : YIELDFRONTHEDGE&SHIFT
          Differential : 1.5
          Conceded     : 1.499999999999988
          Residue      : 1.154631945610163e-14
          rt           : (AGV2 PLCB 190.4 206.)
          ---------------------------------------
          ----------Conceded by One Tech----------
          Type         : Possitive Concesion
          Depth        : 203
          Technique    : YIELDTOL
          Differential : 1.154631945610163e-14
          Conceded     : 1.154631945610163e-14
          Residue      : 0.
          rt           : (AGV2 PLCB 190.4 206.)
          ---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 204
Technique    : SHIFTLEFT
Differential : 2.599999999999994
Conceded     : 0.
Residue      : 2.599999999999994
rt           : (AGV3 PLCB 171. 193.)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 204
Technique    : YIELDFRONTHEDGE&SHIFT
Differential : 2.599999999999994
Conceded     : 0.
Residue      : 2.599999999999994
rt           : (AGV3 PLCB 171. 193.)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 204
Technique    : YIELDENDHEDGE
Differential : 2.599999999999994
Conceded     : 1.5
Residue      : 1.099999999999994
rt           : (AGV3 PLCB 171. 191.5)
---------------------------------------
----------Concession Disp---------
Type         : Possitive Concesion
Depth        : 204
Technique    : SHIFTRIGHT
```

```
Differential : 1.099999999999994
Conceded     : 1.099999999999994
Residue      : 0.
rt           : (AGV2 PLCB 191.5 207.1)
---------------------------------------
          -----Start Nested Nego-----
          Depth: 204
          Dir  : RIGHT
          RT1  : (AGV2 PLCB 191.5 207.1)
          RT2  : (AGV1 PLCB 206 232)
          DF   : 1.099999999999994
         -----------------------------
         ----------Conceded by One Tech----------
         Type         : Possitive Concesion
         Depth        : 204
         Technique    : YIELDENDHEDGE
         Differential : 1.099999999999994
         Conceded     : 0.
         Residue      : 1.099999999999994
         rt           : (AGV2 PLCB 191.5 207.1)
         ---------------------------------------
         ----------Conceded by One Tech----------
         Type         : Possitive Concesion
         Depth        : 204
         Technique    : YIELDFRONTHEDGE&SHIFT
         Differential : 1.099999999999994
         Conceded     : 0.
         Residue      : 1.099999999999994
         rt           : (AGV2 PLCB 191.5 207.1)
         ---------------------------------------
         ----------Conceded by One Tech----------
         Type         : Possitive Concesion
         Depth        : 204
         Technique    : YIELDTOL
         Differential : 1.099999999999994
         Conceded     : 1.099999999999994
         Residue      : 0.
         rt           : (AGV2 PLCB 191.5 206.)
         ---------------------------------------

RT : (AGV3 PLCG 330 384)
```

From the two examples, we can see that how the negotiation is going depends largely on what negotiation techniques are used and the resources yielded in each round. In our implementation, these two problems are solved by assuming that a negotiation technique is chosen by selecting the first technique in a prioritized technique list and that the concession amount in every is the fixed number which is given prior to the negotiation process. However, by doing so, the characteristics of negotiation could not be sufficiently expressed. In real case of negotiation, negotiation techniques and concession amount should be chosen dynamically according to the updated information the negotiator can obtain. This could be a very complex process and could be a very interesting topic in the future as the continuation of the project.

B.3 Summary

In this appendix, we described Kwa's iterative negotiation model and its implementation. We found that in order to allow backtracking to be performed, one important premise is that the history of the negotiation must be preserved. We reimplemented Kwa's model based on these thoughts and made the negotiation process layered so that the history of the negotiation can be kept in an ordered way.

The results of the examples gave an outline of how the reimplemented negotiation process works.

It is better to say the implementation provides the necessary information for backtracking than to say it performed backtracking for we have not considered the searching strategies yet.

Appendix C

Trace of Spring-based Negotiation

```
Scheduling for AGV1.....
The provisional  local schedule for AGV1 is as follow:
(X29 (41.52982248432122 76.52982248432122))
(X24 (60.21637080720203 95.21637080720203))
(X25 (95.38049966821754 130.3804996682175))
(X20 (108.4649616413781 143.4649616413781))
(X15 (129.1927313754573 174.1927313754573))
(X10 (141.2296056943943 176.2296056943943))
(X15 (190.0862171429275 225.0862171429275))
(X14 (226.6669733534124 261.6669733534124))
(X13 (272.6923784661721 327.6923784661721))
The conflict-free local schedule for AGV1 is as follow:
(X29 (41.52982248432122 76.52982248432122))
(X24 (60.21637080720203 95.21637080720203))
(X25 (95.38049966821754 130.3804996682175))
(X20 (108.4649616413781 143.4649616413781))
(X15 (129.1927313754573 174.1927313754573))
(X10 (141.2296056943943 176.2296056943943))
(X15 (190.0862171429275 225.0862171429275))
(X14 (226.6669733534124 261.6669733534124))
(X13 (272.6923784661721 327.6923784661721))

Scheduling for AGV2.....
The provisional  local schedule for AGV2 is as follow:
(X19 (52.97653978892799 87.97653978892799))
(X14 (68.29423298154666 103.2942329815467))
(X15 (96.47474816186309 131.4747481618631))
(X10 (111.3050954500883 156.3050954500883))
(X15 (153.8190942386078 188.8190942386078))
(X20 (182.5458225195324 217.5458225195324))
(X25 (203.9697658134785 238.9697658134785))
(X24 (246.0402196497438 281.0402196497438))
(X29 (258.4708199191716 318.4708199191716))
Initial state of negotiation between AGV2 and AGV1:
(1) They have reserved RTokens at station X15;
(2) The two Rtokens overlap with a differential of 2.282016786405762;
(3) AGV2 will concede on TOLB side;
(4) AGV1 will concede on TOLA side;
(5) AGV2's RToken is (96.47474816186309 131.4747481618631);
(6) AGV1's RToken is (129.1927313754573 174.1927313754573);
Now the negotiation starts ....
AGV2 conceded on TOLB by 2
AGV2's force changes from 0. to 18.30473039661049
the oppenent's force changes from 20.0004740210011 to 2.553262579895375
```

```
AGV2's internal modification is as
follow:
(0. -0.4477882471213661 0.)
(0. -0.7463137452022863 0.)
(0. -1.54238174008475 -0.4576182599152503)
(0. -0.3159158289219306 0.)
(0. -0.1425026808321945 0.)
(0. -0.0640913199639499 0.)
(0. -0.02840750573830064 0.)
(0. -0.01166202867148058 0.)
(0. -0.002691237385761269 0.)
AGV2's current RTokens are as follows:
(52.52875154180663 87.52875154180663)
(67.54791923634437 102.5479192363444)
(94.93236642177834 129.4747481618631)
(110.9891796211664 155.9891796211664)
(153.6765915577756 188.6765915577756)
(182.4817311995684 217.4817311995684)
(203.9413583077402 238.9413583077402)
(246.0285576210723 281.0285576210723)
(258.4681286817859 318.4681286817859)

AGV1 conceded on TOLA by 1
AGV1's force changes from 0.09305932871413169 to 8.816665049266993
the oppenent's force changes from 20.88585101784847 to 11.73348581954323
AGV1's internal modification is as
follow:
(0. 0.05627775333550034 7.105427357601002e-15)
(0. 0.09379625555916959 0.)
(0. 0.1938455948222924 -1.4210854715202e-14)
(0. 0.2007239868966337 0.)
(-0.4361802860276498 0.5638197139723502 0.)
(0. 0.5438584999763236 0.)
(0. 0.2410570334265287 0.)
(0. 0.09896025582773404 -2.842170943040401e-14)
(0. 0.02283698211408591 0.)
AGV1's current RTokens are as follows:
(41.60814283328735 76.60814283328736)
(60.34690472214558 95.34690472214558)
(95.65026975910089 130.6502697591009)
(108.6090119479829 143.6090119479829)
(130.1927313754573 174.751898122994)
(141.7689759594199 176.7689759594199)
(190.3252848342686 225.3252848342686)
(226.765116931963 261.7651169319629)
(272.7150269842991 327.7150269842991)

Initial state of negotiation between AGV2 and AGV1:
(1) They have reserved RTokens at station X10;
(2) The two Rtokens overlap with a differential of 14.2202036617465;
(3) AGV2 will concede on TOLB side;
(4) AGV1 will concede on TOLA side;
(5) AGV2's RToken is (110.9891796211664 155.9891796211664);
(6) AGV1's RToken is (141.7689759594199 176.7689759594199);
Now the negotiation starts ....
AGV2 conceded on TOLB by 6
AGV2's force changes from 0. to 100.0899717359672
the oppenent's force changes from 165.5896471872104 to 95.72159840560538
AGV2's internal modification is as
follow:
(0. -0.1147500231813012 0.)
(0. -0.1912500386355021 0.)
(0. -0.3952500798467327 0.3952500798467327)
(0. -3.49775070660084 -2.502249293399245)
(0. -1.577758399363063 0.)
(0. -0.7096050250341932 0.)
(0. -0.3145216673946152 0.)
(0. -0.1291194213513336 0.)
```

```
(0. -0.02979678954261544 0.)
AGV2's current RTokens are as follows:
(52.41400151862533 87.41400151862533)
(67.35666919770887 102.3566691977089)
(94.53711634193161 129.4747481618631)
(107.4914289145655 149.9891796211664)
(152.0988331584125 187.0988331584125)
(181.7721261745342 216.7721261745342)
(203.6268366403456 238.6268366403456)
(245.899438199721 280.899438199721)
(258.4383318922432 318.4383318922432)

AGV1 conceded on TOLA by 9
AGV1's force changes from 0. to 81.6494033207569
the oppenent's force changes from 237.2166304306505 to 87.08167282669982
AGV1's internal modification is as
follow:
(0. 0.4065599778834112 -7.105427357601002e-15)
(0. 0.6775999631390448 0.)
(0. 1.40037325715403 -8.526512829121202e-14)
(0. 1.575100926608172 0.)
(0.6333055367772999 4.281523676464587 0.)
(-4.082470166037865 4.917529833962135 0.)
(0. 2.179620532938117 0.)
(0. 0.8947915872061571 -1.13686837721616e-13)
(0. 0.2064903662783308 0.)
AGV1's current RTokens are as follows:
(42.01470281117076 77.01470281117076)
(61.02450468528463 96.02450468528463)
(97.05064301625492 132.0506430162548)
(110.184112874591 145.1841128745909)
(133.8409495151446 179.0334217994586)
(150.7689759594199 181.686505793382)
(192.5049053672067 227.5049053672067)
(227.6599085191691 262.6599085191689)
(272.9215173505775 327.9215173505775)

Initial state of negotiation between AGV2 and AGV1:
(1) They have reserved RTokens at station X15;
(2) The two Rtokens overlap with a differential of 26.93458864104605;
(3) AGV2 will concede on TOLA side;
(4) AGV1 will concede on TOLB side;
(5) AGV2's RToken is (152.0988331584125 187.0988331584125);
(6) AGV1's RToken is (133.8409495151446 179.0334217994586);
Now the negotiation starts ....
AGV2 conceded on TOLA by 13
AGV2's force changes from 0. to 123.8542360149533
the oppenent's force changes from 232.4602168116404 to 120.2631137177369
AGV2's internal modification is as
follow:
(0. 0.1746527079467626 0.)
(0. 0.291087846577966 0.)
(0. 0.6015815495944992 0.)
(0. 1.313129619007469 -0.5252801521771744)
(-3.096355900373908 9.903644099626092 0.)
(0. 4.45421531083673 0.)
(0. 1.974263395938664 0.)
(0. 0.8104870783325282 0.)
(0. 0.1870354796153038 0.)
AGV2's current RTokens are as follows:
(52.58865422657209 87.58865422657209)
(67.64775704428683 102.6477570442868)
(95.13869789152611 130.0763297114576)
(108.804558533573 150.7770290879967)
(165.0988331584125 197.0024772580386)
(186.226341485371 221.226341485371)
(205.6011000362843 240.6011000362843)
(246.7099252780535 281.7099252780535)
```

```
(258.6253673718585 318.6253673718585)

AGV1 conceded on TOLB by 12
AGV1's force changes from 0. to 150.333137354945
the oppenent's force changes from 258.8304974558373 to 142.593471805899
AGV1's internal modification is as
follow:
(0. -0.4163193376224683 -2.842170943040401e-14)
(0. -0.693865562704076 0.)
(0. -1.433988829588486 0.)
(0. -1.672336387385656 0.)
(-0.562810995569464 -4.483343132252799 -7.516656867747201)
(-3.625238473388492 -3.625238473388492 0.)
(0. -1.606831982761662 0.)
(0. -0.6596468139758258 -2.273736754432321e-13)
(0. -0.152226187840597 0.)
AGV1's current RTokens are as follows:
(41.5983834735483 76.59838347354827)
(60.33063912258055 95.33063912258055)
(95.61665418666644 130.6166541866663)
(108.5117764872054 143.5117764872054)
(129.9204173784613 167.0334217994586)
(150.7689759594199 178.0612673199935)
(190.898073384445 225.898073384445)
(227.0002617051933 262.0002617051929)
(272.7692911627369 327.7692911627369)

AGV2 conceded on TOLA by 2
AGV2's force changes from 123.8542360149533 to 143.9051989961156
the oppenent's force changes from 178.1368325850212 to 149.3930524964102
AGV2's internal modification is as
follow:
(0. 0.004136318101586767 0.)
(0. 0.006893863502625663 0.)
(0. 0.014247317905415 -0.014247317905415)
(0. 0.1260811036146947 -0.1260811036146947)
(-0.5012740745290785 1.498725925470922 0.)
(0. 0.5286942007807625 0.)
(0. 0.4916400653306141 0.)
(0. 0.2018311847147061 0.)
(0. 0.04657642724180278 0.)
AGV2's current RTokens are as follows:
(52.59279054467368 87.59279054467368)
(67.65465090778946 102.6546509077895)
(95.15294520943152 130.0763297114576)
(108.9306396371877 150.7770290879967)
(167.0988331584125 198.5012031835095)
(186.7550356861517 221.7550356861517)
(206.0927401016149 241.0927401016149)
(246.9117564627682 281.9117564627682)
(258.6719437991003 318.6719437991003)

The conflict-free local schedule for AGV2 is as follow:
(X19 (52.59279054467368 87.59279054467368))
(X14 (67.65465090778946 102.6546509077895))
(X15 (95.15294520943152 130.0763297114576))
(X10 (108.9306396371877 150.7770290879967))
(X15 (167.0988331584125 198.5012031835095))
(X20 (186.7550356861517 221.7550356861517))
(X25 (206.0927401016149 241.0927401016149))
(X24 (246.9117564627682 281.9117564627682))
(X29 (258.6719437991003 318.6719437991003))

Scheduling for AGV3.....
The provisional  local schedule for AGV3 is as follow:
(X9 (109.9603418205978 144.9603418205978))
(X4 (127.267236367663 162.267236367663))
(X5 (164.4189551598369 199.4189551598369))
```

```
(X10 (179.1833107252355 224.1833107252355))
(X15 (216.2253687080857 251.2253687080857))
(X20 (238.0843391629929 273.0843391629929))
(X25 (257.6662023932287 292.6662023932287))
(X30 (272.6922005522836 327.6922005522836))
Initial state of negotiation between AGV3 and AGV1:
(1) They have reserved RTokens at station X15;
(2) The two Rtokens overlap with a differential of 9.672704676359331;
(3) AGV3 will concede on TOLA side;
(4) AGV1 will concede on TOLB side;
(5) AGV3's RToken is (216.2253687080857 251.2253687080857);
(6) AGV1's RToken is (190.898073384445 225.898073384445);
Now the negotiation starts ....
AGV3 conceded on TOLA by 4
AGV3's force changes from 0. to 92.04972943103911
the oppenent's force changes from 110.4570657120554 to 64.77922506340245
AGV3's internal modification is as
follow:
(0. 0.2431615111615599 0.)
(0. 0.4052691852692192 0.)
(0. 0.8375563162231288 0.)
(0. 1.828214324659086 0.)
(-1.534162157184028 2.465837842815972 0.)
(0. 0.9666877807662786 0.)
(0. 0.5952316092860883 0.)
(0. 0.1373611406045256 0.)
AGV3's current RTokens are as follows:
(110.2035033317594 145.2035033317594)
(127.6725055529323 162.6725055529323)
(165.2565114760601 200.2565114760601)
(181.0115250498946 226.0115250498946)
(220.2253687080857 253.6912065509017)
(239.0510269437592 274.0510269437592)
(258.2614340025148 293.2614340025148)
(272.8295616928881 327.8295616928881)

AGV1 conceded on TOLB by 6
AGV1's force changes from 0. to 68.51605401143375
the oppenent's force changes from 223.5906110345626 to 84.46027821285321
AGV1's internal modification is as
follow:
(0. -0.007210873770063131 6.677398755527975e-05)
(0. -0.01246328286718779 0.)
(0. -0.02602454720903324 0.001417092402562048)
(0. -0.02723529456392271 0.)
(-0.08575040277153789 -0.08575040277153789 0.08575040277153789)
(-0.1303810574034969 -0.1303810574034969 0.)
(-2.574197299428249 -2.574197299428249 -3.425802700571751)
(0. -1.05677573344957 1.70530256582424e-13)
(0. -0.24387132310369 0.)
AGV1's current RTokens are as follows:
(41.59117259977823 76.59123937376576)
(60.31817583971336 95.31817583971336)
(95.5906296394574 130.5920467318599)
(108.4845411926414 143.4845411926414)
(129.9204173784613 167.0334217994586)
(150.7689759594199 177.93088626259)
(190.898073384445 219.898073384445)
(225.9434859717437 260.9434859717435)
(272.5254198396332 327.5254198396332)

The conflict-free local schedule for AGV3 is as follow:
(X9 (110.2035033317594 145.2035033317594))
(X4 (127.6725055529323 162.6725055529323))
(X5 (165.2565114760601 200.2565114760601))
(X10 (181.0115250498946 226.0115250498946))
(X15 (220.2253687080857 253.6912065509017))
(X20 (239.0510269437592 274.0510269437592))
```

```
(X25 (258.2614340025148 293.2614340025148))
(X30 (272.8295616928881 327.8295616928881))

Scheduling for AGV4.....
The provisional  local schedule for AGV4 is as follow:
(X5 (143.7857315481657 178.7857315481657))
(X10 (158.9762192469428 203.9762192469428))
(X5 (203.0874377688627 238.0874377688627))
(X4 (247.256948136691 282.256948136691))
(X9 (262.5977572623133 317.5977572623133))
Initial state of negotiation between AGV4 and AGV3:
(1) They have reserved RTokens at station X5;
(2) The two Rtokens overlap with a differential of 13.52922007210563;
(3) AGV4 will concede on TOLB side;
(4) AGV3 will concede on TOLA side;
(5) AGV4's RToken is (143.7857315481657 178.7857315481657);
(6) AGV3's RToken is (165.2565114760601 200.2565114760601);
Now the negotiation starts ....
AGV4 conceded on TOLB by 7
AGV4's force changes from 0. to 101.8660300281963
the oppenent's force changes from 192.4344335773837 to 92.86912028789132
AGV4's internal modification is as
follow:
(0. -5.726674624647529 -1.273325375352471)
(0. -1.055621872062943 0.)
(0. -0.9173442905778586 0.)
(0. -0.3765939719215226 0.)
(0. -0.08690630121253662 0.)
AGV4's current RTokens are as follows:
(138.0590569235182 171.7857315481657)
(157.9205973748799 202.9205973748799)
(202.1700934782848 237.1700934782848)
(246.8803541647695 281.8803541647695)
(262.5108509611007 317.5108509611007)

AGV3 conceded on TOLA by 7
AGV3's force changes from 0. to 99.56531328949239
the oppenent's force changes from 196.8811340175983 to 95.01510398940201
AGV3's internal modification is as
follow:
(0. 1.550490419566998 0.)
(0. 2.584150699278297 0.)
(-1.659421888158306 5.340578111841694 0.)
(0. 0.5945783445785082 0.)
(0.3458118688982665 0.3458118688982665 0.)
(0. 0.1176936221505969 0.)
(0. 0.1042245972722071 0.)
(0. 0.0240518301397401 0.)
AGV3's current RTokens are as follows:
(111.7539937513264 146.7539937513262)
(130.2566562522105 165.2566562522105)
(172.25651147606 205.5970895879017)
(181.6061033944731 226.6061033944731)
(220.2253687080857 254.0370184197999)
(239.1687205659098 274.1687205659096)
(258.365658599787 293.365658599787)
(272.8536135230278 327.8536135230278)

Initial state of negotiation between AGV4 and AGV3:
(1) They have reserved RTokens at station X10;
(2) The two Rtokens overlap with a differential of 21.31449398040678;
(3) AGV4 will concede on TOLB side;
(4) AGV3 will concede on TOLA side;
(5) AGV4's RToken is (157.9205973748799 202.9205973748799);
(6) AGV3's RToken is (181.6061033944731 226.6061033944731);
Now the negotiation starts ....
AGV4 conceded on TOLB by 11
AGV4's force changes from 0. to 389.9511874673475
```

```
the oppenent's force changes from 730.0854231265779 to 353.3023917407636
AGV4's internal modification is as
follow:
(0. -0.7350732187991298 0.7350732187991298)
(0. -6.125610156658269 -4.874389843341731)
(0. -5.323206777191075 0.)
(0. -2.18531646642549 0.)
(0. -0.5043037999442959 0.)
AGV4's current RTokens are as follows:
(137.323983704719 171.7857315481657)
(151.7949872182216 191.9205973748799)
(196.8468867010937 231.8468867010937)
(244.695037698344 279.695037698344)
(262.0065471611564 317.0065471611564)

AGV3 conceded on TOLA by 11
AGV3's force changes from 0. to 338.1413563444869
the oppenent's force changes from 755.6011125386595 to 365.6499250713119
AGV3's internal modification is as
follow:
(0. 0.1759858926427711 0.)
(0. 0.2933098210712046 0.)
(0.6061736302139025 0.6061736302139025 0.)
(-5.6356892724082 5.3643107275918 0.)
(1.247889545454086 4.020422721188623 0.)
(0. 2.028927087167091 0.)
(0. 1.79673379653758 0.)
(0. 0.4146308761238515 0.)
AGV3's current RTokens are as follows:
(111.9299796439692 146.9299796439689)
(130.5499660732818 165.5499660732818)
(172.25651147606 206.2032632181156)
(192.6061033944731 231.9704141220649)
(222.9979018838202 258.0574411409885)
(241.1976476530769 276.1976476530766)
(260.1623923963245 295.1623923963245)
(273.2682443991517 328.2682443991517)

Initial state of negotiation between AGV4 and AGV1:
(1) They have reserved RTokens at station X10;
(2) The two Rtokens overlap with a differential of 26.13589904436841;
(3) AGV4 will concede on TOLA side;
(4) AGV1 will concede on TOLB side;
(5) AGV4's RToken is (151.7949872182216 191.9205973748799);
(6) AGV1's RToken is (150.7689759594199 177.93088626259);
Now the negotiation starts ....
AGV4 conceded on TOLA by 7
AGV4's force changes from 0. to 324.1037012695327
the oppenent's force changes from 390.365272566484 to 285.8134106494101
AGV4's internal modification is as
follow:
(0. 0.9107731550739118 -0.2146122713512)
(-4.051296265869212 2.948703734130788 -1.788435594592954)
(0. 2.562448360249107 0.)
(0. 1.051952484733761 0.)
(0. 0.2427582657078347 0.)
AGV4's current RTokens are as follows:
(138.234756859793 172.4818924318884)
(158.7949872182216 193.0808655144177)
(199.4093350613428 234.4093350613428)
(245.7469901830777 280.7469901830777)
(262.2493054268643 317.2493054268643)

AGV1 conceded on TOLB by 20
AGV1's force changes from 0. to 298.7402875881696
the oppenent's force changes from 1307.402457051221 to 279.7018473209796
AGV1's internal modification is as
follow:
```

```
(0. -0.3209444719875449 -1.27897692436818e-13)
(0. -0.5349074533127123 0.)
(0. -1.1054754035126 2.557953848736361e-13)
(0. -1.144701950089029 0.01742968931174005)
(-3.331586064767691 -3.331586064767663 3.331586064767663)
(-5.062985620591405 -5.062985620591405 -14.93701437940859)
(-0.3247695030089517 -0.3247695030089517 0.3247695030089517)
(0. -0.1333264275509123 -5.115907697472721e-13)
(0. -0.03076763712749653 0.)
AGV1's current RTokens are as follows:
(41.27022812779069 76.27029490177807)
(59.78326838640065 94.78326838640065)
(94.4851542359448 129.4865713283475)
(107.3398392425524 142.3572689318642)
(129.9204173784613 167.0334217994586)
(150.7689759594199 157.93088626259)
(190.898073384445 219.898073384445)
(225.8101595441928 260.8101595441921)
(272.4946522025057 327.4946522025057)

Initial state of negotiation between AGV4 and AGV3:
(1) They have reserved RTokens at station X5;
(2) The two Rtokens overlap with a differential of 6.793928156772807;
(3) AGV4 will concede on TOLA side;
(4) AGV3 will concede on TOLB side;
(5) AGV4's RToken is (199.4093350613428 234.4093350613428);
(6) AGV3's RToken is (172.25651147606 206.2032632181156);
Now the negotiation starts ....
AGV4 conceded on TOLA by 4
AGV4's force changes from 0. to 141.9820071883558
the oppenent's force changes from 231.612515178742 to 95.24809693104564
AGV4's internal modification is as
follow:
(0. 0.06180072355360267 -0.06180072355360267)
(0.5150060296135166 0.5150060296135166 -0.5150060296135166)
(-1.774775089854444 2.225224910145556 0.)
(0. 0.9135133841650713 0.)
(0. 0.2108107809612534 0.)
AGV4's current RTokens are as follows:
(138.2965575833466 172.4818924318884)
(158.7949872182216 193.0808655144177)
(203.4093350613428 236.6345599714884)
(246.6605035672428 281.6605035672428)
(262.4601162078255 317.4601162078255)

AGV3 conceded on TOLB by 3
AGV3's force changes from 0. to 102.2733136857723
the oppenent's force changes from 197.4825410460659 to 137.8884526406389
AGV3's internal modification is as
follow:
(0. -0.3760968692623834 0.)
(0. -0.6268281154373199 0.)
(-1.29544477190376 -1.29544477190376 -1.70455522809624)
(-0.1002882350233847 -0.1002882350233847 0.)
(0. -0.08298355822523718 0.)
(0. -0.07065789492060048 0.)
(0. -0.06257170531102929 0.)
(0. -0.01443962430261081 0.)
AGV3's current RTokens are as follows:
(111.5538827747068 146.5538827747065)
(129.9231379578444 164.9231379578444)
(172.25651147606 203.2032632181156)
(192.6061033944731 231.8701258870415)
(222.914918325595 257.9744575827633)
(241.1269897581563 276.126989758156)
(260.0998206910135 295.0998206910135)
(273.2538047748491 328.2538047748491)
```

```
The conflict-free local schedule for AGV4 is as follow:
(X5 (138.2965575833466 172.4818924318884))
(X10 (158.7949872182216 193.0808655144177))
(X5 (203.4093350613428 236.6345599714884))
(X4 (246.6605035672428 281.6605035672428))
(X9 (262.4601162078255 317.4601162078255))

Scheduling for AGV5.....
The provisional  local schedule for AGV5 is as follow:
(X20 (144.6653585742045 179.6653585742045))
(X15 (158.1089309570075 193.1089309570075))
(X10 (198.9584573111489 243.9584573111489))
(X15 (230.3361747024533 265.3361747024533))
(X14 (265.9380085620597 300.9380085620597))
(X19 (278.8318481297061 328.8318481297061))
Initial state of negotiation between AGV5 and AGV2:
(1) They have reserved RTokens at station X15;
(2) The two Rtokens overlap with a differential of 26.01009779859504;
(3) AGV5 will concede on TOLB side;
(4) AGV2 will concede on TOLA side;
(5) AGV5's RToken is (158.1089309570075 193.1089309570075);
(6) AGV2's RToken is (167.0988331584125 198.5012031835095);
Now the negotiation starts ....
AGV5 conceded on TOLB by 13
AGV5's force changes from 0. to 476.2319668087947
the oppenent's force changes from 765.6866635993969 to 454.9166271914877
AGV5's internal modification is as
follow:
(-2.4546156119128 -4.775195262796672 0.)
(0. -8.237680331911974 -4.762319668088026)
(0. -1.528237020550421 0.)
(0. -1.285939034121014 1.989519660128281e-13)
(0. -0.5279118140075525 0.)
(0. -0.1218258032326389 0.)
AGV5's current RTokens are as follows:
(142.3447789233206 174.8901633114078)
(149.8712506250956 180.1089309570075)
(197.4302202905985 242.4302202905985)
(229.0502356683322 264.0502356683324)
(265.4100967480522 300.4100967480522)
(278.7100223264734 328.7100223264734)

AGV2 conceded on TOLA by 14
AGV2's force changes from 143.9051989961156 to 479.4834635023561
the oppenent's force changes from 1032.096666059611 to 433.9377659511263
AGV2's internal modification is as
follow:
(0. 0.01548448036990635 0.)
(0. 0.02580746728312988 0.)
(0. 0.05333543238504035 -0.05333543238504035)
(0. 0.4719899016441218 -0.4719899016441218)
(-8.389456612656204 5.610543387343796 -5.610543387343796)
(0. 5.0317130418548 0.)
(0. 4.654151218039885 0.)
(0. 2.221558103760799 0.)
(0. 0.5126672547140743 0.)
AGV2's current RTokens are as follows:
(52.60827502504358 87.60827502504358)
(67.68045837507259 102.6804583750726)
(95.20628064181656 130.0763297114576)
(109.4026295388318 150.7770290879967)
(181.0988331584125 198.5012031835095)
(191.7867487280065 226.7867487280065)
(210.7468913196548 245.7468913196548)
(249.133314566529 284.133314566529)
(259.1846110538144 319.1846110538144)

Initial state of negotiation between AGV5 and AGV1:
```

```
(1) They have reserved RTokens at station X15;
(2) The two Rtokens overlap with a differential of 17.16217117436298;
(3) AGV5 will concede on TOLA side;
(4) AGV1 will concede on TOLB side;
(5) AGV5's RToken is (149.8712506250956 180.1089309570075);
(6) AGV1's RToken is (129.9204173784613 167.0334217994586);
Now the negotiation starts ....
AGV5 conceded on TOLA by 6
AGV5's force changes from 0. to 339.7838842868358
the oppenent's force changes from 362.4444475580057 to 264.7805241081946
AGV5's internal modification is as
follow:
(1.239124360538881 1.239124360538881 0.)
(-3.397838842868396 2.602161157131604 -1.45667387157232)
(0. 0.4827474305310488 0.)
(0. 0.4062090868064274 5.684341886080801e-14)
(0. 0.1667595198468916 0.)
(0. 0.0384829661185222 0.)
AGV5's current RTokens are as follows:
(142.3447789233206 176.1292876719467)
(155.8712506250956 181.2544182425668)
(197.9129677211295 242.9129677211295)
(229.4564447551387 264.4564447551389)
(265.5768562678991 300.5768562678991)
(278.748505292592 328.748505292592)

AGV1 conceded on TOLB by 12
AGV1's force changes from 83.09028559041573 to 266.2754790412897
the oppenent's force changes from 1049.509924230323 to 286.5120857020236
AGV1's internal modification is as
follow:
(0. -0.2835489421120982 -7.815970093361102e-14)
(0. -0.472581570186847 0.)
(0. -0.9766685783860964 0.)
(0. -1.011324560199853 1.70530256582424e-13)
(-2.840740327456245 -2.840740327456245 -9.159259672543755)
(-1.915992964202303 -1.915992964202303 1.915992964202303)
(-0.1229029923018174 -0.1229029923018174 0.1229029923018174)
(0. -0.05045491262913515 3.126388037344441e-13)
(0. -0.01164344137583839 0.)
AGV1's current RTokens are as follows:
(40.99218837141403 75.99225514540134)
(59.31986879243955 94.31986879243955)
(93.52746174175857 128.5288788341613)
(106.3481641114757 141.3655938007876)
(129.9204173784613 155.0334217994586)
(150.7689759594199 157.93088626259)
(190.898073384445 219.898073384445)
(225.7619128887861 260.7619128887857)
(272.4835183589504 327.4835183589504)

Initial state of negotiation between AGV5 and AGV3:
(1) They have reserved RTokens at station X10;
(2) The two Rtokens overlap with a differential of 33.95715816591198;
(3) AGV5 will concede on TOLA side;
(4) AGV3 will concede on TOLB side;
(5) AGV5's RToken is (197.9129677211295 242.9129677211295);
(6) AGV3's RToken is (192.6061033944731 231.8701258870415);
Now the negotiation starts ....
AGV5 conceded on TOLA by 16
AGV5's force changes from 0. to 711.326268444113
the oppenent's force changes from 1341.565885668199 to 709.444255647965
AGV5's internal modification is as
follow:
(0.2288430210014667 0.2288430210014667 0.)
(0.4805703441033984 0.4805703441033984 -0.4805703441033984)
(-7.113262684440997 8.886737315559003 0.)
(0. 7.477768334615575 -2.842170943040401e-14)
```

```
(0. 3.069820684737351 0.)
(0. 0.7084201580164518 0.)
AGV5's current RTokens are as follows:
(142.3447789233206 176.3581306929482)
(155.8712506250956 181.2544182425668)
(213.9129677211295 251.7997050366886)
(236.9342130897543 271.9342130897545)
(268.6466769526364 303.6466769526364)
(279.4569254506084 329.4569254506084)

AGV3 conceded on TOLB by 18
AGV3's force changes from 0. to 728.7466424745692
the oppenent's force changes from 1509.663662820295 to 709.4216108206672
AGV3's internal modification is as
follow:
(0. -0.1095384690827075 0.)
(0. -0.18256411513795 0.)
(-0.3772991712851876 -0.3772991712851876 0.3772991712851876)
(-5.854222625423716 -5.854222625423716 -12.14577737457628)
(-1.219797777487003 -4.32847967443189 -1.70530256582424e-13)
(0. -3.062091064984827 0.)
(0. -2.711661024850514 0.)
(0. -0.6257679288116265 0.)
AGV3's current RTokens are as follows:
(111.4443443056241 146.4443443056238)
(129.7405738427065 164.7405738427065)
(172.25651147606 203.2032632181156)
(192.6061033944731 213.8701258870415)
(219.8062364286501 253.6459779083312)
(238.0648986931715 273.0648986931712)
(257.388159666163 292.388159666163)
(272.6280368460374 327.6280368460374)

Initial state of negotiation between AGV5 and AGV3:
(1) They have reserved RTokens at station X15;
(2) The two Rtokens overlap with a differential of 16.711764818577;
(3) AGV5 will concede on TOLA side;
(4) AGV3 will concede on TOLB side;
(5) AGV5's RToken is (236.9342130897543 271.9342130897545);
(6) AGV3's RToken is (219.8062364286501 253.6459779083312);
Now the negotiation starts ....
AGV5 conceded on TOLA by 8
AGV5's force changes from 0. to 368.3231212567395
the oppenent's force changes from 691.3713869975343 to 360.4086696529255
AGV5's internal modification is as
follow:
(0.0420765038738864 0.0420765038738864 0.)
(0.08836065813511595 0.08836065813511595 -0.08836065813511595)
(1.633970900435003 1.633970900434974 0.)
(-3.683231212567421 4.316768787432608 0.)
(0. 2.640654931694826 0.)
(0. 0.6093819073141731 0.)
AGV5's current RTokens are as follows:
(142.3447789233206 176.4002071968221)
(155.8712506250956 181.2544182425668)
(213.9129677211295 253.4336759371235)
(244.9342130897543 276.2509818771869)
(271.2873318843313 306.2873318843313)
(280.0663073579226 330.0663073579226)

AGV3 conceded on TOLB by 9
AGV3's force changes from 0. to 372.1536416716735
the oppenent's force changes from 780.6712633784989 to 354.6802780408091
AGV3's internal modification is as
follow:
(0. -0.02025499094870042 0.)
(0. -0.03375831824794773 0.)
(-0.06976719104551421 -0.06976719104551421 0.06976719104551421)
```

```
(-1.082516738481218 -1.082516738481218 1.082516738481218)
(-2.797439305472096 -2.797439305472096 -6.202560694527904)
(0. -1.01628477086993 0.)
(0. -0.7686004072656942 0.)
(0. -0.1773693247536698 0.)
AGV3's current RTokens are as follows:
(111.4240893146754 146.4240893146751)
(129.7068155244585 164.7068155244585)
(172.25651147606 203.2032632181156)
(192.6061033944731 213.8701258870415)
(219.8062364286501 244.6459779083312)
(237.0486139223015 272.0486139223012)
(256.6195592588973 291.6195592588973)
(272.4506675212838 327.4506675212838)

The conflict-free local schedule for AGV5 is as follow:
(X20 (142.3447789233206 176.4002071968221))
(X15 (155.8712506250956 181.2544182425668))
(X10 (213.9129677211295 253.4336759371235))
(X15 (244.9342130897543 276.2509818771869))
(X14 (271.2873318843313 306.2873318843313))
(X19 (280.0663073579226 330.0663073579226))
```

Appendix D

Provisional Schedules and Final Schedules

This appendix lists the provisional local schedules and the final schedules (after negotiation) in an example given in Chapter 7.

D.1 Provisional Schedule

```
AGV1:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X29            62.            5.              (42. 77.))
(X24            80.            5.              (60. 95.))
(X25            115.           5.              (95. 130.))
(X20            128.           5.              (108. 143.))
(X15            159.           15.             (129. 174.))
(X10            161.           5.              (141. 176.))
(X15            210.           5.              (190. 225.))
(X14            247.           5.              (227. 262.))
(X13            313.           25.             (273. 328.))
=======================================================

AGV2:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X19          73.              5.            (53. 88.))
(X14          88.              5.            (68. 103.))
(X15          116.             5.            (96. 131.))
(X10          141.             15.           (111. 156.))
(X15          174.             5.            (154. 189.))
(X20          203.             5.            (183. 218.))
(X25          224.             5.            (204. 239.))
(X24          266.             5.            (246. 281.))
(X29          303.             30.           (258. 318.))
=======================================================
```

```
AGV3:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X9             130.            5.              (110. 145.))
(X4             147.            5.              (127. 162.))
(X5             184.            5.              (164. 199.))
(X10            209.            15.             (179. 224.))
(X15            236.            5.              (216. 251.))
(X20            258.            5.              (238. 273.))
(X25            278.            5.              (258. 293.))
(X30            313.            25.             (273. 328.))
=======================================================
```

```
AGV4:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X5             164.            5.              (144. 179.))
(X10            189.            15.             (159. 204.))
(X5             223.            5.              (203. 238.))
(X4             267.            5.              (247. 282.))
(X9             303.            25.             (263. 318.))
=======================================================
```

```
AGV5:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X20            165.            5.              (145. 180.))
(X15            178.            5.              (158. 193.))
(X10            229.            15.             (199. 244.))
(X15            250.            5.              (230. 265.))
(X14            286.            5.              (266. 301.))
(X19            314.            20.             (279. 329.))
=======================================================
```

D.2 Final Results

```
AGV1:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X29            61.             5.                (41. 76.))
(X24            79.             5.                (59. 94.))
(X25            114.            5.                (94. 129.))
(X20            126.            5.                (106. 141.))
(X15            153.            15.               (130. 155.))
(X10            156.            5.                (151. 158.))
(X15            208.            5.                (191. 220.))
(X14            246.            5.                (226. 261.))
(X13            312.            25.               (272. 327.))
=======================================================
```

```
AGV2:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X19             73.            5.              (53. 88.))
(X14             88.            5.              (68. 103.))
(X15             115.           5.              (95. 130.))
(X10             139.           15.             (109. 151.))
(X15             189.           5.              (181. 199.))
(X20             212.           5.              (192. 227.))
(X25             231.           5.              (211. 246.))
(X24             269.           5.              (249. 284.))
(X29             304.           30.             (259. 319.))
=======================================================
```

```
AGV3:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X9              131.          5.              (111. 146.))
(X4              150.          5.              (130. 165.))
(X5              189.          5.              (172. 203.))
(X10             210.          15.             (193. 214.))
(X15             236.          5.              (220. 245.))
(X20             257.          5.              (237. 272.))
(X25             277.          5.              (257. 292.))
(X30             312.          25.             (272. 327.))
=======================================================
```

```
AGV4:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X5             158.           5.              (138. 172.))
(X10            185.           15.             (159. 193.))
(X5             222.           5.              (203. 237.))
(X4             267.           5.              (247. 282.))
(X9             302.           25.             (262. 317.))
=======================================================
```

```
AGV5:
Station       ArrivalTime     OperationTime        RToken
=======================================================
(X20            161.           5.              (142. 176.))
(X15            173.           5.              (156. 181.))
(X10            238.           15.             (214. 253.))
(X15            261.           5.              (245. 276.))
(X14            291.           5.              (271. 306.))
(X19            315.           20.             (280. 330.))
=======================================================
```

Appendix E

Theoretical Analysis on the Occurrence of Conflicts of AGV Local Schedules

E.1 Introduction

In an automated guided vehicle system (AGVs), the scheduling problem is as important as other job shop scheduling problems (JSSP). To make an AGV schedule robust, a intuitive way is tolerant planning and scheduling. However tolerance planning and scheduling brings a new problem to the AGVs, i.e., the contest of resources. In other word, the chance of conflicts among the AGV plans and schedules is bigger than that in the non-tolerant case. In our previous work, according to the nature of AGVs, we take AGVs as distributed problem solving (DPS), more acurately, as distributed artificial intelligence (DAI) problems. The negotiation mechanism is applied to resolve AGV scheduling conflicts, and some experimental results are obtained. In this appendix, the occurrence of conflicts of AGV schedules is discussed from the aspects of experimental and theoretical analysis. The theoretical analysis is useful in guiding us to investigate various negotiation methods, to find the most suitable negotiation strategies in real circumstances.

E.2 Experimental Results

E.2.1 The Outline of the Experimental Method

The experiment is originally aimed at comparing the qualities of the negotiations by different strategies. The by-products are the occurrence of conflicts of AGV scheduling, i.e., the occurrence of conflicts between a newly generated AGV local schedule and all other AGV schedules which are accepted. By accepted is meant that an AGV schedule is conflict-free, or an AGV schedule is not conflict-free but the conflict has been resolved.

To do such an experiment, large quantity of experimental results are expected to be gained so that the experiment is convincing.

- Generating Random Data: In order to do many experiments, a large set of data is needed for generating random factory LAYOUT and random AGV schedules. An executable function $ran-fun(n, s)$ (in C programming) is defined to generate the n random numbers when the seed is s.
- Generating a Layout: Here a layout is described by a set , LAYOUT, of arcs. Each arc is in the following form:

$$arc = (x_i, x_j d_{ij})$$

  where x_i, x_j are stations; d_{ij} is the distance from x_i to x_j. The layout is based on Linn Product Ltd. AGV Layout (see Figure 6.1)

- Generating a route plan RP: A route plan

$$RP = (x_1, x_2, ..., x_i, ...)$$

  Note that

$$(x_i, x_{i+1}, d_{i(i+1)})$$

  must be an arc. Each RP is randomly generated.

- Generating a local schedule: Once a route plan, RP, is given, a set of job can be generated. Each job is specified by the prefered starting time, preferred finishing time, weight (importance), loading/unloading time at each stations. These data are also randomly generated.
- Once job specification is given, then a local schedule can be determined by applying Equation 6.35. Let us denote each ramdomly generated schedule by Local-Sch.

The main points of the experiment is described as follow:

1. Initial States: $n = 0$; $f_c(n) = 0$ (the number of conflicting AGV Local-Sch); $f_d(n) = 0$ (the number of schedules in RT-DB);
2. Generate a Local-Sch;
3. If Local-Sch is conflict-free with all schedules in RT-DB, GOTO 7; GOTO 4 otherwise;
4. $f_c(n) = f_c(n) + 1$;
5. Call negotiation procedures: if the conflict is resolved, put the negotiated schedule into RT-DB and GOTO 6; GOTO 7 otherwise;
6. $f_d(n) = f_d(n) + 1$;
7. $n = n + 1$; GOTO 2

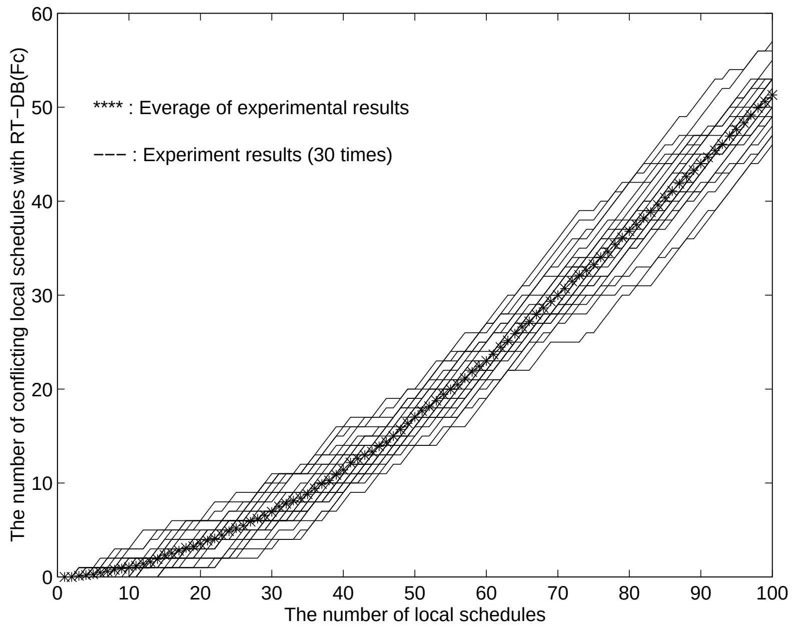


Figure E.1: The Number of Conflict Schedules Out of n Schedules

E.3 The Experiment Results

The experimental results include a large amount of data, such as the quality of negotiations by different negotiation strategies. Of our interest here is the frequency of occurrence of conflict AGV schedules, i.e., the $f_c(n)$. In Figure E.1, we can see the curve of $f_c(n)$. 30 experiments have been done, each time with one hundred local schedules (Local-Schs). The average $f_c(n)$ is also shown in the same figure. It is very useful to know the trend of $f_c(n)$. It indicates how often an AGV local schedule, which is in conflict with one or more schedules in RT-DB, will occur. This can guide each negotiator to choose proper negotiation strategies at different stages.

E.4 Theoretical Analysis

To gain the $f_c(n)$ curve through experiments is time consuming. This motivates us to seek the theoretical statistical analysis of the occurrence of $f_c(n)$. We use following Notations:

m : the number of schedules in RT-DB;

n : the number of total local schedules produced;

$p_c(m)$: the probability of conflict of a newly produced AGV local schedule with RT-DB.

$p_r(m)$: the probability of resolving a conflicting local schedule when m schedules are in RT-DB.

$p(m)$: The probability of conflict of a newly produced local schedule and any single schedule in RT-DB when m schedules are in it.

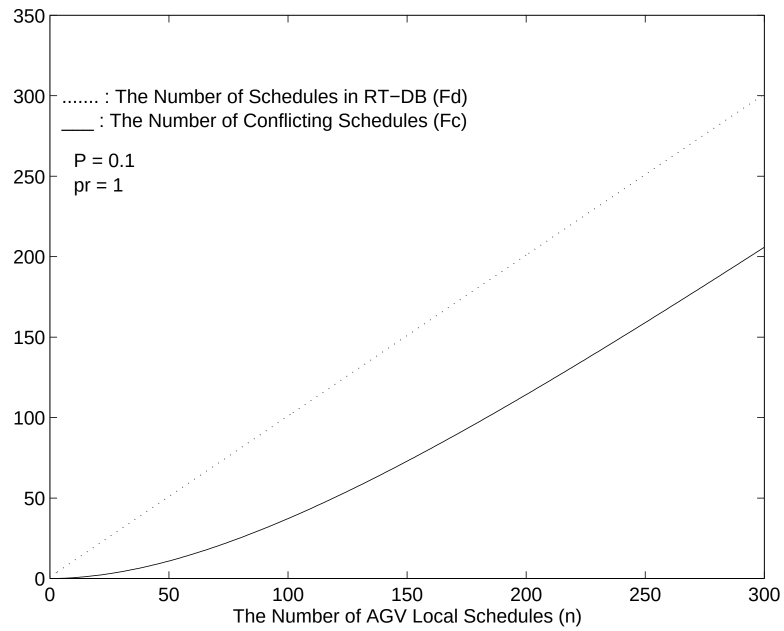


Figure E.2: Theoretical Analysis of $F_d(n)$ and $F_c(n)$: Case 1

$q(m)$: $q(m) = 1 - p(m)$
Now we can derive the expected $f_c(n)$ and $f_d(n)$ statistically. We first investigate two simple cases, then consider the general case.

E.4.1 Simple Case 1

Let's consider the case in which RT-DB will accept any local schedules regardless of whether the schedule is conflict-free or not. Then the following recurrence formula is suitable for calculating $f_c(n)$ and $f_d(n)$:

$$f_d(n+1) = f_d(n) + 1 \tag{E.1}$$

$$f_c(n+1) = f_c(n) + p_c(f_d(n)) \tag{E.2}$$

Equation E.1 is very obvious. Equation E.2 says that the total expected number of conflicts is increased by $p_c(f_d(n))$ when $(n+1)th$ local schedule is produced.
The initial states are:

$$\left.\begin{array}{l} f_d(0) = 0 \\ f_c(0) = 0 \end{array}\right\} \tag{E.3}$$

According to Equation E.1 and Equation E.3, we have:

$$f_d(n+1) = n + 1 \tag{E.4}$$

So the number of AGV schedules in RT-DB increases by 1 as a new schedule is produced. Figure E.2 shows how $f_c(n)$ and $f_d(n)$ vary with n. $f_d(n)$ is always a straight line, while $f_c(n)$ is a curve which depends on the probability of occurrence of conflict of a schedule and RT-DB (see Fig E.2).

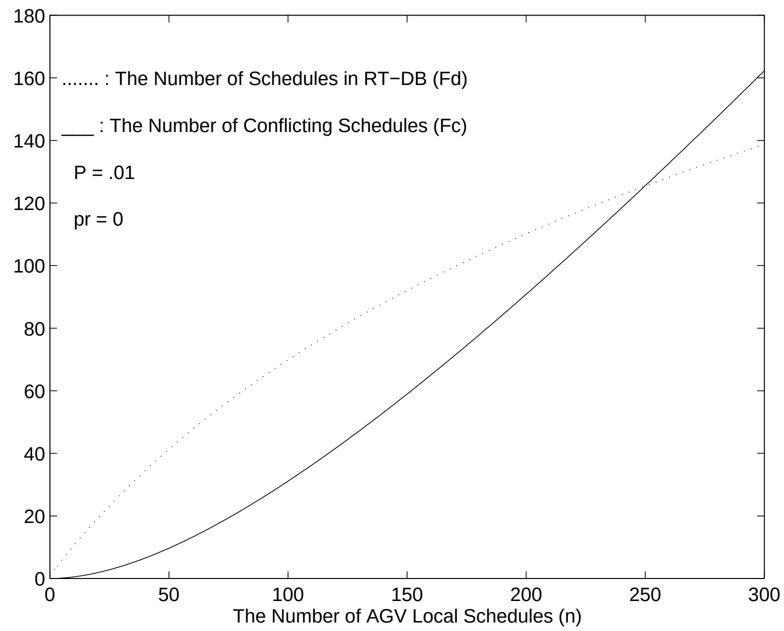


Figure E.3: Theoretical Analysis of $F_d(n)$ and $F_c(n)$: Case 2

E.4.2 Simple Case 2

Another simple case is such that only conflict-free local schedules can be inserted into RT-DB and that a local schedule which is in conflict with RT-DB will be put aside regardless of whether the conflict can be resolved or not. In this case, the following formulas exist:

$$f_d(n+1) = f_d(n) + (1 - p_c(f_d(n))) \tag{E.5}$$

$$f_c(n+1) = f_c(n) + p_c(f_d(n)) \tag{E.6}$$

The initial states are the same as shown in Equation E.3. A property in this case is that the sum of $f_d(n+1)$ and $f_c(n+1)$ is a fixed number $n+1$, i.e.

$$f_d(n+1) + f_c(n+1) = n + 1 \tag{E.7}$$

Fig. E.3 shows the curves of $f_d(n)$ and $f_c(n)$.

E.4.3 The General Case

In the real situation, the two simple cases above are very rare. When a conflict occures, various efforts will be made to resolve it. If it is resolved it will be committed (i.e. to be inserted into RT-DB). More explicitly, if a new local schedule is conflict-free, it will be inserted into RT-DB; it is not, and the conflict is resolved, it will be inserted into RT-DB as well; otherwise put it aside. In this case, we have:

$$f_d(n+1) \quad = \quad f_d(n) + (1 - p_c(f_d(n))) +$$

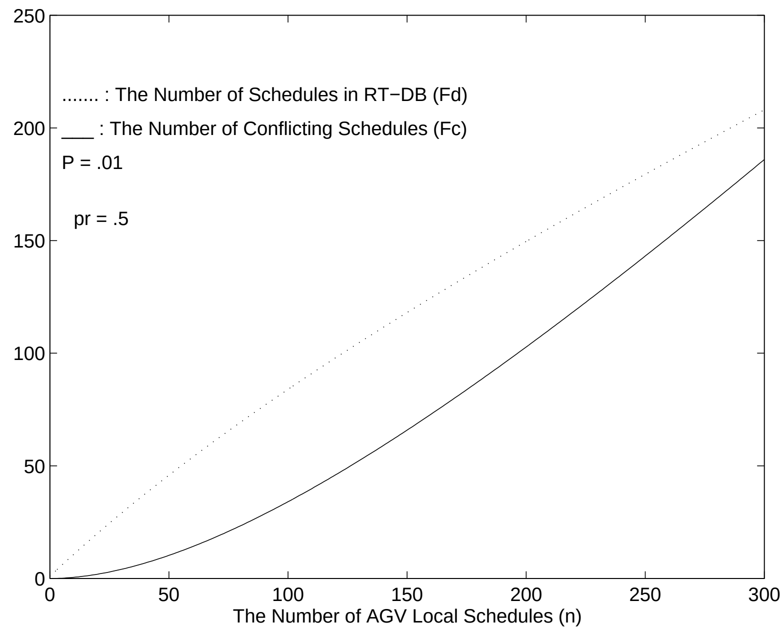


Figure E.4: Theoretical Analysis of $F_d(n)$ and $F_c(n)$: General Case

$$p_c(f_d(n)) * p_r(f_d(n)) \tag{E.8}$$

$$f_c(n+1) = f_c(n) + p_c(f_d(n)) \tag{E.9}$$

Equation E.3 gives the initial states. We can easily find that the two simple cases in Section E.4.1 and Section E.4.2 are two extremes of the general case. In Equation E.8, $p_c(f_d(n)) * p_r(f_d(n))$ represents the probability of conflict of a new schedule and RT-DB and the conflict is resolved.

E.4.4 The properties of $f_d(n)$ and $f_c(n)$

It is obvious that $f_d(n)$ and $f_c(n)$ are both increasing with n. They also depends on $p_r(m)$ and $p_c(m)$. $p_c(m)$ will be further discussed in Section E.5, we will know then that it can be approximatized by a binomial formula. $p_r(m)$ influences on $f_d(n)$ in the following way (see Fig. E.5): $f_d(n)$ is inversely proportional to $p_r(m)$.

$p_r(m)$ influences on $f_c(n)$ in the otherway round (see Fig. E.6).

E.4.5 Summary of the Theoretical Analysis So Far

Equation E.8 and Equation E.9 with the initial state, Equation E.3 are a set of recurrence formulas used to calculate the expected number, $f_c(n)$, of Local-Schs which are in conflict with RT-DB when n Local-Schs have been considered before, and the expected number, $f_d(n)$, of AGV schedules in RT-DB. $f_c(n) - ((n+1) - f_d(n))$ actually reflects the expected Local-Schs which are in conflict with RT-DB and are not resolved. So they are useful in analysing the resource intensity at different negotiation stages.

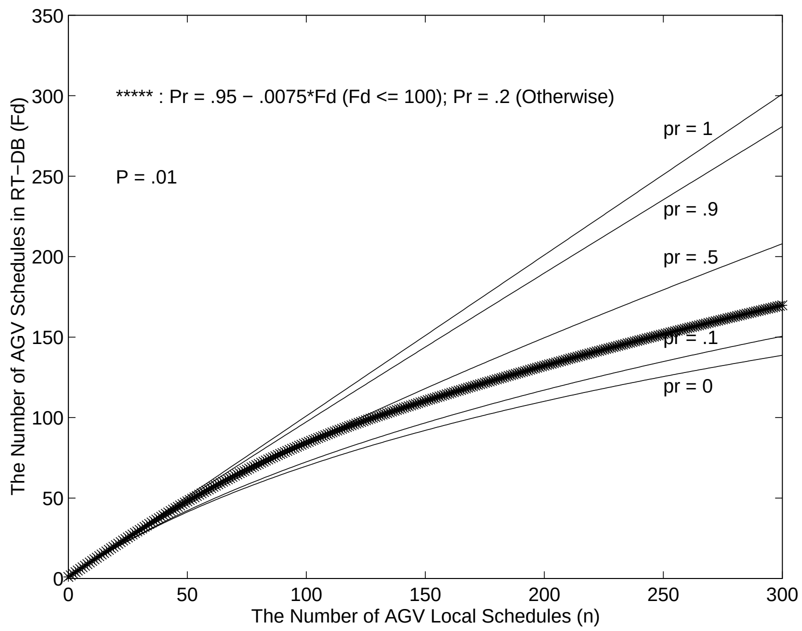


Figure E.5: $f_d(n)$ with different p_r

E.5 Discussion About $p_c(m)$

From Equation E.8, we can see that the $f_d(n)$ and $f_c(n)$ can be solved if $p_c(m)$ and $p_r(m)$ are known. Then what determines $p_c(m)$ and $p_r(m)$? How to calculate them? In this section we will discuss the approximate calculation of $p_c(m)$. There are two obvious properties of $p_c(m)$:

1. $p_c(m)$ monotonically increases with m;
2. $p_c(m)$ positively proportional to the average resource each AGV schedule needs; The more resources a schedule needs, the more possible the conflict of it with other schedules is.

Suppose the probability of a conflict of any schedule in RT-DB and a new local schedule is $p(m)$ when there are m schedules in RT-DB, in addition to these, let's also assume the m Local-Schs in RT-DB are independant, then $p_c(m)$ can be calculated by the following equation:

$$p_c(m) = \sum_{k=1}^{m} p_{m,k} \tag{E.10}$$

$$p_{m,k} = \begin{pmatrix} n \\ k \end{pmatrix} p(m)^k (1 - p(m))^{(m-k)} \tag{E.11}$$

Where $p_{m,k}$ is the Binomial Formula which calculates the probability of conflict of a local schedule (Local-Sch) with k ($k <= m$) in RT-DB when the total number of

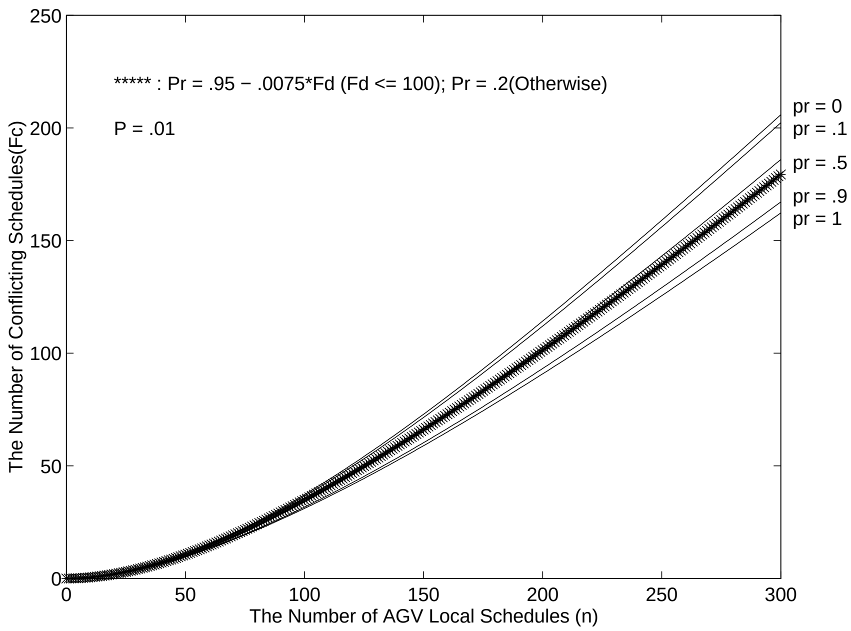


Figure E.6: $f_c(n)$ with different p_r

schedules in RT-DB is m and the probability of conflict of the local schedule with any schedule in RT-DB is $p(m)$. The following formula is equivalent to Equation E.10:

$$p_c(m) = 1 - (1 - p(m))^m = 1 - q(m)^m \tag{E.12}$$

When the negotiation mechanism is applied to resolve conflicts of AGV schedule, $p(m)$ is not a constant, or strictly speaking, $p_c(m)$ no longer obeys the Binomial Formula. This is because of that the allocation of resources is modified during the negotiation: yielding negotiation techniques reduce the resources; shifting negotiation techniques shift the resources while leave the total resource each AGV occupies unchanged. But as a method of approximate calculation, we may still use Equation E.12 to calculate $p_c(m)$. The $p(m)$ is not a constant in general case, but depends on the strategies of negotiation. We have some theorems about this.

Theorem E.1 *If only yielding techniques are used during the negotiation and all the conflicts are resoluable by the technique then $p_c(m)$ can be calculated by Equation E.12 and $p(m) = p(1)$.*

Proof:
As we know, $p(1)$ is the probability of conflict between two random Local-Schs which have never been modified. Suppose we now produce a RT-DB in the following way:

Method 1

1. Initialize RT-DB;

2. Produce a random Local-Sch;
3. Insert Local-Sch into RT-DB, if Local-Sch is conflict-free with RT-DB;
4. Insert Local-Sch into RT-DB, if Local-Sch is not conflict-free with RT-DB;
5. Repeat from 2;

In this case, it is obvious that this is a coin-throw experiment, therefore $p(m) = p(1)$ will be always true, since no Local-Sch is modified from the beginning to the end. So we can denote:

$$p_c(m) = f(p(m), m) = f(p(1), m) \tag{E.13}$$

Here, function $f(p, m)$ calculates the probability of conflict between a new Local-Sch with a RT-DB in which m Local-Schs are located. The RT-DB is produced in Method 1. p is the probability of conflict between Local-Sch and any single schedule in RT-DB.

Now let's look at another method of producing RT-DB:

Method 2

1. Initialize RT-DB;
2. Produce a random Local-Sch;
3. Insert Local-Sch into RT-DB, if Local-Sch is conflict-free with RT-DB;
4. If Local-Sch is not conflict-free with RT-DB, resolve the conflict with the yielding only techniques and then insert the negotiatied Local-Sch into RT-DB;
5. Repeat from 2;

What is different between Method 1 and Method 2 is that Method 1 allows the existence of overlap and Method 2 does not. Let's denote:
S: the overall resource;
S_0: the expected resource each Local-Sch needs;
$S_{m,i}$: the resource assigned to the m AGV s in the RT-DB; i = 1 or 2 represents Method 1 or Method 2 respectively;
$\psi_i(s)$: the allocation of the resource s over S (statistically);
$q_{c,i}(m)$: $q_{c,i}(m) = 1 - p_{c,i}(m)$ (i = 1, 2);

Then we can know that $q_{c,i}$ can be decided by two factors: $S - S_{m,i}$, the unoccupied resource and $\psi_i(S - S_{m,i})$, the allocation of it. We use the following function to represent this:

$$q_{c,i}(m) = \pi(S - S_{m,i}, \psi_i(S - S_{m,i})) \tag{E.14}$$

Obviously, if

$$S - S_{m,1} = S - S_{m,2} \tag{E.15}$$

and

$$\psi_1(S - S_{m,1}) = \psi_2(S - S_{m,2}) \tag{E.16}$$

then,

$$q_{c,1}(m) = q_{c,2}(m) \tag{E.17}$$

exists.

We know:

$$S - S_1^1 = S - S_1^2 = S - S_0 \tag{E.18}$$

$$\psi_1(S - S_0) = \psi_2(S - S_0) \tag{E.19}$$

exists. So

$$q_{c,1}(1) = q_{c,2}(1) \tag{E.20}$$

exists. Now suppose:

$$S - S_{m,1} = S - S_{m,2} \tag{E.21}$$

$$\psi_1(S - S_{m,1}) = \psi_2(S - S_{m,2}) \tag{E.22}$$

exists and there is one Local-Sch coming (randomly). If we use Method 1, then:

$$S - S_{m+1,1} = S - S_{m,1} - S_0 + \theta_1 \tag{E.23}$$

Where θ_1 is the overlapping part between $S_{m,1}$ and the new Local-Sch.
If we use Method 2, then before negotiation, if the new Local-Sch is forced to be put in RT-DB, then:

$$S - S_{m+1,2} = S - S_{m,2} - S_0 + \theta_2 \tag{E.24}$$

And we also have:

$$\theta_2 = \theta_1 \tag{E.25}$$

$$S - S_{m+1,1} = S - S_{m+1,2} \tag{E.26}$$

$$\psi_1(S - S_{m+1,1}) = \psi_2(S - S_{m+1,2}) \tag{E.27}$$

because Equation E.21, Equation E.22 exist. After negotiation, the overlapping is resolved through yielding techniques. But this will not decrease the unoccupied resource (certainly will not increase suppose each AGV is selfish). So Equation E.25 and Equation E.26 will still exist after negotiation. Therefore:

$$q_{c,1}(m+1) = q_{c,2}(m+1) \tag{E.28}$$

will exist according to Equation E.15, Equation E.16 and Equation E.17. So we proved $q_{c,1}(m) = q_{c,2}(m)$ for all m (m =1, 2, ...). Therefore we proved $p_{c,1}(m) = p_{c,2}(m)$ for all m (m =1, 2, ...), that is:

$$p_{c,2}(m) = p_{c,1}(m) = 1 - (1 - p(m))^m \tag{E.29}$$

$$p(m) = p(1) \tag{E.30}$$

This is what we want to prove.

So if only yielding strategies are used in negotiation, then Equation E.12 is a good approximate calculation and $p(m) = p(1)$.

Theorem E.2 *If only shifting techniques are used during the negotiation and all the conflicts are resolvable by the technique, and if $p_c(m)$ is calculated by Equation E.12, then $p(m) \geq p(1)$.*

The proof of the theorem is omitted here, but the reader can think in the following way: (1) If only yielding techniques are used, $p(m) = p(1)$, as we know from Theorem E.1; (2) For any $m(m \geq 2)$, when a conflict occurs, shifting skill is used to resolve it. So the resource required is more than that in the case of yielding strategies. The conflict chance for a new schedule will be higher than that in the latter case and we can easily know from Equation E.12 that the $p(m)$ will be bigger than that in the latter case; (3) Because, in the case of yielding strategies, $p(m) = p(1)$, so in the case of shifting strategies, $p(m) \geq p(1)$.

E.6 Compare Theoretical Results with the Experimental Results

In this section, we take $p(m)$ as constant, $p(m) = p(1)$. Though experiments we obtain the parameters, $p(1)$ and $p_r(m)$. Then we calculate $f_d(n)$ and $f_c(n)$ according to Equation E.8, Equation E.9. Finally the theoretical results of $f_d(n)$ and $f_c(n)$ will be compared with the results from experiments.

E.6.1 Using Experiments to Obtain $p(m)$

Experiment:

- Purpose: Get the experimental data of the probability of conflict between two random local schedules.
- Method: produce 100 pairs of random local schedules each time; check each pair to see if they are in conflict; record the number of pairs with conflict:
  1. Initialization: $i = 0$; $n = 0$;
  2. Get two Local-Schs;
  3. If the two local schedules are in conflict, $i = i + 1$;
  4. $n = n + 1$
  5. If $n < 100$, repeat 2;
  6. End.
- Results: the following result comes from 100 such experiments. They are shown in Table E.1

Let's take Table E.1 as a matrix: $[a_{i,j}]$ where, $a_{i,j}$ is a number which indicates the number of occurrence of conflicting pairs out of 100 pairs of random Local-Schs. So the probability of a conflict of each pairs, i.e., $p(1)$, based on a 100-pair experiment can be approximated as:

$$p_{i,j} = a_{i,j}/100 \tag{E.31}$$

3	1	2	1	1	0	2	0	1	0
1	2	2	0	1	2	2	2	3	0
1	3	0	0	1	2	3	3	2	1
2	2	3	0	1	0	0	3	1	2
2	0	2	0	3	1	1	1	1	1
0	1	0	2	2	1	0	2	2	1
0	1	0	1	3	1	1	2	1	0
0	0	0	2	1	3	2	0	2	1
1	3	0	0	2	0	3	1	1	1
1	1	0	1	0	2	3	0	0	2

Table E.1: The Experiment Results on $p(m)$

Since we have 100 100-pair experiments, then we can calculate the average of $p_{i,j}$:

$$\overline{p_{i,j}} = (\sum_{i=0}^{10}\sum_{j=0}^{10} p_{i,j})/100 = 0.0122 \tag{E.32}$$

$\overline{p_{i,j}}$, as a proximation of $p(1)$, is more objective than any $p_{i,j}$, since it is based on a larger number of experiments. The more experiments are done, the more objective $p(1)$ can be achieved. This is the principle of classical probabity. We can also prove the objectiveness of approximation of $p(1)$ as $\overline{p_{i,j}}$ through following evidence:
As we know, suppose in an experiment, the probabity of event A is p, then if the experiment is repeated n times, the event occurs k times with probablity $p_{n,k}$, according to Bernoulli's formula:

$$p_{n,k} = \begin{pmatrix} n \\ k \end{pmatrix} p^k(1-p)^{n-k}(k = 0, 1, ..., n) \tag{E.33}$$

where, $\begin{pmatrix} n \\ k \end{pmatrix}$ is the binormial coefficient. If $n \gg k$, then $p_{n,k}$ can be approximated as (Poisson Formula):

$$p_{n,k} \simeq \frac{\lambda_k}{k!}e^{-\lambda} \tag{E.34}$$

where $\lambda = np$. In the experiment on $p(m)$, the event interested is the conflict between two Local-Schs; $n = 100$ (pairs of Local-Schs). $k = a_{i,j}(k = 1, 2, ..., 100)$. If we take $\overline{p_{i,j}}$ as p, then:

$$\lambda = 100 \times 0.0122 = 1.22 \tag{E.35}$$

According to Equation E.33 we have:

$$\begin{aligned} p_{100,0} &= 0.295 \\ p_{100,1} &= 0.360 \\ p_{100,2} &= 0.220 \\ p_{100,3} &= 0.089 \\ p_{100,4} &= 0.027 \\ p_{100,5} &= 0.001 \\ p_{100,6} &= 0 \end{aligned} \tag{E.36}$$

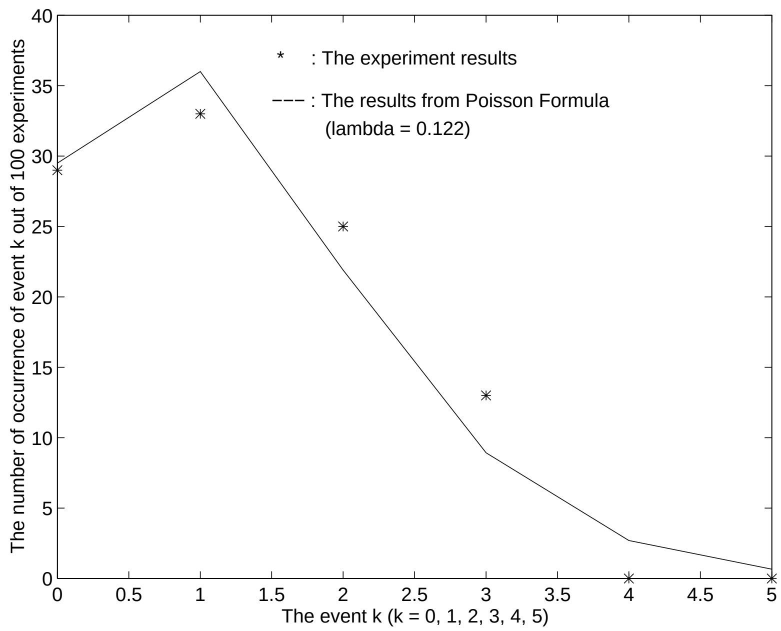


Figure E.7: The probability distribution of $p(1)$

These numbers are shown in Figure E.7 by the solid line.
Now, let's examine Table E.1, we can see:

$$\begin{aligned} p_{100,0} &= 29/100 = 0.29 \\ p_{100,1} &= .33 \\ p_{100,2} &= 0.25 \\ p_{100,3} &= 0.13 \\ p_{100,4} &= 0 \end{aligned} \tag{E.37}$$

These numbers are illustrated in Figure E.7 by asteriks. We can see they fit the Poisson distribution very well.

E.6.2 Using Experiments to Obtain $p_r(m)$

- Purpose: Get the experimental data of the probability of resolving a conflict when there are n schedules in RT-DB. Denote the probability as $p_r(n)$.
- Method: Let the number of schedules in RT-DB be m; produce 100 local schedules each of them in conflict with RT-DB; try to resolve the conflict by negotiation procedures but do not insert any schedule into RT-DB until all 100 schedules are tried; record the number, N_{pr}, of schedules with conflicts resolved.
- Results: we choose: m = 30, 50, 70, 90, and the $N_{pr}(m)$ is listed as follow ($N(m)$ is the number of Local-Schs out of which 100 and only 100 Local-Sch s conflicting with RT-DB are found. This date is used in Section E.5). The results are shown in Table E.2 and Fig. E.8

	m	Npr(m)	pr(m)	N(m)	pc(m)	p(m)
*EXP15	15	94	.94			
EXP30	30	85	.85	246	0.407	0.0173
*EXP40	40	81	.81			
EXP50	50	74	.74	192	0.521	0.0146
*EXP60	60	71	.71			
EXP70	70	66	.66	168	0.595	0.0128
EXP90	90	61	.61	129	0.775	0.0164
EXP90	90	61	.61	140	0.714	0.0138

* The data are obtained in an additional experiment but they are not used in the following discussion.

Table E.2: The Experiment Results on $p_r(m)$

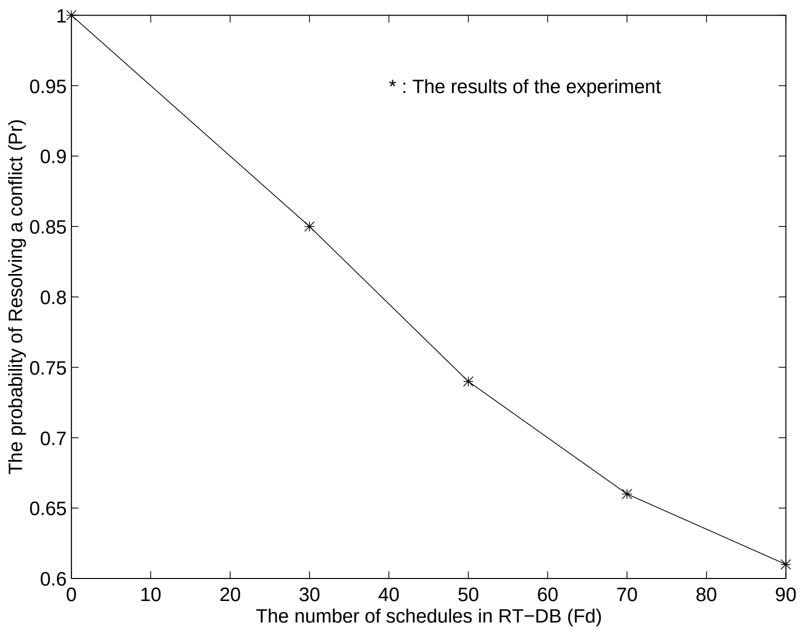


Figure E.8: The experiment results $p_r(m)$

E.6.3 Compare Theoretical Results with the Experimental Results

The lines drawn by asterisks in Fig E.9 and Fig E.10 are respectively the average of the results from the 30 times experiments on $f_d(n)$ and $f_c(n)$. In Fig. E.10 and Fig. E.9, $p_r(m)$ comes from the experiment on $p_r(m)$ (see: Section E.6.2):

$$p_r(m) = \begin{cases} 1 - \frac{1-0.85}{30}m & 0 \leq m < 30 \\ 0.85 - \frac{0.85-0.74}{20}m & 30 \leq m < 50 \\ 0.74 - \frac{0.74-0.66}{20}m & 50 \leq m < 70 \\ 0.66 - \frac{0.66-0.61}{20}m & otherwise \end{cases} \tag{E.38}$$

We firstly assume that $p(m) = p(1)$ and $p(1)$ comes from the experiment on $p(1)$ (see: Section E.6.1). We can see from the two figures that: the theoretically calculated $f_d(n)$ is very close to that from the experiment; but for $f_c(n)$, the error of the theoretical calculation and the experiment increases with n. The reason can be given by Theorem E.2 which states that $p(m) \geq p(1)$ in the case of negotiation with pure shifting strategies.

Through experiments, we find that when $p(m) = 0.175$, both $f_d(n)$ and $f_c(n)$ from the theoretical calculation are respectively very close to that from the experiment. The dotted line in Fig. E.10 also shows the theoretical calculation with $p(m) = 0.0145$ which comes from the average of p(m) given in Table E.2:

$$p(m) = \frac{0.0122 + 0.0173 + 0.0146 + 0.0128 + 0.0164 + 0.0138}{6} \tag{E.39}$$

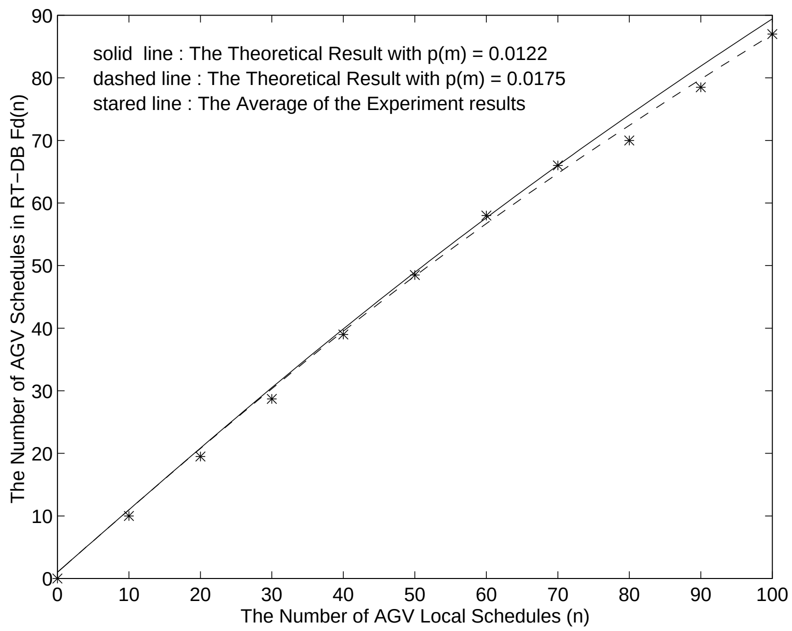


Figure E.9: The Theoretical Results and the Experiment Results of $f_d(n)$ with Mixed Strategies

E.6.4 Negotiation with pure strategy

We can see from Figure E.10, if we use $p(1)$ $(= 0.0122)$ to approximate $p(m)$, the error of the theoretical result and experiment result of $f_c(m)$ increases with $p(m)$. This is caused mainly by the shifting strategies according to Theorem E.1 and Theorem E.2. This can also be proved by experiments. Figure E.11 and Figure E.12 are the experiment results with respectively pure shifting strategies and pure yielding strategies. Figure E.13 and Figure E.14 are showing the comparison of theoretical results and experiment results. We can conclude from the comparison that the error in Figure E.10 is mainly due to shifting strategies.

E.7 Conclusion

In this appedix, the theoretical analyses of the occurrence of conflict schedules $(f_d(n))$ are discussed in various aspects. The experiment results are achieved which, as are seen, are very close to the theoretical results. Those analysis methods as well as the experiment results may be used in further studies on AGV negotiation.
We also made some conjectures which have not been strictly proven. But the experiment results strongly support them.

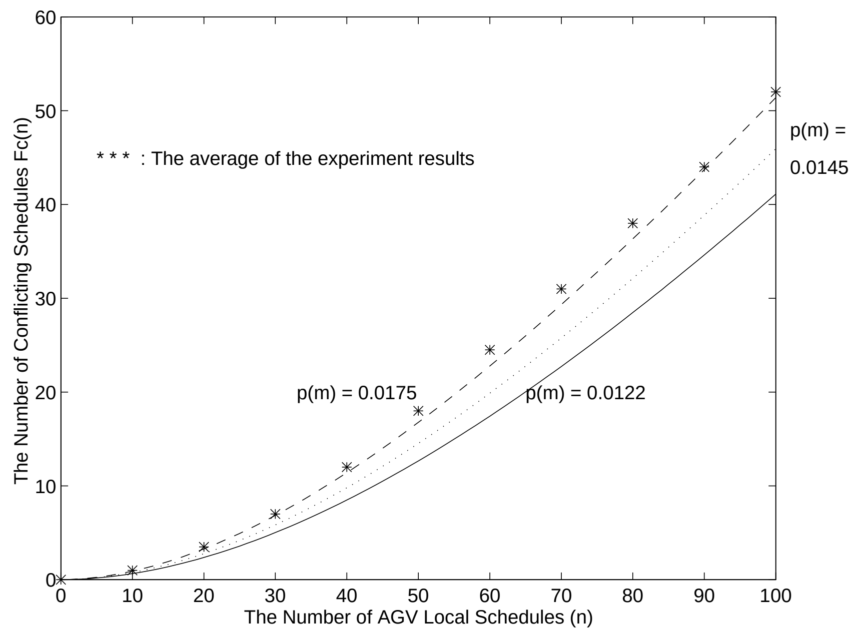


Figure E.10: The Theoretical Results and the Experiment Results of $f_c(n)$ with Mixed Strategies

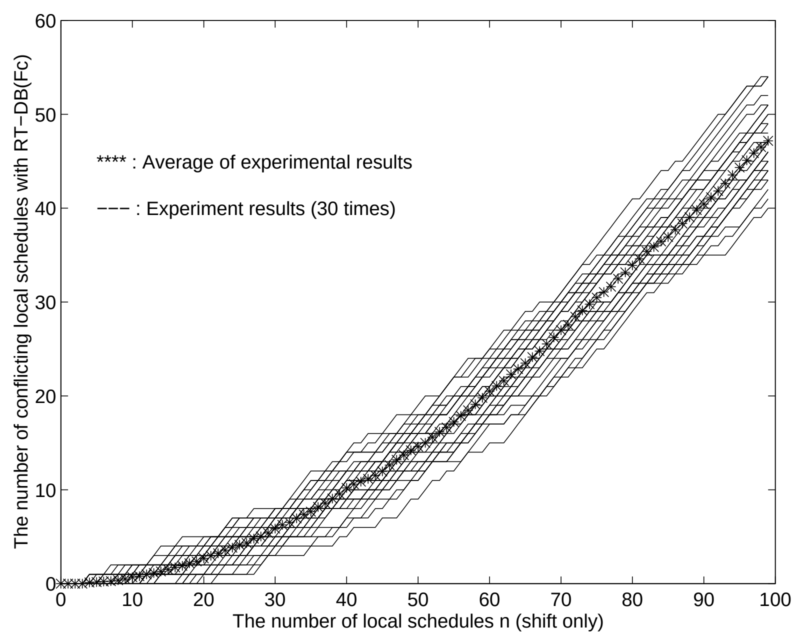


Figure E.11: Experiment Results of $f_c(n)$ with Pure Strategy of Shifting

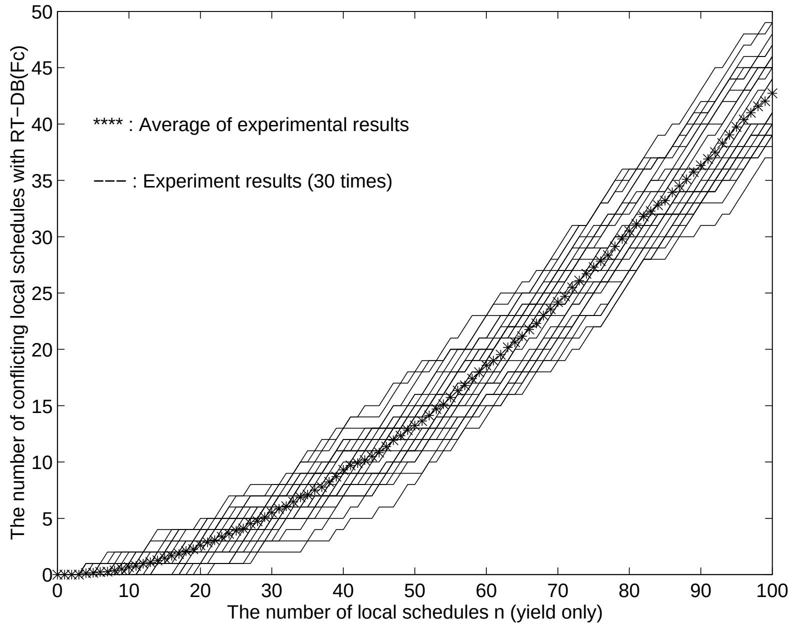


Figure E.12: The Experiment Results of $f_c(n)$ with Pure Strategy of Yielding

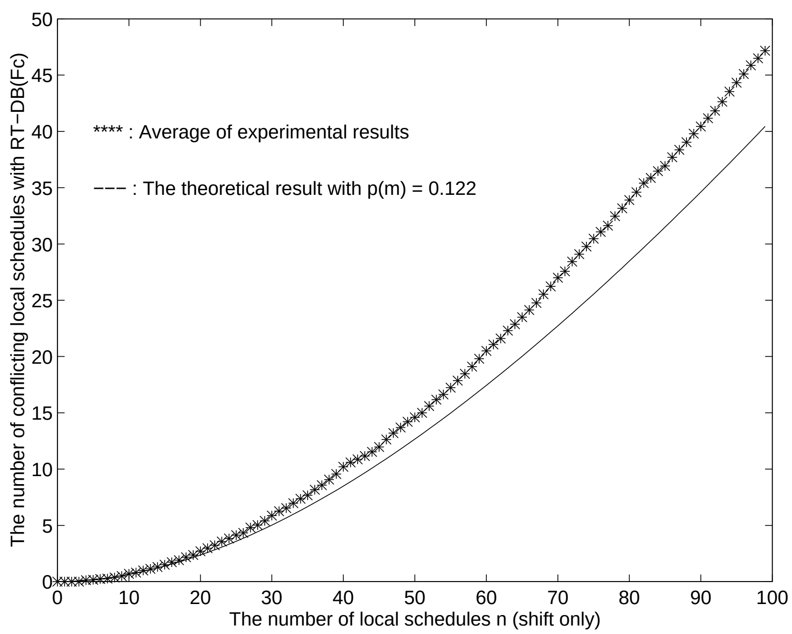


Figure E.13: The Theoretical Results and the Experiment Results of $f_c(n)$ with Pure Strategy of Shifting

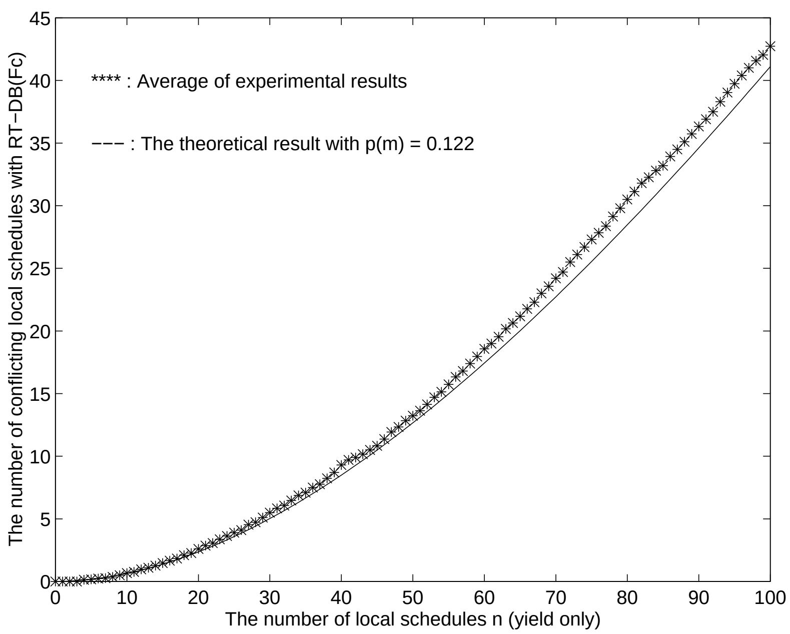


Figure E.14: The Theoretical Results and the Experiment Results of $f_c(n)$ with Pure Strategy of Yielding