Using Distributed Agents to Create University Course Timetables Addressing Essential & Desirable Constraints and Fair Allocation of Resources

PENNEE WANGMAETEEKUL

Using Distributed Agents to Create University Course Timetables Addressing Essential & Desirable Constraints and Fair Allocation of Resources

Pennee Wangmaeteekul

A Thesis presented for the degree of

Doctor of Philosophy

School of Engineering & Computing Sciences

Durham University

December 2011

Abstract

In this study, the *University Course Timetabling Problem (UCTP)* has been investigated. This is a form of *Constraint Satisfaction Problem (CSP)* and belongs to the *NP-complete* class. The nature of a such problem is highly descriptive, a solution therefore involves combining many aspects of the problem. Although various timetabling algorithms have been continuously developed for nearly half a century, a gap still exists between the theoretical and practical aspects of university timetabling.

This research is aimed to narrow the gap. We created an agent-based model for solving the university course timetabling problem, where this model not only considers a set of essential constraints upon the teaching activities, but also a set of desirable constraints that correspond to real-world needs. The model also seeks to provide fair allocation of resources. The capabilities of agents are harnessed for the activities of decision making, collaboration, coordination and negotiation by embedding them within the protocol designs. The resulting set of university course timetables involve the participation of every element in the system, with each agent taking responsibility for organising of its own course timetable, cooperating together to resolve problems. There are two types of agents in the model; these are *Year-Programme Agent* and *Rooms Agent*. In this study, we have used four different principles for organising the interaction between the agents: *First-In-First-Out & Sequential (FIFOSeq)*, *First-In-First-Out & Interleaved (FIFOInt)*, *Round-Robin & Sequential (RRSeq)* and *Round-Robin & Interleaved (RRInt)*. The problem formulation and data instances of the third track of *the Second International Timetabling Competition (ITC-2007)* have been used as benchmarks for validating these implemented timetables. The validated results not only compare the four principles with each other; but also compare them with other timetabling techniques used for *ITC-2007*.

The four different principles were able to successfully schedule all lectures in different periods, with no instances of two lectures occupying the same room at the same time. The lectures belonging to the same curriculum or taught by the same teacher do not conflict. Every lecture has been assigned a teacher before scheduling. The capacity of every assigned room is greater than, or equal to, the number of students in that course. The lectures of each course have been spread across the minimum number of working days with more than 98 percent success, and for more than 75 percent of the lectures under the same curriculum, it has been possible to avoid isolated deliveries. We conclude that the *RRInt* principle gives the most consistent likelihood of ensuring that each *YPA*

in the system gets the best and fairest chance to obtain its resources.

Declaration

No part of the material presented in this thesis has previously been submitted by the author in support of an application for another degree or qualification of this or any other university or other institute of learning. All the work presented here is the sole work of the author and no one else.
This research has been documented, in part, within the following publications:

- Wangmaeteekul, P. and Budgen, D. 2011. *Using Agents to Create a University Timetable Addressing Essential & Desirable Constraints and Fair Allocation of Resources.* In Proceeding on *IADIS International Conference Intelligent Systems and Agents 2011.*

To my beloved grandfather and father
Changjun Wang &Yuanqin Wang
who were brave and dedicated their whole life
for their family—"*Wangmaeteekul*"

Acknowledgements

This PhD thesis is unable to accomplish if lack of many parts participated in. So, I would like to use this opportunity to thank everyone who has contributed him/herself towards this research.

- My father who supported and gave me maths foundation. My mother who looks after her family the best. My younger sister & two younger brothers who always share both good and bad times together.
- My two great supervisors Prof. David Budgen and Dr. Rafael Bordini who always stand beside me and support me to achieve the goal.
- My second supervisor Prof. Malcolm Munro who helped me finding out the weaknesses of my work and Dr. Andy Hatch who advised me to find some datasets in order to prove my work.
- Prof. Nikolay Mehandjiev who gave me many valuable comments to improve the menuscript.
- Dr. Nick Holliman and Dr. Shamus Smith who questioned many valuable points which lead to review more literature.
- My colleague Dr. Patricia Shaw who gave an idea to apply negotiation in my work.
- My teachers who have dedicated their teaching and my friends who keep supporting and cheer me up.
- The financial support from school of Engineering and Computing Sciences, Durham University; which gave me a good chance to join

  (1) The European Agent Systems Summer School 2008 in Lisbon, Portugal.

  (2) IADIS International Conference Intelligent Systems and Agent 2011 in Rome, Italy.
- The Royal Thai government which gives the financial support and Prince of Songkhla University which allows me to leave for study in UK.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter starts by describing the motivation for this research—namely investigating the use of agents to address the highly-complex timetabling problem. Then it describes why the significant autonomous properties of agents make them suitable for solving the timetabling problem. The research method is described, the intelligent-agents architecture is proposed, and the approach to validating the results is presented. The chapter ends by identifying the contributions of this research, together with an overview of the structure of the thesis.

1.1 Motivation

In the real world, we often encounter situations which need us to make a sequence of decisions to solve a problem. An example of such a problem, which we are used to, and that has been solved many times in our daily life is:

> You buy something for 20 pounds; and assume that in your wallet you have one twenty- pound note, two ten-pound notes and three five-pound notes. There are many ways to pay. The set of possible answers is {(20), (10,10), (10,5,5)}.

Although it seems we are able to solve such a problem easily, we actually need to perform many steps to formulate different aspects of the problem until reaching a satisfactory result. This type of problem can be formulated in terms of a set of requirements that some specific properties are met. The answers that 'satisfy' this are the set which meet all of the requirements; such that in each step of the decision-making a part of the answer

is met. On the other hand, the many possible answers which do not match the needs need to be rejected; that means in each step of the decision-making if any element of the answer is not met, then the answer is an unsatisfied one. The class of problems that we can formulate in this manner is called *Constraint Satisfaction Problems (CSPs)*. A *CSP* is a highly descriptive problem for which a solution involves combining many aspects of the problem together, and is one that can be viewed as a high-dimensional combinatorial problem (Apt, 2003). Currently it is a highly-active research field which can be applied to many problem domains such as those identified in (Apt, 2003) which are listed below:

- *Interactive Graphic Systems:* used for interpreting objects in 3D scenes—scene analysis.
- *Operating Research Problems*: often used in scheduling problems and optimization problems.
- *Molecular Biology*: applied to DNA sequencing, construction of 3D models of proteins.
- *Business Applications*: implemented in product scheduling systems, and in stock trading systems.
- *Electrical Engineering*: employed to locate faults in circuits, to plan the circuit layouts, and to test and verify the circuit design.
- *Numerical Computing*: applied in applications that involve solving polynomial constraints with guaranteed precision.
- *Natural Language Processing*: taking a major role in implementing efficient parsers.
- *Computer Algebra*: deployed for solving and/or simplifying equations over algebraic structures.
- *Database System*: used in ensuring and/or restoring data consistency in databases.

As a constraint satisfied problem is a non-trivial problem, very long time may be needed to solve such a problem when using an exhaustive search (Yang and Paranjape, 2011). This complexity of *CSPs* inspired us to conduct this research to investigate a way to simplify such problems. In order to conduct the study, we needed to choose one problem domain to be a representative of *CSPs*, and chose the university course timetabling

problem. This is because not only is this the domain we are used to, but also it contains both great complexity and a variety of requirements. The evident interest in this topic is demonstrated by the duration of this research field which has posed a challenge for various scientific communities from different disciplines such as *Operation Research*, *Artificial Intelligence*, and *Multi Agents* for almost half a century. Timetabling involves a wide set of activities which contribute towards making a timetable. As defined *Collins Concise Dictionary* (4th Edition), this means a table of events arranged according to the time when they take place. Consequently, it is a timetable which specifies which people are to meet, at which location and at what time.

Educational timetabling is one of the subclasses in timetabling and is in the class of real-world *NP-complete* problems (Even et al., 1975). All events described in such a table take place in educational institutes. It involves scheduling a set of resources over limited lengths of time, and subject to certain conditions (which we term *Essential & Desirable* constraints). The resources are students, staff, rooms, courses, time and equipment. From the definition provided by Burke, Kindston and de Werra(2004), this is a problem which can be defined in term of four parameters: a finite set of timeslots; a finite set of resources; a finite set of meetings; and a finite set of constraints. The principal challenge of this problem is how to assign times and resources to the meetings in order to satisfy as many of the constraints as possible.

Although a range of potential algorithms have been proposed for nearly a half century of research into timetabling, such as *Graph Colouring Algorithms*, *Simulated Annealing*, *Tabu Search*, *Genetic Algorithms* and *Constraint-Based Programming*-one recent research paper (McCollum, 2007) observes that a gap still remains between the theoretical and practical aspects of university timetabling. In order to evaluate particular techniques and approaches, automated timetabling usually ignores the human factors and deals with only the data sets. However, for a workable timetable we need to satisfy the essential constraints imposed by the teaching activities and also the desirable constraints that correspond to the real-world needs of both lecturers and students.

1.2 Agents and the Timetabling Problem

A number of papers have argued that agent technology offers a suitable mechanism for addressing timetabling problems (Causmaecker et al., 2002,Causmaecker et al., 2003,Carter, 2000). As an intelligent agent is capable of autonomous action, it is able to work on behalf

of its representative to achieve its goals. As described in (Causmaecker et al., 2003), by exploiting agents' characteristics, a set of agents could be used to resolve the conflicting expectations and constraints in the timetabling problem by negotiating and collaborating with each other. Furthermore, Carter(2000) has suggested that a distributed software architecture can adequately describe the timetabling problem. This is because agents can provide a model of distributed timetabling, with the distributed aspect arising from the presence of separated, autonomous components which do not communicate in every detail, but do need to exchange more coarse grained information. The solutions come from the collected information which is obtained from the expressions of interest and degrees of agreement of individual agents. For real world operators, better decisions are usually obtained through a *negotiation* process in which all partners actively search for better solutions and find ways to alleviate another partner's problems. Hence, given the significant and powerful properties which are provided by agent technology, it is clear that applying a multiagent model to the scheduling timetable problem domain has the potential to cope with difficult situations and address real world needs.
Agent-based technology has been employed by researchers to address the university timetabling problem over the last decade, for example:

- Kaplansky and Meisels (2004) proposed a model that assigned agents to the roles of *Scheduling Agents* and *Room Agent*.
- Strnad and Guid (2007) defined a model which is comprised of three sorts of agents working together—*Course Agents*, *Scheduling Agents* and *Central Agent*.
- Oprea (2007) designed four types of agents, working collaboratively according to the university's organization levels, that are *Person Agents*, *Department Agents*, *Faculty Agents* and *University Agent*.

Each researcher has proposed a different model that satisfies a different range of constraints when generating timetables for any particular institute. However, some consider only the essential constraints; while others take into account both essential constraints and that subset of the desirable constraints which correspond to their perception of other needs.
For this research we have investigated the use of an agent-based model for solving the university course timetabling problem by generating resource allocation solutions to all participants in a system, under the constraints and formulation which are defined by the

International Timetabling Competition 2007 (ITC-2007), and also seeking to provide fair allocation among them. The distributed timetabling model has been adapted from the model proposed by Kaplansky and Meisels (2004), by adding more flexibility and more roles to each type of agents, which have been redefinded as *Year-Program Agent (YPA)* and *Rooms Agent (RA)* respectively.
We aim to narrow down the existing gap by trying to implement a system imitating the way that human planners work in the real world. Crucial characteristics of the agent for negotiation and collaboration are taken into account through designed protocols for solving conflicts and collaborating between agents just as people do in real life, since having all of the participants actively searching for solutions might alleviate each other's problems. Each agent makes decisions based on the incomplete information available to it, its own selfish manner, its own capability and the circumstances that it meets at any given moment. For this study, we therefore aim to investigate the following two research questions:

> 1) How to apply agent-based technology to allocate the university resources so as to satisfy the essential constraints for serving the teaching activities and to address those desirable constraints that correspond to real world needs?
>
> 2) How to ensure that each agent gets a fair chance to acquire the resources that it needs?

1.3 Research Method

The key features of the process employed for developing multiagent systems involve the use of both *experiments* and *software prototyping* to increase an agent's ability and to add more features to the system on an incremental basis.
This study uses a distributed model for solving university course timetabling problems by defining two types of agents working together in the roles of *Year-Programme Agent* and *a Rooms Agent*.

- A *Year-Programme Agent (YPA)* is assigned to the task of generating the timetable for one level of a particular programme. It is responsible for organizing the subjects/modules/courses which that programme's students have to take in one particular term. The resulting timetable should satisfy all of the essential constraints

and as many as possible of the desirable constraints. A crucial role of the *Year-Programme Agent* is therefore to collaborate with other *YPAs* to resolve any allocation problems by reallocating rooms between *YPAs*.

- The *Rooms Agent (RA)* manages the rooms (resources) and will book the requested room when that room is vacant. It also coordinates the *Year-Programme Agents* to work together in order to avoid overlaps across the shared modules. Moreover, the *Rooms Agent* takes responsibility for ordering access to the resources by the *Year-Programme Agents* in such a way as to ensure fairness.

The total number of agents in this system is therefore the number of *Year-Programme Agents* which are required for the different degree programmes and levels, plus one *Rooms Agent*. This is a flexible architecture in that it makes no assumption about the length of a degree programme (2, 3, 4 years).
The agents model aims to mimic a human scheduler's behaviour in the real world by performing two stages of allocation:

(1) Between *YPAs* and the *Rooms Agent* to provide an initial allocation of resources (the *initial allocation phase* (*S1*)). Two different interactive allocation algorithms that have been investigated in order to optimize the chances for each *YPA* to acquire the desired set of resources, where these are:

- *First-In-First-Out (FIFO)* algorithm
- *Round-Robin (RR)* algorithm

(2) Between *YPAs* to refine the allocation of rooms (the *negotiation phase* (*S2*)).

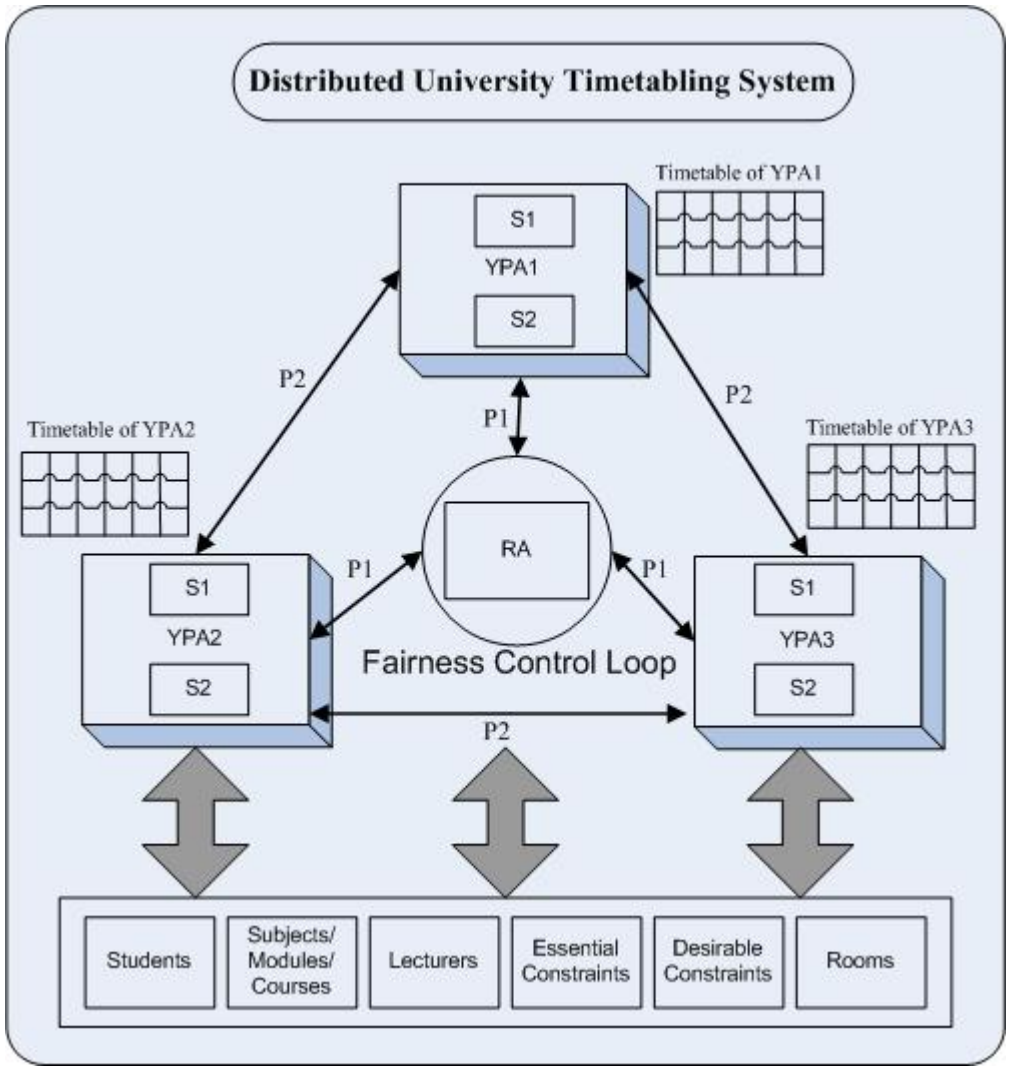

Figure 1.1: An overview of the agent-based architecture

Figure 1.1 shows an example of the architecture used, comprising three *YPAs* and one *RA*. Each *YPA* is composed of two segments of code that perform the actions required for phases *S1* and *S2*, while *P1* and *P2* represent the interaction protocols employed in the initial allocation phase and the negotiation phase.

There are two distinct forms of implementation that have been investigated for producing a set of timetables. These forms through which the *Year-Programme Agents* and *the Rooms Agent* work together to create the set of timetables are:

- ***Sequential***: start by running the initial allocation phase until this phase terminates; and then subsequently run the negotiation phase if these are any constraint-mismatched elements remaining after initial allocation.
- ***Interleaved***: start by running the initial allocation phase, interleaving this with a negotiation phase if any *YPA* is facing a constraint-mismatch problem. After the problem has been resolved, the initial allocation phase will continue.

	Allocation Form	
Organizing Form	**FIFO**	**RR**
Sequential	*FIFOSeq*	*RRSeq*
Interleaved	*FIFOInt*	*RRInt*

Table 1.1: Principles of interaction

In this research we therefore have combined these to create four different principles that have then been used to implement the set of distributed university course timetables. These are: *First-In-First-Out & Sequential (FIFOSeq)*, *First-In-First-Out & Interleaved (FIFOInt)*, *Round-Robin & Sequential (RRSeq)* and *Round-Robin & Interleaved (RRInt)*, as shown in Table 1.1. The investigation has been concerned with exploring the different qualities resulting from the use of these four principles, by comparing the timetabling results in terms of how well the constraints were addressed and the degree of fairness achieved when allocating rooms to *YPAs*. The problem formula and data instances of the third track of *the Second International Timetabling Competition (ITC-2007)* were used as benchmarks for this, in order to aid validation of these organized timetables.

1.4 Contributions

Over the past decade, agent technology has been playing an important role for solving university course timetabling problem (see section 3.4). However, there are several key contributions which make this research different from the others.

1. McCollum (2007) insists that a gap between the theoretical and practical aspects of university timetabling still remains, as most research is directed at evaluating particular techniques and approaches. An automated timetabling ignores the human factors and chooses to deal with only the data sets. Both McCollum and Carter (2000) assert that the process for producing a workable timetable still remains with

a combination of lecturing and administrative staff rather than through the use of an automated component. Much more work is needed to investigate the formulation and modeling of the problem. Therefore, in this research we have employed a model which imitates the timetabling processes that human planners use in the real world; each participant makes its decisions according to its own needs under any circumstance at that given moment. Negotiation and collaboration are embodied within a set of protocols designed for solving conflicts, and for collaborating among the participants in order to reach the best solution. The results help to narrow the gap between theory and practice in university course timetabling research.

2. The datasets which are used in this research are taken from the real world. They have been selected from a large set of interesting cases from a real-world institute—the University of Udine in Italy (Gaspero et al., 2007). The experimental results show that the principles used in our agent model are able to achieve the goal and also satisfy the defined essential & desirable constraints. That means the model could be workable in a real-world situation.

3. In the real world, planners get better decisions through a negotiation process in which all partners actively search for better solutions and find the ways to alleviate other partner's problems. This idea has been adapted and applied to the interaction protocol in the negotiation phase. The testing results show that the number of organized lectures has been increased after applying the negotiation phase following the initial allocation phase. That means the constraint-mismatched problems which are faced by some participants in the system have been resolved through the participation of the other participants. In the literature review, we found that other research that used agents for solving university courses timetabling problem applied negotiation for the purpose of resource allocation, whereas this research has applied it for the purpose of alleviation.

4. Apart from the distributed timetabling model which has been adapted from the model proposed by Kaplansky and Meisels (2004), we also solve the overlap problem between shared subjects in different way. The original one sorted it by defining a negotiation protocol for solving conflict among scheduling agents in order to reach shared course slots, whereas our model assigns an agent that takes responsibility for organising this shared course, while the other shared course agents take a role in avoiding clashes with the shared slots when they are organizing their own timeta-

bles. From the testing results, we found that all lectures have been scheduled and all conflicts have been avoided for hundred percent of cases. That means the way we sorted the overlap problem was successful.

5. The model seeks to allocate resources fairly by applying a *Round-Robin* algorithm over the *RA* controlled loop in the automated timetabling mechanism, in order to ensure that each agent gets an equal chance to access the resources. It is different from other researches under the same problem domain which do not address the issue of fairness; neither those that using agent technology nor those that use classical timetabling techniques. Our research results provide a new model for managing university course timetabling, by considering not only constraints, but also the issue of fairness during timetable implemention.

6. The model has created a base for extending the way that constraints are handled during timetable creation. In this research, four different principles have been investigated for managing allocation of resources and for subsequent negotiation, in order to find practical solutions. Moreover, the validated results not only compare the four principles with each other; but also compare them with other timetabling techniques as well.

1.5 Thesis Outline

This thesis is organized into nine chapters, as follows:

Chapter 2 reviews the background of agent technology. It starts with a description of intelligent agent and multi-agent systems, followed by an agent's architecture —*Beliefs Desires and Intentions* (*BDI*) and the interaction between agents. Then, the concept of game theory is reviewed; and this chapter ends by summarizing multiagent resource allocation.

Chapter 3, the literature investigating in university timetabling is reviewed. This covers the different types of educational timetables, the existing problems in university course timetabling domain, the suitability of applying agents for solving timetabling problems, the survey of published papers that have used agents to address university course timetabling, and the classical methodologies which have been applied to timetabling problems. This chapter ends

with a summary table that compares our model with those used agents by the other researchers.

Chapter 4 describes the reasons for addressing the problem of university course timetabling through the use of an intelligent-agents model, and its evaluation via the defined problem formula from *ITC-2007*. The following sections present the details of intelligent agent architecture, the allocation algorithms, the organizing forms and the university structural scenario defined for *ITC-2007*.

In Chapter 5, the agents and necessary interaction protocols are described in detail. It starts with a description of all modules in *the Year Program Agent (YPA)* and *the Rooms Agent (RA)*, and then follows by the interaction protocol design between *YPA* and *RA* which it has been used in initial resource allocation phase; ending with the interaction protocol design for the negotiation phase between a *YPA* and the other *YPAs*.

Chapter 6 contains information about the benchmark data set. It describes the background of *the International Timetabling Competition (ITC)* and information about *Curriculum-Based Course Timetabling*. The competition problem formula, data instances and file format are described. In addition, it describes the means to validate the timetabling results and includes a set of reference results.

Chapter 7 has been dedicated to presenting the experimental research method. It describes how the experiments are set up according to the defined hypotheses; how the independent variables of each experiment are organised for each of these four different principles; and finally how the dependent variables are measured.

Chapter 8 discusses the experimental outcomes. It explains the relation between causes and effects of each hypothesis; then, analyses and interpretes the experimental results. The research questions are also answered at the end of this chapter.

Finally, Chapter 9 presents the conclusions drawn from this thesis. Some suggestions are given regarding possible ways to apply the results in other problem domains or to extent then by adding more constraints to the model.

Chapter 2

Background in MultiAgent System

This chapter is concerned with the fundamental theories involved in developing an intelligent agents system. It firstly defines what intelligent agent and multiagent system are; then following with a discussion about agent architecture—*BDI* and *Jason* are introduced. Then, we discuss interaction among agents and the concept of game theory in the following section. At the end of the chapter, an overview of multiagent resource allocation is provided.

2.1 Intelligent Agent

An "*intelligent agent*" is defined as a computer system which is capable of autonomous action to interact with its environment in order to meet its goal (Wooldridge, 2002).
A "*multiagent system* (*MAS*)" is a reactive system which is composed of multiple intelligent agents that maintain an interaction with their environment. Each agent is an individual module in a concurrent system that means each agent is a reactive subsystem, interacting with its own environment which largely consists of other modules (Wooldridge, 2002,Pnueli, 1986).
Agents are capable of sensing their environment via sensors and have a group of possible actions that they can perform via effectors or actuators in order to modify the environment. The processes connectivity the input and output are plans which an agent uses to decide what to do in order to achieve its goals. According to the definition of a rational agent, provided by Wooldridge and Jennings in (1995), a rational agent should have the following properties:

- *Autonomy*: the agent chooses to believe or do what it wants. It works as a representative of a user in order to achieve the goals, which will be bounded by given plans which define the ways in which an agent can act to achieve goals and subgoals. Therefore, an autonomous agent makes independent decisions about how to achieve its delegated goals. All decisions are under its own control.

- *Proactiveness*: an agent is able to exhibit goal-directed behavior proactively; in contrast an object is passive until something invokes a method on it.

- *Reactiveness*: an agent is enabled to be reactive to change in the environment. When its plans have gone wrong, the agent is able to choose an alternative course of action. Some responses are reflexes, while some need more deliberation. To achieve the balancing between goal-directed and reactive behavior is a design compromise.

- *Social Ability*: an agent has an ability to cooperate and coordinate its activities with other agents in order to achieve its/their goals. That is, agents are able to communicate their beliefs, goals, and plans between one another.

2.2 The BDI Architecture

Belief-Desire-Intention (BDI) is a type of intelligent agent architecture that has been proposed by Rao and Georgeff (1991). Apart from *belief-desire-intention*, there are *logic-based architectures*, *reactive architectures* and *layered architectures*, as presented in (Weiss, 2000). However, this research focuses on agents that are developed based on the *BDI* model. This is because the *BDI* model has been developed for describing human behaviour and as an abstraction tool useful for formulating complex systems (Bordini et al., 2007). It main elements are:

- *Beliefs*: the beliefs of an agent represent the agent's knowledge. The content of the knowledge can be anything; for instance knowledge about the agent's environment or about its history.

- *Desires*: the desires of an agent are a set of long-time goals. A goal is typically a description of a desired state of the environment. The desires provide the agent with the motivations to act.

- *Intentions*: the intentions are a subset of the desires, or the intentions may be considered as a set of plans for achieving the goals in the desires. When initialized,

the intentions contain some of the goals from the desires. These goals should not contradict each other. The intentions are viewed as something the agent has dedicated itself to trying to fulfill. This gives stability to the system since it means that the agent will not try to attain contradictory goals.

A computational model of *BDI* (Bordini et al., 2007) is shown in Figure 2.1

```
1. B ←B0            /* B0 are initial beliefs */
2. I ← I0           /* I0 are initial intentions*/
3. while true do
4.      get next percept ρ via sensors;
5.      B ← brf(B, ρ);
6.      D ← options(B,I);
7.      I ← filter(B,D,I);
8.      Π← plan(B,I,Ac);          /* Ac is the set of actions */
9.      while not (empty(Π) or succeeded(I,B) or impossible(I,B)) do
10.          α ← first element of Π;
11.          execute(α);
12.          Π ← tail of Π;
13.          observe environment to get next percept ρ;
14.          B ← brf(B, ρ);
15.          if reconsider(I,B) then
16.              D ← optons(B,I);
17.              I ← filter(B,D,I);
18.           end-if
19.           if not sound(Π,I,B) then
20.              Π ← plan(B,I,Ac);
21.           end-if
22.     end-while
23. end-while
```

Figure 2.1: A computational model of *BDI* from (Bordini et al., 2007)

The overall control loop for a *BDI* agent is shown in Figure 2.1 and works as follows:

B_0 and I_0 are initial beliefs and intentions and have been assigned to variables B and I respectively in Lines (1) and (2).

The *while* loop between Lines (3) and (23) iterates through the following actions:

Line (4) the agent observes its environment via sensors, and gets the next percept (ρ).

Line (5) the agent updates its beliefs via the belief revision function by passing its current belief and the new percept (ρ), which then returns new beliefs to the agent.

Line (6) the agent determines its desires via the option function by passing its beliefs and intensions.

Line (7) the agent updates its intentions via the filter function by passing its beliefs, designs and intentions.

Line (8) the agent generates it's a set of plans via the plan function by passing beliefs, intentions and the set of actions.

The inner *while* loop between Lines (9) and (22) repeats the following actions:

The agent reconsiders its intensions by evaluating from the possibility of success and then executes the set of plans to achieve the goals. The agent picks off each action from its plans and executes it until the plans are empty

Line (13) the agent re-observes its environment to get the next percept (ρ).

Line (14) the agent updates its beliefs via its belief revision function by passing current beliefs and the new percept (ρ), and then returns new beliefs to agent.

The Lines (15) to (18) the agent makes the decision that it would change its new intentions or not by deciding via its reconsider function.

Line (19) the agent checks the current set of plans whether or not it is compatible with its intentions and its beliefs.

Line (20) if the set of plans are not compatible, then the agent changes a new set of plans.

For this research we use *Jason* as a tool for implementing the proposed models. *Jason* (Bordini et al., 2007) is an agent programming language which provides a Java-based platform used for developing multiagent systems. It is an extension and an interpreter of *AgentSpeak* which is based on the *BDI* architecture and has been implemented by Rao (1996). The user uses it for programming the behavior of individual agents and to

customize most aspects of an agent or multiagent system. *Jason* provides annotation of beliefs, annotation of plans, method of producing plan libraries and speech-act based inter-agent communication. Moreover, it provides *Java-based* definitions for environment giving agents within the system percept and interact with the environment. It also provides the facility for running a multiagent system distributed over a network and the extensibility for user to define "internal action". Besides *Jason*, there are other languages which are based on BDI architecture such as *PRS* (Georgeff and Lansky, 1987) , *JAM* (Huber, 2001), *JACK* (Busetta et al., 1999), *dMARS*(Kinny, 1993), *2APL* (Dastani, 2008), *JADE*(Bellifemine et al., 2007). The *Jason* has been chosen to be an implementing tool as we have experienced in *Java* programming language and having technical consultants in *Jason*.

2.3 MultiAgent Interaction

In (Jennings, 2000) the author illustrates how the agents in a multiagent system interact with each other through communication and how each one is able to act in an environment which has its own spheres of influence. These spheres of influence may coincide; in such an event the agent needs some kinds of relationships to prioritize them.
Interaction between the agents causes different preferences in a multiagent system. Each agent owns different values of utility which it is the result of the state of the preferences.

Consider a domain which has only two agents i and j respectively:

Each agent has two possible actions to perform that are C (*cooperate*) and N (*non-cooperate*)

The set of outcomes or states is $\Omega = \{\omega_1, \omega_2, \dots\}$

The environment is determined by function $\tau : AcXAc \rightarrow \Omega$

Therefore, all of the possible outcome of this function are

$$\tau(N,N)=\omega_1,\ \tau(N,C)=\omega_2,\ \tau(C,N)=\omega_3,\ \tau(C,C)=\omega_4$$

and the utilities of each agent are

$$u_i(\omega_1) = R_1,\ u_i(\omega_2) = R_2,\ u_i(\omega_3) = R_3,\ u_i(\omega_4) = R_4$$

$$u_j(\omega_1) = R_5,\ u_j(\omega_2) = R_6,\ u_j(\omega_3) = R_7,\ u_j(\omega_4) = R_8$$

under the same environment each agent gets a different utility value and as it is assumed to be self-interested, each would choose the action which would give it the highest value, or both agents would choose the action that give both of them as much as possible (reaching equilibrium).
Nash Equilibrium (Wooldridge, 2002) has been defined as follows:

> Let s1 and s2 be two strategies, if these two conditions are satisfied:
>
> (1) under the assumption that agent i plays s1, agent j can do no better than play s2; and
>
> (2) under the assumption that agent j plays s2, agent i can do no better than play s1
>
> Then, both agents reach "*Nash Equilibrium*".

The study of interactions among self-interested agents is known as *Game Theory* (Binmore, 1991). It is a mathematical-based theory which is applied in multiagent systems. It helps with telling about what the properties of an appropriate/optimal solution are, rather than telling how a solution has been computed; as the problem of computing a solution in the multiagent research field is computationally very hard, e.g. *NP-complete* or worse. It seems more understandable when modeling human/artificial agent societies in game theory.
Games are classified into four classes by Gibbons (1958): *static games of complete information*, *dynamic games of complete information*, *static games of incomplete information*, and *dynamic games of incomplete informatio*n. In a game of incomplete information, a player does not know another player's payoff. The example of this situation is an auction where a bidder does not know how much another bidder will offer as a bid for the goods being sold. These four classes of games correspond to four notions of equilibrium in games: *Nash equilibrium*, *sub-game-perfect Nash equilibrium*, *Bayesian Nash equilibrium*, and *perfect Bayesian equilibrium*. More details are provided in (Fudenberg and Tirole, 1991). According to Nash equilibrium, it states that in equilibrium every agent will select a maximum utility strategy, given the strategy of every other agent.

> Let $s = (s_1, s_2, ..., s_i)$ be the joint strategies of all agents, or strategy profile,
>
> and $s_{-i} = (s_1, ..., s_{i-1}, s_{i+1}, s_i)$ means the strategy of every agent except agent i

and $\Theta_{-i} = (\Theta_1, ..., \Theta_{i-1}, \Theta_{i+1,}\Theta_i)$denotes the preference of every agent except i

Definition A strategy profile $s = (s_1, s_2, ..., s_i)$ is in ***Nash equilibrium*** if every agent maximizes its expected utility, for every i,

$$u_i\left(s_i\left(\Theta_i\right), s_{-i}\left(\Theta_{-i}\right), \Theta_i\right) \geq u_i\left(s_i^{'}\left(\Theta_i\right), s_{-i}\left(\Theta_{-i}\right), \Theta_i\right) \text{ for all } s_i^{'} \neq s_i$$

Strategy $s_i\left(\Theta_i\right)$ under preference Θ_i gives agent i the greatest utility value when comparing with the other strategies $s_i^{'}\left(\Theta_i\right)$ under the same circumstance.
This research applied the concept of interaction among agents by having each one choosing the best response strategy and maintaining the best benefit of individual agents at any given moment until the system eventually terminated. More information of this is in section 4.3—allocation protocols.
Game theory has been applied as a model in negotiation processes for decades. Each player plays according to the rules of the game. The players use strategies and the payoffs to encounter among each other in order to choose his/her best option and gain the maximize results. There are seven elements that are composed to form a game: *players*, *actions*, *information*, *strategies*, *payoffs*, *outcomes* and *equilibrium*. A player chooses a strategy to respond according to the rules of a game. The chosen strategy results in a payoff. Game theory is addressed on the benefit of the individual player. It can be used to design a negotiation mechanism, and particularly for the definition of protocols that restricts the number of strategies used by the parties (Sierra et al., 1997) .

2.4 MultiAgent Resource Allocation (MARA)

Chevaleyre et al. have defined the meaning of *Multiagent Resource Allocation (MARA)* as the process which distributes a number of items among a number of agents. The items that are being distributed are resources, while agents are the entities receiving them (Chevaleyre et al., 2006).
There are two possible types of resources, these are:

- *Divisible resource*: one that may be divided into fractions and be distributed to different agents.
- *Indivisible resource*: one that may or may not be possible for different agents to share the same resource.

A particular distribution of resources among agents is called an *allocation*, and the set of resources assigned to a particular agent is called the *bundle* allocated to that agent. An agent may/may not have preferences about the bundles it received. When an agent has preferences about the bundle received by an other agent, this is termed *externalities*. A good mechanism design should provide an incentive for agents to reveal their preferences truthfully.
There are two possible ways to do the allocation procedure, which these are:

- *Centralized*: so that an agent decides on the final allocation of resources among all agents, such as auction protocols.
- *Distributed*: when the allocations are determined as a result of a sequence of local negotiation steps. The computational burden of finding an allocation solution is shared among several agents such as a contract net protocol.

Both procedures have the same objective to find either a *feasible* or *optimal* allocation. A *feasible* allocation means that the allocation solution is satisfied, whereas an *optimal* allocation means that allocation solution is the best available among several feasible solutions (given a maximum utility value).
Research in the multiagent resource allocation field encompasses not only the theoretical analysis of computational complexity but also the design of efficient algorithms. It is related to many disciplines such as *Computer Science*, *Artificial Intelligence*, *Decision Theory*, *Microeconomics*, and *Social Choice Theory*; results of many subfields of researches under *MARA*, for instance, *Preferences, Social Welfare*, *Complexity, Negotiation*, *Algorithm Design, Mechanism Design*, *Implementation*, *Simulation and Experimentation*, *Interplay of Theory and Applications*.

2.4.1 Preference Representation

In *MARA*, each agent needs to express its relative or absolute preferences when faced with a choice between options. These options have different potential in resource allocations. A *preference structure* represents an agent's preferences over a set of alternatives X, and there are several choices that can be made regarding the definition of a mathematical model for preference structures. These are (Chevaleyre et al., 2006):

- A *cardinal preference structure* consists of an evaluation(*utility*) function $u : X \rightarrow Val$, where *Val* is either a set of numerical values (quantitative) or a scale of qualitative values.

- An *ordinal preference structure* consists of a binary relation between alternatives, denoted by $\preceq$, which is reflexive and transitive. Strict preference $x \prec y$ if and only if $x \leq y$ but not $y \leq x$; and $x \simeq y$ if and only if both $x \leq y$ and $y \leq x$.

- A *binary preference structure* is a function which maps a partition of X into a set of good and a set of bad states. A binary preference structure can be seen as both an ordinal preference structure and cardinal preference structure.

- A *fuzzy preference structure* is a fuzzy relation over X i.e. a function $\mu : X \times X \rightarrow [0, 1]$. $\mu(x, y)$ is the degree to which x is preferred over y. Fuzzy preferences are more general than both ordinal and cardinal preferences.

Quantitative preference can be used not only for comparing the satisfaction of a given agent for different alternatives, but also for expressing preference intensity, which allows for interpersonal comparison. As any cardinal preference induces an ordinal preference, a utility function can be defined as the complete weak order$\leq_u$ given by $x \leq_u y$ if and only if $u(x) \leq_u u(y)$. In our proposed model, we also chose to apply a cardinal quantitative preference for expressing the different choices of resources which each agent will choose the best-fit and the most preferred resource to occupy.

2.4.2 Social Welfare

The main role of *MARA* is to allocate the resources among agents. The aggregation of individual preferences can be modeled as a notation of *social welfare*. For each agent, if it calculates its own preferences by using the utility function, that is the mapping from bundles of resources to numerical values, then the total value of individual utilities, which is called *utilitarian social welfare*, can be used to measure the quality of the allocations in the system as a whole. A *social welfare ordering* is a mapping from the preferences of the agents in a society to the preferences of the society as a whole.
Let $A = \{1, ..., n\}$be a set of agents. Each of these agents i is provided with either a utility function u_i or a preference relation $\leq_i$. An allocation P is a mapping from agents to bundles of resources; P(i) is the bundle held by agent i in allocation P. $P \leq_i Q$ means agent i is more like allocation Q than allocation P, which it is an abbreviation of and $u_i(P)$ is a short form of $u_i(P(i))$.

2.4.2.1 Collective Utility Function

If individual agents use utility functions to represent their preferences, then every allocation P gives a utility vector $< u_1(P),...,u_n(P) >$. A *collective utility function (CUF)* is a mapping from such vectors to numerical values; that means it is a function from allocation P to numerical values. Every *CUF* induces a social welfare ordering: The allocation Q is socially preferred over allocation P if and only if $sw(P) \leq sw(Q)$.

- *Utilitarian Social Welfare* is defined as the sum of individual utilities:

$$sw_u(P) = \sum_{i \in A} u_i(P)$$

- *Egalitarian Social Welfare* is the utility value of the agent that is currently worst off:

$$sw_e(P) = min\,\{u_i(P) | i \in A\}$$

- *Nash Product* is the product of individual utilities:

$$sw_N(P) = \prod_{i \in A} u_i(P)$$

- *Elitist Social Welfare* is the utility value of the agent that is currently best off:

$$sw_{el}(P) = max\,\{u_i(P) | i \in A\}$$

- *Rank Dictators* is defined the k^{th} smallest utility assigned to allocation P by any of the agents in A. The k-rank dictator *CUF* sw_k is:

$$sw_k(P) = (V_p^{\uparrow})_k$$

In our research, we have applied utilitarian social welfare by employing a *Round-Robin (RR)* algorithm in section 4.3.1.2 to reorder the sequence of *Year-Program Agents (YPAs)* for each round of organizing the timetable.

2.4.3 Allocation Procedures

The aim of the allocation procedure is to find a suitable allocation of resources which might be either *centralized* or *distributed*. For *MARA*, allocation procedures involve the following three issues.

- *Protocols*: there are two subjects that need to be part of protocol design, there are the ontology and communication protocols.
- *Strategies*: the protocols for each agent should be designed in a way that provides incentives for the agent to negotiate and to reveal the genuinely desirable profile (mechanism design).
- *Algorithms*: the algorithms for solving the computational problems when the agent is faced by problems and employed in negotiation.

2.4.3.1 Auction Protocols

An *auction* is a centralized mechanism which is designed for allocating goods, tasks and resources among several agents. Under the auction protocol, one agent is known as the auctioneer, while a collection of agents is known as the *bidders*. The goal of the auction is for the *auctioneer* to allocate the goods to one of the bidders. The auctioneer desires to maximize the price of the goods that are allocated, whereas bidders desire to minimize the price. The auctioneer tries to achieve his/her goal through an appropriate auction mechanism, at the same time as the bidders also try to reach their own goal, so each agent uses its own strategy that will conform to the rules of encounter. Eventually this mechanism will lead the system to reach an optimal result. There are many types of auction including:

- *English Auction*: an open outcry auction, the bid in each round must be greater than the previous bid, and the winning participant pays the highest announced price.
- *Dutch Auction*: an open outcry auction, the bid in each round must be lower than the previous bid. And the winning participant pays the lowest announced price.
- *Sealed First-Price Auction*: this is a concealed auction, the submitted bids are compared and the person with the highest bid wins the award, and pays the amount of his bid to the seller.

- *Sealed Second-Price Auction (Vickrey auctions)*: this is also a concealed auction, the submitted bids are compared and the person with the highest bid wins the award, and pays the second highest bid price.

In our research, we have not applied any auction mechanism for allocationg resources as other research has done. This is because an agent has the potential to choose the resource which meets its needs. More details about allocation of resources are given in section 4.3 and section 5.3.

2.4.3.2 Negotiation Protocols

Negotiation protocols are mechanisms which have been designed to work in a distributed setting. Negotiation is a technique used to reach common agreement, as discussed in (Rosenschein and Zlotkin, 1994). Generally, there are four components in any negotiation set (Wooldridge, 2002):

- A *negotiation set* denotes a set which contains the space of possible proposals
- A *protocol* means a set of rules which govern the interaction.
- A *collection of strategies* for every agent there is a set of strategies which is used for determining which proposals the agent will make.
- A *rule* identifies the information when a deal is unable to be reached; or what this agreement deal is.

The process of negotiation usually proceeds via a series of rounds with each agent making proposals according to its own strategy. The proposals must be chosen from the negotiation set and must be legal as defined by the protocol. If agreement is reached, then the negotiation process will be terminated.
The degree of complexity in negotiation can be analyzed in terms of these four factors:

- *The number of attributes: a single-issue or multiple-issue negotiation*
- *The number of possible deals*
- *The number of unclear attributes under negotiation*
- *The number of involved agents in negotiation*

There are three forms of interaction among the agents:

- *One-to-one (bilateral) negotiation*: when there are two parties negotiating with each other.
- *Many-to-one (auction) negotiation*: when there is one auctioneer negotiating with many bidders, such as the bidding processing in the classical contract net protocol.
- *Many-to-many (distributed and multilateral) negotiation*: when there is a set of parties negotiating with another set of parties in order to reach agreements.

For any type of interaction among agents, there are usually two possible interdependent negotiation situations (Lewicki et al., 2010):

- A *distributive (win-lose/zero-sum)* situation arises in the situation of having a large number of demands while supplies are limited.
- An *integrative (win-win/non-zero-sum)* situation arises when the negotiators search for situations which give benefits for both parties. In our research, we also have employed the win-win interdependence in our negotiation protocol as well. More details of negotiation protocol are in section 4.3.2 and section 5.4.
  - The first winner is the agent who frees his current occupied resource and switches to acquire the new resource which gives a utility value not less than the one he had before.
  - The second winner is the agent that takes the freed resource.

2.4.3.3 Contract-Net Protocol (CNP)

The *Contract Net Protocol (CNP)* (Smith, 1980) is a high level communication protocol for cooperative task sharing in a Distributed Problem Solver. It imitates the way that a company puts contracts out for tender. As presented in Figure 2.2, the process of Contract Net involves, by the following steps:

- ***Task announcement processing***: the manager announces a task in three possible ways:
  - *A general broadcast*: used in the case which the manager lacks knowledge of the other nodes' capabilities, he/she therefore broadcasts to every node.

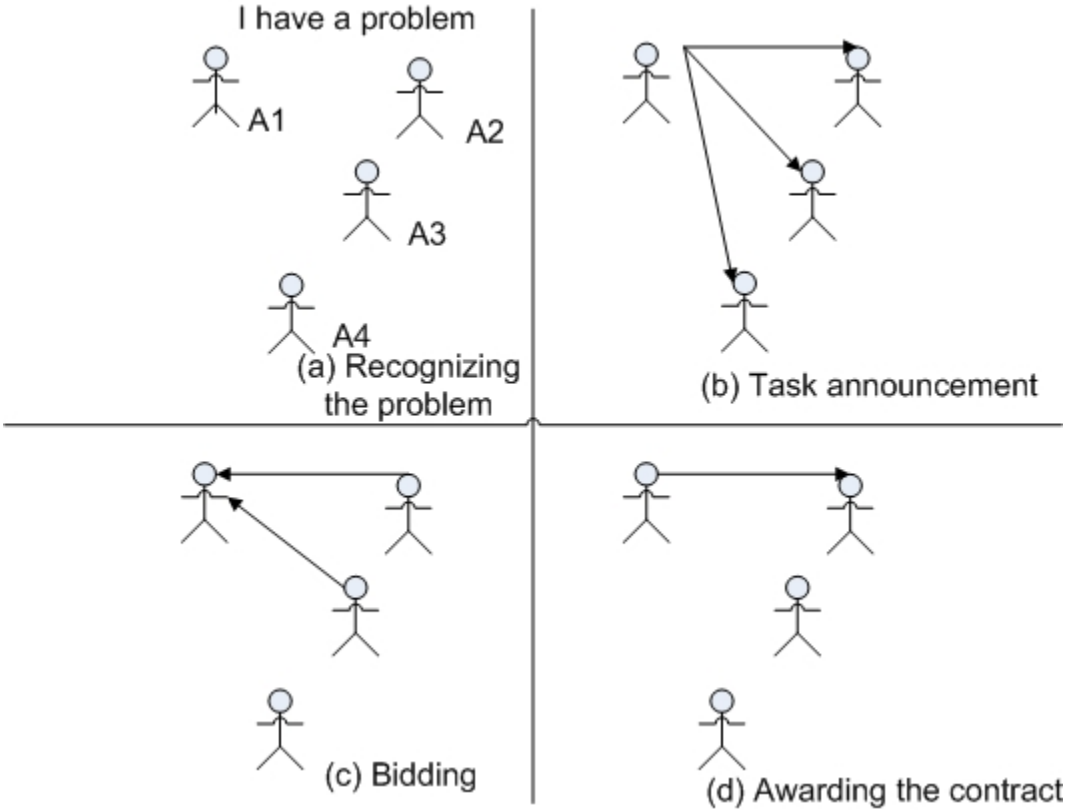

Figure 2.2: The Contract Net protocol (CNP) from(Smith, 1980)

  - *A limited broadcast*: used in the case which the manager possesses some knowledge about which nodes in the net are likely to have appropriate capabilities and broadcasts only to the specific set of these nodes.
  - *A point-to-point announcement*: used in the case which the manager knows which node is appropriate for the specific task. However, the node also has the option of refusing the task.

- ***Bid processing***: the nodes in the net listen to the task announcements and evaluate themselves. If a node finds that the proposed task is appropriate for its capabilities, it then submits a bid to the manager. When the deadline is reached, the manager awards the task to a single bidder, which is called the contractor.

- ***Award processing***: The successful bidder must complete the task, while the agents who were unsuccessful in the bid delete the task information.

- ***Request and inform processing***: The manager receives the inform message immediately after issuing the request, if the information is available. Otherwise, the manager will be informed that the information is unknown. At the conclusion of a task, the contractor will send the detail of the achieved task to the manager.

In our research, the idea of contract net protocol has been extended and applied in our negotiation part of our model in order to switch the resource to the one who is most able to use it. Additionally, we found many researchers have extended the contract net protocol and employed in their work as well, including:

- The *TRACONET* system, developed by Sandgolm (1993), applies contract net protocol to allow negotiation over the exchange of bundles of resources.
- Sousa et al. (2003) have adapted the contract net protocol for bidders propagating constraints between them in order to guarantee the coherence of different operations which are related to the same task.
- Golfarelli et al. (1997) have adapted the original contract net protocol for agents to bid for resources which they are interested in for the propose of exchanging.
- Aknine et al.(2004) have designed a concurrent contract net which allows many managers to negotiate simultaneously with many contractors. The extension consists of a pre-bidding and pre-assignment phase which is added before the final bidding and assignment phase.

For any type of negotiations, the rational agent will be defined as an agent that will agree to deal, the result which gives a positive payoff for itself. That means the agents will agree only on mutually beneficial deals. Many mutually beneficial deals have been backed up by the concrete strategies. However, aggregate negotiation strategies may prevent agents from identifying any mutually beneficial deal, but the agents will certainly agree on some deal which meets rationality criteria (Sandholm, 1998). When such a deal exists, it is called *convergence properties of a negotiation framework*. In (Sandholm, 1998), Sandholm states that "under the context of negotiation over finitely many indivisible resources, any sequence of deals that is mutually beneficial will consequently result in an allocation with maximal utilitarian social welfare. If there is no infinite sequence of mutually beneficial deals and the agents keep on making such deals, then the system will converge to an allocation that maximizes the sum of individual utilities". Moreover, in (Endriss et al.,

2003), their results state that "any sequence of mutually beneficial deals without side payments will converge to a Pareto optimal allocation".

2.4.4 Mechanism Design

A mechanism or protocol is the set of rules that given an encounter between agents in a multiagent system and that are used in order to reach an agreement (Rosenschein and Zlotkin, 1994). By following the rules while negotiating, an individual agent can eventually reach his/her maximize value of the sum of the utility functions. Therefore, mechanism design is the design principle used for governing multiagent interactions, so that the system is free from deadlocks, live-locks and so on, similar to the design of a conventional communication protocol, but also with the following properties added (Sandholm, 1999):

- *Guarantee success*: a mechanism can guarantee success if it ensures that agreement is definitely to be reached.
- *Maximizing social welfare*: a mechanism could maximize social welfare if it ensures that any outcome maximizes the sum of the utilities of negotiation participants.
- *Pareto efficiency*: a negotiation leads to the *Pareto* efficiency if there is no other outcome that will make at least one agent better off without making at least one other agent worse off.
- *Individual rationality*: the designed protocol is in the best interests of negotiation participants. This property is very important, since if there is no such property; the agent has no incentive to participate in negotiation.
- *Stability*: the designed protocol is stable if it provides all agents with an incentive to behave in a particular way—reaching *Nash equilibrium.*
- *Simplicity*: the designed protocol should be simple enough to make the appropriate strategy for negotiation participant obvious. Each agent can then determine an optimal strategy.
- *Distribution*: the protocol should not only have no single point of failure, but should also minimize communication between agents.

In our research, the rules for encounters between agents, that have been designed for working in either the initial allocation phase or the negotiation phase have met some of above properties. These are *Guarantee success, Maximizing social welfare, Individual rationality, Stability* and *Simplicity.*

2.5 Summary

In this chapter, we provide some background about agent-based system of the form employed in this research. It provides a description of intelligent agent and multiagent systems. *BDI* architecture is addressed, as it is a model that has been developed for describing human behaviour and for formulating complex systems which provides the basis for our implementing tool *Jason.* The concepts of multiagent interaction and *Game Theory* have been reviewed. In any given circumstance, each self-interested agent would choose the action which would give it the highest returned value, or both selfish agents would reach "*Nash Equilibrium*". At the end of this chapter, an overview of multiagent resource allocation concept is summarized by covering the folowing topics: *preference representation*, *social welfare*, *allocation procedures* and *mechanism design.*

Chapter 3

The University Course Timetabling Problem

This chapter provides background about university course timetabling issues. It starts by describing the types of timetables used in the education domain. It then presents a list of university course timetabling issues which need to be addressed. Thirdly, it describes the reasons which why agent technology is suitable for solving timetabling problems. For the fourth section, some examples of university course timetabling research, which has used agents, are reviewed. Then, in the fifth section, several classical timetabling techniques are described; and the last section of this chapter then summarizes how other university course timetabling models compare with the proposed model.

3.1 Educational Timetabling

Educational timetabling involves all activities which are relevant to creating a timetable in educational institutes. It belongs to the class of *NP-complete* problems. That means it is unlikely that there is a method which can solve it in a polynomial amount of time (Burke et al., 1997). The educational timetabling context involves scheduling a set of courses to specific blocks of time and to rooms, subject to certain conditions. The resources for this context are students, staff, rooms, courses, time and equipment. A survey of automated timetabling by Schaerf (1999) categorized educational timetabling into three different types of scheduling where each differs in terms of the type of institute and the type of constraints.

- *School Timetabling*: This provide the scheduling for all the classes of a school, normally organized in weekly periods, and aiming to avoid such situations as a teacher being assigned to any two classes at the same time and vice versa.

- *Course Timetabling*: This is also a weekly scheduling structure used for all the lectures of a set of university courses, aiming to avoid any overlap of lectures for courses which are taken by the students on a particular programme.

- *Examination Timetabling*: This is the scheduling of the exams for a set of university courses, needing to avoid any overlap of exams for courses which are taken together by the students, and aiming to spread the exams across the assessment period as much as possible.

Each type of timetabling involves consideration of different aspects. Examination timetabling needs to ensure that any two consecutive exams for the same student should be separated by sufficient time, school timetabling places a high priority on teacher availability and reducing idle time for students, whereas course timetabling aims to find the timetable with the fewest conflicts or that is conflict free, enabling the students from different curricula to take as many combinations of courses as they wish. The obvious distinction between course scheduling and examination scheduling is that one room must be occupied for one lecture at any time when that lecture is scheduled, while for exams one room is often shared by students from many courses, or students in one course are spread across rooms. Course timetabling must address the wider context of the use and availability of space much more than examination timetabling.
This research focuses on solving university course timetabling problems. As with the other timetabling problem domains, a solution requires a number of constraints to be satisfied. The constraints which are involved in this problem can be divided into two categories (Burke et al., 1997).

- *Essential (Hard) constraints* consist of set of constraints which must be met in full. A timetable which meets essential constraints is a feasible solution; and we term it —a *valid* timetable.

- *Desirable (Soft) constraints* consist of set of constraints which do not need to be fully satisfied. A valid timetable which meets some desirable constraints is termed —a *good* timetable. It is the satisfaction of soft constraints that can make one valid timetable better than another (Woods and Trenaman, 1999).

Table 3.1 presents a set of *essential* and *desirable* constraints which have been considered in university course timetabling problems (Burke and Petrovic, 2002,Causmaecker et al., 2006)

Essential constraints	**Desirable constraints**
Lecturers and students can only be in one location at the same time.	Lecturers express preferences for certain time slots.
There should be sufficient resources for any time slots.	One course may need to be scheduled before/after the other.
A room can hold at most one lecture at a time.	Some student groups must have one day off per week.
Class rooms have a limited capacity.	Lectures should be scheduled in 2 or 3 consecutive time slots from a student point of view. Scheduling a single lecture on one of the teaching days is to be avoided.
The number of class rooms is limited.	Scheduling lectures in the last time slot of the day is to be avoided.
Feasible time slots are the periods that occur on weekdays.	The number of lecturing hours per days should be spread evenly over the week.
The timetable period is for one term.	Gaps between lectures are to be avoided.
	Lectures may prefer to have all their lectures in a number of days.
	Lectures may prefer to teach in a particular room.

Table 3.1: *Essential* and *desirable* constraints from(Burke and Petrovic, 2002Causmaecker et al., 2006)

3.2 The Gap in University Timetabling Problems

Barry McCollum has suggested that a gap between the theoretical and practical aspects of university timetabling still remains (McCollum, 2007). He argues that most research is directed at evaluating particular techniques and approaches, and that automated timetabling ignores the human factors and chooses to deal with only the data sets.

Problems	**Current Solutions**
Rising student numbers	Significant human interaction
The series of events that exist in one module	Fuzzy algorithms, split by objectives, meta-heuristics
The goal of optimization	High level heuristics + Optimization techniques
The timetabling of associated events together (pathways within a particular school)	Left for further research

Table 3.2: Timetabling problems and solutions from (McCollum, 2007)

Furthermore, Carter and Laporte have observed that very few of the research methods proposed for timetabling have been implemented and used in an institution, including the ones which have been implemented, namely the search technique on benchmark datasets (Carter and Laporte, 1998). This because there has been insufficient investigation of real world issues and understanding of the methodologies used by human schedulers. Moreover, each institution must satisfy a range of different constraints when generating its timetables, which means that finding a generally applicable solution to the complex problem is extremely difficult. A range of proposed research approaches are based on real world problems but are still based upon simplifying assumptions when modeling a problem. Therefore, they chiefly form an initial research test bed through presenting powerful search techniques. Both McCollum and Carter assert that the process for producing a workable timetable still remains with a combination of lecturing and administrative staff rather than through the use of an automated component. Much more work is needed to investigate the formulation and modeling of the problem. (Carter, 2000,McCollum, 2007). Automated timetabling repeatedly struggles, not only to cope with rising student numbers, but also with the series of events that needed to be included in one module e.g. lectures, seminar, tutorials, practical classes and laboratory classes. Moreover, the goal of optimization is sacrificed over and over for the purpose of getting a solution. Table 3.2 presents a list of timetabling problems and the current solutions being used for these.

McCollum (2007) has identified a number of significant challenges facing university course timetabling researchers and needing further investigation i.e.

1. how to produce timetables within an institution that seem to be fair and equitable

to all interested parties.

2. the balancing between the size of each subclass of any course while still offering students maximum choice.

3. how to construct and optimize the solutions by keeping a balance of all the stakeholders' satisfaction e.g. student choice, staff preference and room usage.

3.3 Employing Agents in Timetabling Problems

In recent years, many papers have been published that argue that multiagent technology is suitable for solving timetabling problems, as in (Causmaecker et al., 2003). The authors observe that within each timetabling problem, there always appeared concurrently the differentiation between local level that relates only to its own needs, and global level that relates to the needs of the whole system. Both levels need decision support capability. If using the multiagent paradigm, agents could address the above problem by negotiating and collaborating with each other. Another supporting reason has been noted by Carter (2000), who observes that a distributed software architecture can adequately describe timetabling problems. Decision support in distributed systems with autonomous, asynchronous components can be modeled by a multiagent system. Where agents are used to provide a model for distributed timetabling, the distribution arises from the presence of separated, autonomous components which do not communicate in every detail, but do exchange more coarse grained information. The decisions made by an agent must be based on incomplete information, depending on individual capacity or its own knowledge. The agents will typically use expressions of interest and degrees of agreement obtained from individual agents. For real world operators, better decisions are obtained through a negotiation process in which all partners actively search for better solutions and find the ways to alleviate other partner's problems. Autonomous agents negotiate on behalf of the operators that they represent; hence agent technology has the potential to resolve timetabling problems, as it provides mechanisms to develop such a decision-making system while running in a dynamic and distributed manner.

3.4 Literature Review: UCTP Solved by Use of Agents

Researchers still continue to solve timetabling problems from different domains by different techniques; including using multiagent technology to propose various models to sort out university timetabling problems.
The following section summaries research applying agent technology to address university course timetabling problems over the last ten years.

(1) **Kaplansky and Meisels** (2004) have developed a model that assigns agents to the roles of *Scheduling Agents* and a *Room Agent*.

- *Scheduling Agent*: takes responsibility for generating the timetables for one particular department by employing *Constraints Satisfaction Problem (CSP)* techniques in the agent. Each department not only needs to schedule a set of lectures for each course within a given number of time periods and rooms, but also needs to solve different problems that correspond with departmental needs; such as ownership of different resources, different teaching requirements, different preferences and different strategies for utilizing the teaching spaces.
- *Room Agent*: takes a responsibility for cooperating with the scheduling agents to meet the inter-departmental constraints and is responsible for assigning rooms to each scheduling agent.

In this research, negotiation protocol has been defined for *Scheduling Agents* to negotiate among each other to reach the shared timeslot if all of them share the same course. That means the course will be taken by more than one department and will be scheduled on the same timeslot on timetables of all the shared departments. The inter-agent negotiation process is divided into nine steps and roughly grouped into three stages as follows:

- The first stage (steps 1–3) focuses on each department's solution
  - Step 1: *Self assessment*—each Scheduling Agent assesses the solutions generated by expressing the *Average Computation Cost (ACC)* for each solution found; the ACC value is the average number of constraint checks per solution which the agent has found.

  - Step 2: *Direction of the inter agent constraints*—the result of step 1 can be used to determine agent's constraint; the agent which has more complex of the same constraint direction is called *Course Manage Agent,* while the other agents on the same constraint direction are called *Course User Agents.*
  - Step 3: *Calculate best local solution*—the agent computes its best local solution with the shared courses set at fixed time slots; the best solution cost is called *Agent Maximal Cost (AMaxC)*

- The second stage (steps 4–8) eliminates the conflicts between the *Scheduling Agents*
  - Step 4: *Initial suggestion*— every *Course Manager* sends the time slots of its Shared Courses to the agents that will use these courses; and then the agents respond by calculating their own Agent Cost Range ($ARangeC = AMaxC - AMinC$) and send this back to the *Course Manager.*
  - Step 5: *Normalizing Agents costs*— the *Course Manager* generates a common scale by normalizing the complexity of all Course User Agents from the values of *ARangeC.*
  - Step 6: *Requests for Change*— each *Course User Agent* seeks to improve its local timetable by changing the time slot of one of its shared courses and solving its local problem. It sends a request for change and a list of suggested timeslots to the *Course Manager Agent*
  - Step 7: *Approve Change*— when a *Course Manager Agent* receives a request and a list of suggested timeslots. It searches for the best solution by looking from the accompanied list. If the sum of all Change Costs is lower than the Expected Gain; then the change request is approved. Otherwise, all agents proceed to the bidding step.
  - Step 8: *Bidding*— the bidding takes place between the initial *Course User Agent* and the *Course Manager Agent.* If the bid succeeds, then the request will be approved.

- In the final stage (step 9) *Scheduling Agents* interact with *Room Agent*

asking for rooms

- Step 9: *Negotiation with Room Agent*— when a university timetable is stable, each agent sends a request for each course to the *Room Agent*. If at a specific timeslot the demand for rooms exceeds supply, then the *Room Agent* sends a '*refuse*' message to all requesting agents. After each agent gets the '*refuse*' message, it tries to change the timeslot and sends to *Room Agent* again. If the *Room Agent* gets enough response messages agreeing to change, then the problem is solved. Otherwise, the *Room Agent* begins an auction for rooms for this timeslot.

Drawbacks can be found in this negotiation model when considering conflicts for either timeslots or rooms. According to the protocol, exhaustion in the *Scheduling Agent* might cause infinite loops to occur. Moreover, although the authors argue that this research is ongoing (from 2004) they have not published any actual course timetabling results so far.

(2) **Oprea** (2007) defined a multiagent system that could be used to schedule university course timetables. The architecture of the system is defined to have the same pattern of university's organization levels. The proposed model is composed of four types of agents working together, which these are:

- *Main Scheduler Agent (MSA)*: a timetabling scheduler at university level.
- *Faculty Scheduler Agent (FSA)*: a timetabling scheduler at faculty level.
- *Expert Assistant Agent (EAA)*: a timetabling scheduler at department level.
- *Personal Agent (PA)*: a scheduler that works on behalf of a professor to manage his/her teaching activities.

The professors submit their preferences, which are a list of options for teaching courses, to the department timetable specialists who organize the department timetables. After that, the department timetables will be proposed to the faculty scheduler.

The university course timetable scheduling is divided into two procedures:

- Faculty course timetable scheduling allocates days and times for courses.

- University course timetable scheduling allocates rooms for courses.

The system will terminate successfully, when all courses have been assigned date-time slots and rooms. However, if any conflict occurs, a negotiation activity will be started. The following scenarios show how the system sorts out some forms of conflict:

- *Date-time slots conflicts*: between *Expert Assistant Agent* and *Personal Agents*. These occur when several professors prefer the same date-time slot. The *Expert Assistant Agent* sends a message to all professors who are involved in the conflict problem, and waits for a solution by negotiation among them. If the negotiation can reach an agreement, then the *Expert Assistant Agent* will reschedule the timetable according to the agreement reached. Otherwise, if the *Expert Assistant Agent* receives no solution, it starts a persuasion process of negotiation to suggest a possible solution.
- *No room is available*: the *Main Schedule Agent* sends a signal to start the negotiation among *Faculty Scheduler Agents*. Each *Faculty Scheduler Agent* solve the conflict by passing the message to the corresponding *Expert Assistants*, or in some situations the *Expert Assistants* will continue to pass the message to the *Personal Agents* involved to negotiate directly. However, if no solution can be found, the *Main Scheduler Agent* will start a persuasion process among the *Faculty Scheduler Agents* and this will pass through to the lower level in turn.

The analysis and design phase of this research used *Gaia* (Cernuzzi et al., 2004) as a methodology, while the evaluation method used interaction diagrams for presenting their solution.

Oprea designed the program to work through the levels of university organization (*Person, Department, Faculty and University*). However, although Personal Agent concerns the preferences of the staff, the needs of the students still have been omitted from this research. Other desirable constraints such as the shared courses are not recognized in this model.

In the evaluation part, Oprea employs interaction diagrams as a regular way to evaluate the multiagent system, to design the communication process between

agents, and to verify that the system executes the sequences of communication correctly. Several examples of message flow fragmentation are given to present how a communication process is analyzed. However, this paper does not mention any experimental results; nor details of the negotiation protocol, the persuasion protocol; and the way to resolve the worst case when no solution is found.

(3) **Strnad and Guid** (2007) also proposed a multiagent system for generating university course timetables. Their context is one of assigning computer-equipped classrooms to practical courses that are conducted by teaching assistants. Each assistant is responsible for more than one course, and more than one student class may attend each course. Student classes are divided into several smaller groups to meet the classroom capacity. The model consists of following types of agents:

- *Course Agent*: each agent represents an individual course which has to allocate teaching assistant and technical resources through negotiation.
- *Scheduling Agent*: takes responsibility for negotiating directly about combining specific courses.
- *Central Agent*: is responsible for communication among agents.

Each *Course Agent* has a list of initial assertions or requests which have been prioritized for expressing preference. At any time periods, the designed timetable can be evaluated by considering from the agent private evaluating value and the system global evaluating value, as $E^{(K)} = E_G^{(K)} + \sum E_i^{(K)}$ where E_G is global evaluation and E_i is course agent i private evaluation at any K time. The negotiation protocol is defined in steps as shown in Figure 3.1.

This approach aims to solve any conflicts through the negotiation protocol. Nevertheless, even though each agent in this model represents an individual course and tries to allocate teaching assistant and technical resources through negotiation, the paper does not give any details about what the preferences are and how they are prioritized. In addition, the needs of students are still neglected in this study.

(1) Generate the initial timetable $T^{(0)} = \cup_i Q_i = \cup_i P_i$ *when* Q *is all requests of agent* i *and* P *is the available periods for the fulfillment of the agent* i *actual allocation requirements.*
(2) Distribute agents into conflict groups
(3) Perform a round of incipient withdrawals
(4) Repeat.
4.1 Rebuild the conflict groups.
4.2 Run a concession round, until producing timetable $T^{(k+1)}$
4.3 If no concession was possible, run a renovation round.
4.4 Run a withdrawal round, until producing timetable $T^{(k+2)}$
(5) Until a feasible timetable is produced or no concession, renovation and withdrawal were executed in the last round.

Figure 3.1: The negotiation protocol from (Strnad and Guid, 2007)

For the experiments and evaluation, there are three data sets, which consist of a set of courses, a set of classrooms, a set of requests and a set of conflicts. Three approaches are set up to organize timetables; namely manual approach, multi agents approach, and genetic algorithm approach. Each organized timetable has been evaluated through function of $E^{(K)} = E_G^{(K)} + \sum E_i^{(K)}$ at any K time. The evaluated results show that timetables which are products of MAS give better evaluation values when comparing with the ones which are products of manual approach and of genetic algorithmic approach.

(4) **Gaspero et al.** (2004) designed an automatic scheduling system based on a multiagent architecture. In each department, there are three types of agents working cooperatively on different roles as follows:

- *Solver Agent* takes a role to generate timetable for its department by using two common local search techniques—*hill climbing* and *tabu search*. The *hill climbing* advantage is fast and provides some diversification, whereas t*abu search* intensifies its search in the promising areas of the search space. The timetable should meet *hard constraints* (H) and minimize violations of the *soft constraints* (S), using the following list:
  - *Lectures*: every lecture of courses must be organized. (H)
  - *Room Occupancy*: two distinct lectures cannot take place in the same room at the same time. (H)

  - *Teacher Conflicts*: lectures of courses with common teacher must be scheduled at different times. (H)
  - *Availabilities*: a course cannot be scheduled in a period when a lecturer is unavailable. (H)
  - *Student Conflicts*: lectures of courses with students in common should be scheduled at different times. (S)
  - *Room Capacity*: the number of students should be less than or equal the number of seats of the room. (S)

- *Negotiator Agent* sells and buys bids with other departments' negotiator agents. The buying bids are inferred directly from the Solver's results, while the selling bids are decided by itself according to the chosen strategies that involve selling everything that is not used, or selling only the useless resources, or using an intermediate behaviour.
- *Manager Agent* maintains quotation prices for the needed resources. The Negotiator Agent uses these quotations to make profitable bids when selling resources, whereas the Solver Agent uses them to estimate the cost when buying the missing resources. The maintained information provides probability values for buying a specific room x that has capacity c at timeslot t by presenting the price list p1, p2, ... that correspond with probability value v1, v2, ... in turn. That means $Quotation(t, c) = \{(p1, v1), (p2, v2), ...\}$.

The model employs three agents to work together on behalf of one department under the roles of *Solver*, *Negotiator* and *Manager*. The mission of these three agents is not only to organize optimal timetables for the department, but also to buy any necessary resources and sell unnecessary resources on. However, the considered constraints in this study are rather fundamental needs.

Two real instances from their university are used for experiments. Each instance comprises five departments. Timetabling for one week divides into twenty periods, including four two-hour slots per day.

- The first experiment aims to validate the Solver's capability in deliberating successful trades. Every department gets the same configuration. The Negotiator works as a bold trader, while the Manager is fed with

the bidding frequencies coming from a set of test runs. The experiments are run with values from no trading (α=0.0) to max trading (α=9.5). The risk level (α) is a member of following set {0.0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95} and for each risk level the experiment is performed twenty times, recording the outcomes of every department.

The first experiment result shows that the intermediate values of α lead to the better results, and give more uniform distribution of the gain among all departments. That means all departments in the market participate effectively.

- The second experiment is designed to investigate the behavior of a system when the agents have different levels of risk. Four agents have been set with their risk level at α=0.6, while one agent has been set to an extreme risk at α=0.2 or α=0.95.

The second experiment result shows that a low risk agent tends to make fewer buy bids at high price when comparing with a high risk agent, and so it gets more successful purchases.

(5) **Yan Yang and Paranjape** (2011) have sought to harness the capability of agents for implementing an intelligent decision-making system. Agent autonomy and a flexible communication methodology are the base for a solution to create the back-bone of the system. There are four types of agents that have been defined in this proposed model, where these are:

- *Course Agent*: takes a role as a course representative to maintain information such as course title, class times, classroom, instructor's name and teaching time preferences. It is a mobile agent having capabilities of negotiation and communication with other *Course Agents* through a *Signboard Agent* in order to find a mutually acceptable timetable.
- *Interface Agent*: provides a *GUI* interface to input course information or to input course information via data file when having many courses to enter in at a time.
- *Publisher Agent*: collects the organized timetables from *Signboard Agents*, sorts the course schedules and serves the result as a formatted text file.
- *Signboard Agent*: assigns available resources to *Course Agents*; it has the responsibility for against the breaking of *hard constraints* from *Course*

Agents' requests, and for giving an advice to *Course Agents* about conflicts. Moreover, it takes the role of coordinator to identify the conflicts of *Course Agents*; not only to record when the timetable is organized, but also to do updating when the *Course Agents* have changed any information on the organized timetable and to send the results out to the *Publish Agent* for publishing.

Essential and *desirable constraints* which are defined for this research are:

- *Essential Constraints*
  - No instructor teaches more than one course at the same time.
  - No more than one instructor can be scheduled to teach in the same classroom at the same timeslot
  - No more than one course can be scheduled in the same classroom at the same timeslot.
- *Desirable Constraints*
  - The time preferences which relate to each course.

This agent-based model is designed to work on a platform which represents a weekday, so there are five weekday platforms running simultaneously (under the assumption of five working days per week). On each weekday, there are *Course Agents* and *Signboard Agent*. *Course Agent* carries course information and finds suitable resources through negotiation with other *Course Agents* and cooperates with the *Signboard Agent*, while the *Signboard Agent* acts as a repository of the current schedule to record course information. Moreover, *Signboard Agent* provides a communication facility, allowing *Course Agents* to identify other courses competing for the same resources. A *Course Agent* is started on the platform corresponding to the weekday on which the course has the first preference to be scheduled. If acceptable course resources are not available on the current platform, the *Course Agent* will move to a new weekday platform which is the second most preference and then repeat the negotiation with other *Course Agents*. After the *Course Agent* has negotiated completely, a course timetable will be generated and will be kept in the *Signboard Agent*. An *Interface Agent* is responsible for inputting course

information, whereas a *Publish Agent* collects course schedules from individual *Signboard Agent* and creates the final output which is a complete weekly course timetable in the mandatory format.

This research shows three main results:

- How preference times cause an effect on the number of messages and timetabling duration (in sec), demonstrated by running an experiment with a range of different numbers of courses and different percentages of preference on each group of the courses.

  The first experimental result shows that the hard and soft constraints in most scenarios are satisfied except for the last two scenarios which were run under impossible soft constraints; that are the number of available resources was less than the number of preference timeslots; resulting in the number of messages and the scheduling time sharply increasing over these two scenarios.

- How preference times affect the number of messages and timetabling duration (in sec) by running an experiment that uses a fixed number of courses and fixed the percentage of preference at one hundred percent by ranging different number of preference timeslots in morning and afternoon instead.

  The second experiment result shows that the plotted graph between the duration of the experimental run times and the increasing number of inconsistent soft constraints is a parabola shape.

- How the number of instructors and number of per week workloads affect the number of message and number of timetabling duration(in sec), when the number of lectures is fixed, but the number of courses is varied, and a range of different number of morning and afternoon preferences are used.

  The third experiment result shows that for all the scenarios the *hard constraints* are satisfied and no violations occur. However,

> the plotted graph obviously shows that the increased number of lectures per course causes a direct affect on the chance of conflicts. Although the idea of harnessing agent capacities for implementing the decision part distinguishes this research from the other ones in this review, the constraints which have been considered here are fundamental needs. University course timetabling problems in real world are much more complex.

The Table 3.3 summarizes information about research in UCTPs which have been solved by agent-based technology.

Researchers	**Constraints**		**Models**	**Protocols**	**Exp**	**Res**	**Eva**
	E	D					
Kaplansky and Meisels, 2004	n/a	n/a	Scheduling Agents Rooms Agent	Implement CSP techniques in SAs for organizing and evaluating TBs; and well-defined negotiation for solving conflict among SAs in order to reach shared course slot.	n/a	n/a	n/a
Oprea, 2007	n/a	3	Personal Agents Department Agents Faculty Agents University Agent	Defined operations in every type of agents; including communication messages between any consecutive levels; and negotiation activities are set.	n/a	n/a	Interaction Diagram
Strnad and Guid, 2007	n/a	n/a	Course Agents Scheduling Agents Central Agent	Operations of each type of agents are defined; and negotiation protocol is set to reach agreement to occupy the resources.	Y	Y	Y
Gaspero et al., 2004	4	2	Solver Agent Negotiator Agent Manager Agent	Local search algorithms with different cost functions are defined for resources trading which follows by the negotiation mechanism	Y	Y	Y
Yan Yang and Paranjape, 2011	3	1	Course Agent Signboard Agent Interface Agent Publisher Agent	Define protocol by using agent's abilities to do heuristic decision making and apply agent's characters as autonomy, negotiation and cooperation to sort out the problem	Y	Y	Y

Table 3.3: Summary of papers that use agent-based technology for solving UCTP

Exp = Experiment; Res = Results; Eva = Evaluation

3.5 Classical Timetabling Techniques

As demonstrated in a survey of automated timetabling literature (Schaerf, 1999,Burke and Petrovic, 2002), there is a wide variety of methods for solving timetabling problems which have been investigated. They can be divided roughly as follows:

3.5.1 Sequential methods

The concept here is to start by ordering events using domain heuristics and to assign the events sequentially into valid time periods so that no events in the period are in conflict with each other. A conflict-free timetable can be modeled as a graph colouring algorithm. Given an undirected graph G = (V,E), the algorithm (Garey and Johnson, 1979) tries to find a partition of V into a minimum number of colour classes—$c_1, c_2, ..., c_k$— where no two vertices can be in the same colour class if there is an edge between them. According to the sequential methods, the priorities of the events that have the largest number of conflicts (high colour degree) would be scheduled first.
The Figure 3.2 and Figure 3.3 provide an example of a simple timetable which can be solved by graph coloring technique (Lewis, 2007). Each event is represented as a vertex. Each edge between any pair of vertices corresponds to a pair of events that cannot be assigned at the same timeslot. Each timeslot corresponds to a colour; therefore the feasible solution requires no more than the number of timeslots which means the possible colors should not exceed the number of timeslots.
Initial mapping is shown in Figure 3.2.

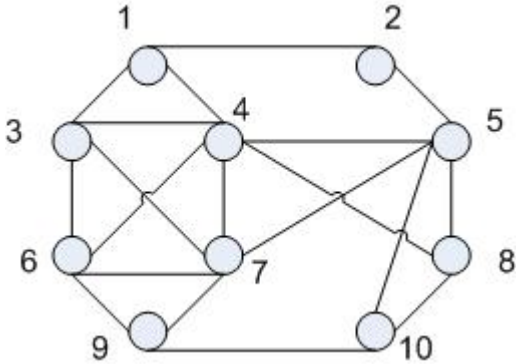

Figure 3.2: Initial mapping graph from (Lewis, 2007)

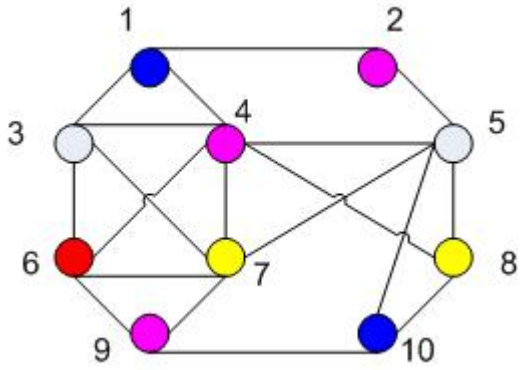

Figure 3.3: Assigned colours and mapped graph from (Lewis, 2007)

The result after the vertices and edges have been assigned colours and mapped between different timeslots is shown in Figure 3.3.

The graph coloring solution can be converted to a valid timetable in Table 3.4, where each colour represents a timeslot and no pair of adjacent vertices have been assigned to the same timeslot.

1	2	3	4	5
Event 1	Event 4	Event 3	Event 7	Event 6
Event 10	Event 9	Event 5	Event 8	
	Event 2			

Table 3.4: The converted timetable from (Lewis, 2007)

3.5.2 Meta-heuristic methods

There are various approaches using meta-heuristics that have been developed for solving timetabling problems over the last two decades. These include:

- *Tabu search*: based on (Glover, 1989, Glover and Laguna, 1993), this technique aims to reach an optimal solution for a problems. It is based on the notion of a neighbour: given P is an optimization problem, let S be the search space of P, and let f be the objective function to minimize. A function N, which depends on

the structure of the specific problem, assigns to each feasible solution $s \in S$ its neighbourhood $N(s) \subseteq S$. Each solution $s^{'} \in N(s)$ is called a neighbour of s.

The algorithm starts from an initial solution s_{init} , which is generated at random, then enters into a loop to navigate the search space so that it can explore a subset of V of the neighborhood *N(s)* of the current solution s; The member of V that gives the minimum value of the object function becomes the new solution. The *Tabu search* algorithm is summarized in Figure 3.4.

1. *Randomly set an initial solution i in S set i* = i and k =0*
2. *Set k = k+1 and generate a subset V of solution in N(i,k)*
3. *Find a best j in V* (i.e f(j) ≤ f(k) for any k in V) and set i = j*
4. *If f(i) < f(i*) then set i* = i*
5. *If a stopping condition is met then stop else go to step 2*

Figure 3.4: *Tabu search* algorithm from (Hertz et al., 1993)

- *Simulated annealing*: this uses a probabilistic local search to find optimal solutions as Tabu search by Kirkpatrick et al (1983), with an extended version described in (Aarts and Korst, 1989).

  The algorithm starts by generating an initial solution, then enters a loop that generates a neighbor of the current solution randomly by iteration. Let Δ be the difference in the objective function between the new solution and the current one. If $\Delta<0$ the new solution is accepted and becomes the current solution, else the new solution is accepted as probability $e^{-\Delta/\mathrm{T}}$, where T is a parameter, called *temperature*. The temperature T is set to an appropriately high initial value T_0, and after a fixed number of iterations, the temperature is decreased by the cooling rate a, and $T_n = a \times T_{n-1}$ where $0 \leq a \leq 1$. The *Simulated annealing* algorithm is summarized in Figure 3.5.

- *Genetic Algorithms*: *genetic algorithm*s are not a local search technique, but a technique for optimizing problems, which has been proposed in(Davis, 1991,Michalewicz, 1994)

1. Randomly set an initial solution i in S set i = i and n = 0*
2. Set n = n+1 and generate a subset V of solution in N(i,k)
3. Find a best j in V (i.e f(j) ≤f(n) for any n in V) and set i = j*
4. If f(i) < f(i) then set i* = i else f(i) is accepted as probability* $e^{-\Delta/T}$ *when* $T_n = a \times T_{n-1}$ *where* $0 \leq a \leq 1$
5. If a stopping condition is met then stop else go to step 2

Figure 3.5: *Simulated annealing* algorithm from(Hertz et al., 1993)

The algorithm initiates a set of solutions $\left\{s_1^0, ..., s_n^0\right\}$ by randomness, called population at time 0.

At time t, the value of the objective function is computed for each solution s_i^t to give a set of solutions $\left\{s_1^t, ..., s_n^t\right\}$. Based on a random weighting, n elements of the population at time t are selected. Some solutions in higher probability may be selected more than once if the results give a better value of the objective function. Therefore, the best solutions get more copies, while the worse solutions probably die off at time *t+1* with a new set of solutions $\left\{s_1^{t+1}, ..., s_n^{t+1}\right\}$

The method terminates either when it generates a fixed number of populations, or when the best solution reaches a certain value of the objective function, or when the algorithm does not make any progress for a certain number of iterations. *Genetic Algorithms* are summarized in Figure 3.6.

1. Generate an initial population of candidate solutions $\left\{s_1^0, ..., s_n^0\right\}$ *and t = 0*
2. Apply fitness function to population members $\left\{s_1^t, ..., s_n^t\right\}$ *when t = t*
3. Choose the fitness member to form the new population
4. Apply genetic operators and generate new population $\left\{s_1^{t+1}, ..., s_n^{t+1}\right\}$ *and t = t +1 then go to step 2*

Figure 3.6: *Genetic Algorithm* from (Davis, 1991,Michalewicz, 1994)

3.5.3 Constraint-Based Programming Methods

Constraint-Based Programming Methods search all possible solutions which must satisfy all the constraints (conditions, properties) through the defined *Constraint Satisfaction*

Problem (*CSP*) model (Liu et al., 2002). The strategy is based upon:

1. a set of n variables $X = \{x_1, x_2, ...\}$
2. a finite set d_i is the possible values of each variable x_i and set $D = \{d_1, d, ...\}$
3. and a set of constraints over X is $C = \{c_1, c_2, ...\}$

A solution of *CSP* is an assignment of valid values (from each domain) for each variable and satisfies every constraint at the same time. The results might be

1. one solution with no preference
2. all solutions
3. an optimal or at least a good solution (by defining objective function in term of some or all of the variables)

The *CSP* is a combinatorial problem which can be solved by searching. Although the model consists of a large number of variables and runs through a simple algorithm, the searching of all possible combinations may still take a long time. The following section presents several techniques from the constraint programming paradigm to solve *CSP*, such as *backtracking*, a *consistency technique*, *branch-and-bound*, *global constraints* and *constraint hierarchies*.

3.5.3.1 Techniques in Constraint Programming

- *Backtracking* uses a depth-first-search; at each level it selects a variable and extends the partial assignment of the previous level by selecting a value for that variable. If the new assignment violates some constraints, then it tries with another value for the same variable. If all the values for a given variable have been tried without success, then the algorithm backtracks to the previous assigned variable and selects another value (Torres et al., 2006). Figure 3.7 presents an example of the *backtracking technique*.

- *Consistency technique* is a technique to remove inconsistent values from the domains of the variables. Normally, the *consistency technique* and *backtracking* are used together in order to prune the search space (Torres et al., 2006).

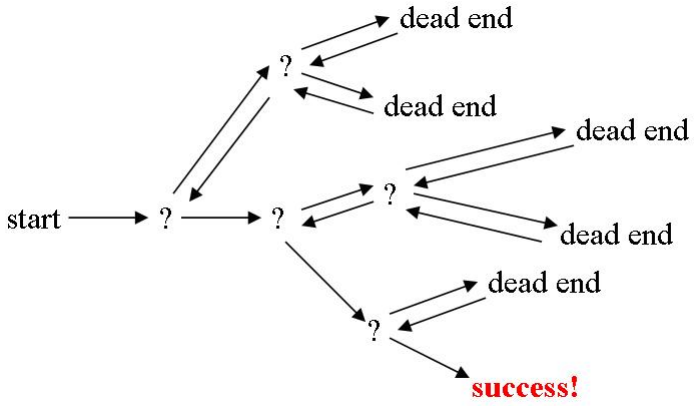

Figure 3.7: *Backtracking* from (Matuszek, 2009)

Figure 3.8 is an example which applies consistency technique for finding the consistent solutions.

(a) The graph has no solutions, as if v1 is Red/Green, then v2 must be Green/Red; and v3 must not be Red and Green. However, there are only two possible choices in v3— Red or Green. Therefore, there has no consistent solution in the graph.

(b) The graph has two solutions (Blue, Red, Green) and (Blue, Green, Red), assumed that if v1 is Blue, then v2 is Red and v3 is Green; or might be v2 is Green and v3 is Red. Therefore, there are two possible consistent solutions.

(c) The graph has exactly one solution (Blue, Red, Green), assumed that if v1 is Blue and v3 is not Red, then v2 must be Red. Therefore, there has only one consistent solution.

- *Branch and Bound (B&B)* is an algorithm for finding an optimal solution. It searches the complete space of solutions for a given problem for the best solution by overcoming the exponentially increasing number of potential solutions (Clausen, 1999). A *branch-and-bound* procedure requires two tools.

  - The first one is a splitting procedure for branching, since its recursive application defines a tree structure (the search tree) whose nodes are the subsets of *S*.

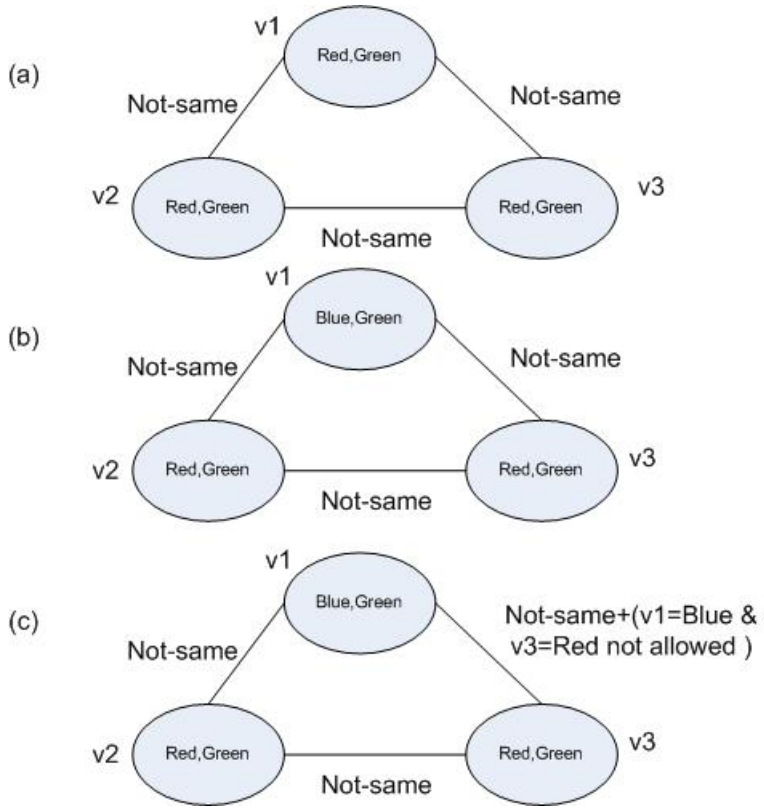

Figure 3.8: Arc Consistencies from (Berwick, 2008)

– Another tool is a procedure that computes upper and lower bounds, called *bounding*.

The recursion stops when the current candidate set S is reduced to a single element; or also when the upper bound for set S matches the lower bound. Figure 3.9 shows an example of branch-and-bound.

- *Global constraint* is a constraint that captures a relation between a non-fixed number of variables. *Global constraints* aim to facilitate the work of the constraint solver by providing them with a better view of the structure of the problem. The drawback of propagation-search technique is the search component will enumerate all possible assignments of values to the variables until it either finds a solution to

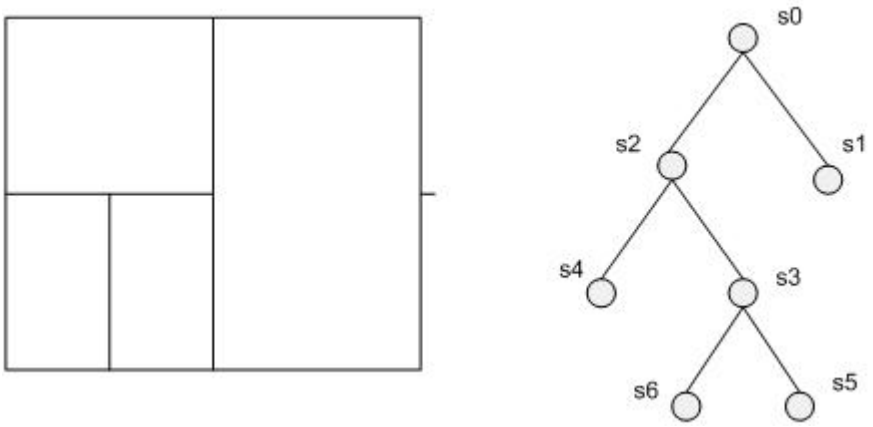

Figure 3.9: *Branch and Bound* example in *R2*, after 3 iterations. The partition of the original rectangle is shown at left; the associated binary tree is shown at right. From (Boyd and Mattingley, 2007)

the *CSP* or exhausts all possible assignments and concludes that a solution does not exist. In the worse case, an exhaustive search has an exponential time complexity. Where partial assignment cannot lead to a solution, the idea of filtering useless variable domains before searching would increase efficiency. If a value is useless with respect to one of the constraints, then it is also useless with respect to the whole *CSP*, but not vice versa (van Hoeve and Katriel, 2006). The *global constraint* can be posted before all the constrained variables are known which brings advantage of earlier domain pruning mainly for a system where not all information is necessarily known (Al-Maqtari et al., 2006).

- *Constraint Hierarchies* is the way to identify which solution of a *CSP* does not exist or where we cannot evaluation which variables satisfy all the constraints, by specifying constraints with hierarchical strengths or preferences for describing such over constraint system. The declaration specifies not only the constraints that require holding, but also weaker constraints at an arbitrary but finite number of strengths. The weaker strength of constraints helps to find a solution, while it does not permit the weakest constraint to influence the result. *Constraint hierarchies* define the comparators to select solutions (the best assignment of values to particular variables) via minimizing errors of violated constraints (Al-Maqtari et al., 2006).

3.6 Summary of research in Educational Timetabling Problems

A list of information which gives details about solutions to educational timetabling problems and which technique has been deployed in each is presented in the Table 3.5:

Table 3.5: Summary of research in Educational Timetabling Problems

Techniques	**Article Name**
Graph Coloring	1.1An introduction to timetabling (Werra, 1985) 1.2 Split Vertices in Vertex Colouring and Their Application in Developing a Solution to the Faculty Timetable Problem (Selim, 1988)
Local Search	***Tabu Search*** 2.1Finding a feasible course schedule using Tabu search (Hertz, 1992) 2.2 Tabu Search Techniques for Examination Timetabling (Gaspero and Schaerf, 2000) 2.3 A tabu search heuristic for a university timetabling problem (Arntzen and Lokketangen, 2003) ***Simulated annealing*** 2.4 A Comparison of Annealing Techniques for Academic Course Scheduling (Elmohamed and Fox, 1997) 2.5 A robust simulated annealing based examination timetabling system (Thompson and Dowsland, 1998)

Genetic Algorithm	3.1 A Genetic Algorithm Based University Timetabling System (Burke et al., 1994) 3.2 A smart genetic algorithm for university timetabling (Rich, 1995) 3.3 Timetabling the Classes of an Entire University with an Evolutionary Algorithm (Paechter et al., 1998) 3.4 Timetabling using a Steady State Genetic Algorithm (Ozcan and Alkan, 2002)
Constraint Programming	4.1 Limited–resource scheduling by generalized rule-based system (Meisels et al., 1991) 4.2 Constraint–based Timetabling with Student Schedules (Rudova and Matyska, 2000) 4.3 University Course Timetabling Using Constraint Handling Rules (Abdennadher and Marte, 2000) 4.4 Automated University Timetabling (Torres et al., 2006) 4.5 Modeling and solution of a complex university course timetabling problem (Murray and Rudova, 2007) 4.6 Investigating Constraint-Based Reasoning for University Timetabling Problem.(Sheau et al., 2009)
Swarm Intelligent	5.1 A MAX-MIN Ant System for the University Course Timetabling Problem (Socha et al., 2002)
Neural Network	6.1 Hopfield neural networks for timetabling: formulations, methods, and comparative results.(Smith et al., 2003)

Multi Agents	7.1 Negotiation among scheduling agents for distributed timetabling (Kaplansky and Meisels, 2004) 7.2 A MultiAgent Architecture for Distributed Course Timetabling (Gaspero et al., 2004) 7.3 An agent based general solution model for the course timetabling problem (Yang et al., 2006) 7.4 MAS_UP–UCT: A multi–Agent System for University Course Timetable Scheduling (Oprea, 2007) 7.5 A multi-agent system for university course timetabling (Strnad and Guid, 2007) 7.6 Implementation of class timetabling using multi agents (M.Nandhini and S.Kanmani, 2009) 7.7A multi-agent system for course timetabling (Yang and Paranjape, 2011)
Mathematical Modeling	8.1 Mathematical programming models and algorithms for a class–faculty assignment problem (Al-Yakoob and Sherali, 2005)

Apart from solving the university course timetabling problem, all of the above techniques have also been applied in various domains such as the meeting scheduler (Sen, 1997, Ernst et al., 2008), scheduling of commercial intervals in a radio or television broadcasting programs (Galitsky, 1999), scheduling of industrial operations (Murthy et al., 1997,Cowling et al., 2003), transportation scheduling (Montana et al., 2000, Parkes and Ungar, 2001), and appointment scheduler in hospitals (Vermeulen et al., 2008). Some papers also propose a general solution model for the course timetabling problem (Yang et al., 2006), a universal method for solving timetabling problems from different domains (Norberciak, 2006) or a model for solving complex distributed constrained problems (Al-Maqtari et al., 2009)

For the issue of fairness, some papers are found such as the research of Al-Yakoob and Sherali (2005) is set for the aim to minimize the individual and collective dissatisfaction of faculty members in a fair fashion; and the research of Oon and Lim (2002) designs a

multi-player game to solve university exam timetabling by imposing a fair and setting rules for the game.

In order to distinguish one from another research which using agents to address the university course timetabling problem , we summarize the above researches in some criteria i.e. the number of considered constraints, aims, models and defined protocols. The summary is shown in Table 3.6.

Researchers	Constraint		Aims		Models	Protocols
	E	D	A	F		
Kaplansky and Meisels, 2004	n/a	n/a	Y	n/a	Scheduling Agents, Rooms Agent	Implement CSP techniques in SAs for organizing and evaluating TBs; and well-defined negotiation for solving conflict among SAs in order to reach shared course slot.
Gaspero et al., 2004	4	2	Y	n/a	Solver Agent, Negotiator Agent, Manager Agent	Local search algorithms with different cost functions are defined for resources trading which follows by the negotiation mechanism
Oprea, 2007	n/a	3	Y	n/a	Personal Agents Department Agents Faculty Agents University Agent	Defined operations in every type of agents; including communication messages between any consecutive levels; and negotiation activities are set.
Strnad and Guid, 2007	n/a	n/a	Y	n/a	Course Agents, Scheduling Agents, Central Agent	Operations of each type of agents are defined; and negotiation protocol is set to reach agreement to occupy the resources.
Yang and Paranjape, 2011	3	1	Y	n/a	Course Agents, Signboard Agents, Interface Agent, Publisher Agent	Define protocol by using agent's abilities to do heuristic decision making and apply agent's characters as autonomy, negotiation and cooperation to sort out the problem.
Wangmaeteekul, 2011	4	3	Y	Y	Year-Program Agents Rooms Agent	Round-Robin algorithm controls automated timetabling process which uses for organizing and evaluating the timetabling results; and negotiation mechanism is designed for reallocating resources when any YPA facing constraint-mismatched problem.

Table 3.6: Summary of research in UCTPs that use agents for solving including our research

E = Essential; D = Desirable; A = Resource Allocation; F = Fairness

3.7 Summary

The nature of the university courses timetabling problem has been reviewed in this chapter. It discusses the types of timetables used in the education domain. The lists of essential constraints and desirable constraints which are involved in such a problem are described. A set of existing problems which forms a gap between theoretical and practical aspects of university timetabling is addressed. The potential roles for using intelligent agents for solving timetabling domain are identified. Some research that has applied agents for solving university courses timetabling over the last decade has been summarized. Then, a wide variety of classical methods which have been used in researches of timetabling problems were reviewed. We summarized the concepts and algorithms by ranging them by techniques. At the end of this chapter, two summary tables are presented. The first one presents a list of research in educational timetabling problems, grouped by solving techniques; the other one presents a list of research in university course timetabling problems that have applied agents, by considering each in terms of the number of constraints, aims *(achievement/fairness)*, types of agents in model and protocol design.

Chapter 4

Research Method

This chapter starts with the reasons for addressing the problem of university course timetabling by using an intelligent-agents model, and its evaluation via the defined problem formula from *ITC-2007*. Then, the following sections describe the details of intelligent agent architecture, the allocation protocol, the forms of organization for allocation and negotiation, and the university structural scenario that was defined for *ITC-2007*.

4.1 Introduction

Over the last decade, many published papers have demonstrated that both intelligent agents and classical timetabling techniques are important tools for resolving university course timetabling problems. From the literature, we also found that each research group has proposed its own model, architecture, and mechanism to deal with this problem for one specific context or set of data instances. Because university course timetabling problem is one of the real–world complex problems. It contains both great complexity and a variety of requirements. So, it is impossible to write a solution formula that suits all requirements of all institutes; as each one also has its own rules, features, costs and constraints. From the research point of view, it is therefore difficult to compare between the results from different studies. This is a drawback of this research field, preventing fair comparisons and impacting assessment the absolute quality of a solution method proposed.

In order to solve the above weakness, a group of researchers has joined together and organized the *International Timetabling Competitions (ITC)*. So far, *ITC-1* in *2002* and

ITC-2 in *2007* have aimed to bridge the gap between research and practice by introducing a significant degree of complexity in the tracks, so that the employed formulations are close to the real world needs (even not all aspects but a degree of generality is kept) and data sets are also taken from the real world. The success of competitions has been confirmed from the wide awareness about them in the community. Moreover, to provide a baseline for others, the competition's results have been posted, by reporting the least penalty value, the name of solving technique and the name of competitor who is the record holder for each instance. However, from the information available, we have not identified any competitor who had solved the problem by employing intelligent agents; despite several papers in the research literature insisting that agent technology has the potential to perform university course timetabling tasks.

Therefore, in this study the aim has been to investigate the use of an agent-based model for solving the university course timetabling problem by generating resource allocation solutions for all participants in a system, under the constraints that have been defined for the *ITC-2007* competition, and also seeking to provide fair allocation among the participants. The distributed timetabling model has been adapted from the model proposed by Kaplansky and Meisels (2004), by adding more flexibility and more roles to each type of agent, which have been redefined as *Year-Program Agent (YPA)* and *Rooms Agent (RA)* respectively. The details of the proposed model will be presented in section 4.2, this being designed to work under the university structural scenario that is described in section 4.5.

In order to evaluate the proposed model, we have used data instances from the 2007 *International Timetabling Competition (ITC-2007)*, allowing us to compare our solutions with those obtained using other approaches. More details of the *ITC-2007* competition are presented in Chapter 6.

4.2 Intelligent Agents Architecture

In the real world, actual university course timetabling focuses on achieving a reasonable distribution of courses over the timetable with minimal conflicts for students progressing normally through their educational programme, which has been categorized in terms/years. In order to develop an effective and powerful agent solution to any problems, the solution must map effectively into the innate and intrinsic characteristics and nature of agents and agent systems (Yang and Paranjape, 2011).Therefore, for this research we

have observed the above principle by mapping agent autonomy into

- *Year Programme autonomy*: working autonomously to organize a set of courses/modules that particular year students of that programme have to study.
- *Rooms Administration autonomy*: working autonomously to manage a set of room resources which are provided for every teaching activity of every curriculum from all parts of the university.

As presented in Chapter1, the architecture that we propose for solving distributed university course timetabling problems is one that is comprised of two types of agents—*Year-Programme Agent* and *the Rooms Agent* working together; by assigning various roles in each type as follows:

- A *Year-Programme Agent (YPA)* is assigned to the task of generating the timetable for one level of a particular programme. It is responsible for organizing the courses/modules which the programme's students have to take in one particular term. The timetable should satisfy all of the essential constraints and satisfy the desirable constraints as many as possible. Another crucial role of the *Year-Programme Agent* is therefore to collaborate with other *YPAs* to sort out any allocation problems by reallocating rooms between them.
- The *Rooms Agent (RA)* manages the rooms (resources) and will book the requested room when that room is vacant. It also coordinates the *YPAs* to work together in order to avoid overlaps across the shared modules. Moreover, the *RA* takes responsibility for ordering access to the resources by the *YPAs* in such a way as to ensure fairness.

The total number of agents in this system is therefore the number of *YPAs*, which required for the different degree programmes and levels, plus one *RA*. It is a flexible architecture in that it makes no assumption about the length of a degree programme (2, 3, 4 years). The model aims to mimic a human scheduler's behavior in the real world by performing two stages of allocation:

(1) Between *YPAs* and the *RA* to provide an initial allocation of resources (the *initial allocation phase—S1*).

(2) Between *YPAs* to refine the allocation of rooms (the *negotiation phase—S2*).

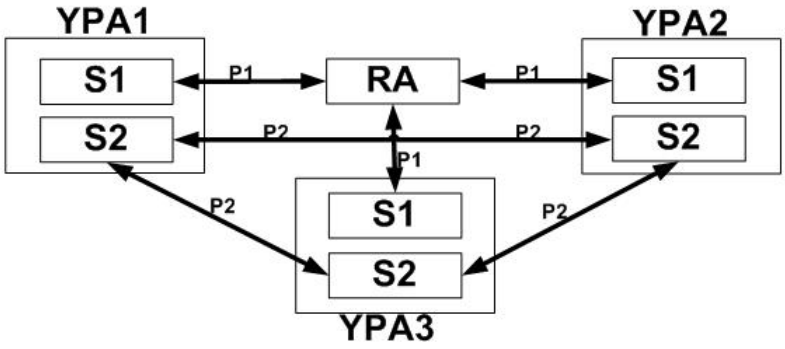

Figure 4.1: The *YPAs* & *RA* architecture

Figure 4.1 shows an example of the chosen architecture, comprising of three *YPAs* and one *RA*. Each *YPA* is composed of two segments of code that perform the actions required for the *initial allocation phase* (*S1*) and the *negotiation phase* (*S2*), while *P1* and *P2* represent the interaction protocols involved in *S1* and *S2* respectively.

4.3 Allocation Protocols

As above mention, there are two stages of allocation from the defined architecture these are:

4.3.1 The initial allocation phase

For the *initial allocation phase*, we have investigated the use of two algorithms for interactive allocation of resources between *YPAs* and the *RA* in order to identify how best to optimise the chance for each *YPA* to acquire the desired set of resources. As agent has the characteristic of being autonomous, it has the ability to prioritize and choose the resources which it prefers over others. We would like to demonstrate that agent's *'selfish'* approach can influence the fairness issue by comparing two forms for allocating resources.

First-In-First-Out (FIFO) and *Round-Robin (RR)* are the two contrasting allocation algorithms which have been chosen to show how using different rules for allocation could cause different degrees of fairness. *FIFO* does not embody any mean of *'fairness'* as sharing of resources, whereas *RR* seeks to achieve an equitable distribution through

'taking turns' for resources.

4.3.1.1 *First-In-First-Out (FIFO) allocation*

Here, each *YPA* in turn makes a complete set of requests to the *RA*. Hence, those *YPAs* which are near the head of the queue can be expected to get the best choice of rooms and times. Each *YPA* has to inform the *RA* when its allocation is completed and the *RA* then goes on to service the requests from the next *YPA* that is waiting in the queue to make its requests. These steps are repeated continuously until the waiting list is empty, as shown in Figure 4.2.Order of *YPAs* in *First-In-First-Out* allocation.

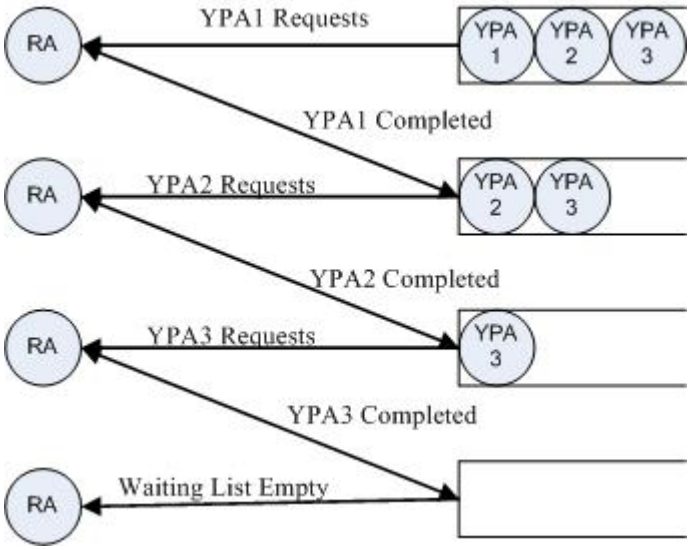

Figure 4.2: Order of allocation to *YPAs* in *First-In-First-Out* allocation

4.3.1.2 *Round-Robin (RR) allocation*

This version of the allocation algorithm is designed to work in rounds. During a round, each *YPA* in turn presents one request to the *RA*, with the requests from the set of *YPAs* being re-ordered dynamically for the next round by using the degree of satisfaction achieved in the current round, using a model that is adapted from *Utilitarian Social Welfare* in section 2.4.2.1. Each *YPA* evaluates this for the resource it got from the former rounds. The *RA* takes part in the activities of queuing the *YPAs*; and launches

them to work as an ordered series. The rule that the *RA* uses to schedule requests from *YPAs* is that of "*the last getting first*". That means the *YPA* that got the lowest total of satisfaction values (*Sv*) from the former rounds will be the first one to request resources in the next round, as shown in Figure 4.3. The order of requests for first round sequence is arranged randomly.

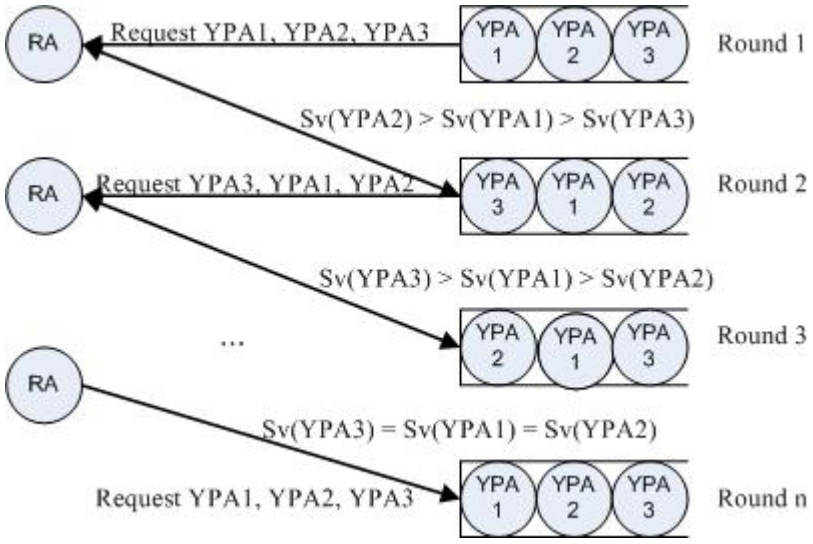

Figure 4.3: Order of allocation to *YPAs* in *Round-Robin* allocation

For both *FIFO* and *RR* allocations, the resource requesting strategy of each *YPA* tries to choose the most suitable timeslot and the best-fitting room from the set of resources which are available at any given moment. The *YPA* chooses the subject which has the highest number of constraints to organize first. In the case that the number of constraints is equal, the subject/lecture which has the large number of students will be organized first. After that it prioritizes timeslots by ranking the preference values from high to low and then chooses the most preferential timeslot(s), where the preference value is determined by many factors such as global time preference information, avoiding the organized subjects timeslots which belonging to the same curriculum or teacher, avoiding clashes among shared course timeslots, avoiding same day assignment if that subject has been organized, and adjacent to the booked timeslots etc. It then requests the *RA* to retrieve a list of vacant rooms for the chosen timeslot(s) that are big enough to contain the number of students taking that subject. When the set of rooms is returned from the

RA, the *YPA* then chooses the best-fit room by considering from the left space which means the smaller space left the best-fit room is; and the *YPA* then requests the *RA* to book that room. However, if the *YPA* is unable to find a resource which meets its constraints under the specified timeslot(s), then the next most preferential timeslot(s) from the prioritized list will be chosen for searching a new set of rooms. This process continues repeatedly until the preference prioritized list is empty. If eventually the *YPA* is unable to find a resource to meet its needs, then that subject/lecture will be moved to be a member of the *constraint-mismatched set*. The detail of initial allocation is described in section 5.3 and the interaction diagram is presented in Figure 5.7.

4.3.2 The negotiation phase

This phase takes place when either of the following two situations arises.

- There is a non-empty set of constraint-mismatched subjects
- It is necessary to reallocate rooms

The negotiation phase is initiated by the *RA*. For example assuming that *YPA—Ag1* has identified a constraint-mismatched subject, the *RA* then allows *Ag1* to solve the problem itself. *Ag1* prioritizes a set of timeslots according to its preferred values from high to low in the prioritized-timeslot list. From the most preferable timeslot, then, *Ag1* broadcasts to all other *YPAs* to find an agent that now occupies a room(s) which can contain the number of students that *Ag1* requires at the specificed timeslot, and that agent is able to help it. Assuming that *YPA Ag2* is now occupying a room—X which meets above property. That means room X is big enough to contain the number of students that *Ag1* requires; and *Ag2* is able to find a new room —Y for its own subject and it is willing to switch to new room Y. Then, *Ag2* replies with the "*Proposing Relief*" message to *Ag1*. After receiving proposals from all other *YPAs*, *Ag1* chooses the best-fitting room under the specified timeslot. If we assume that the chosen room is X, which now it is owned by *Ag2*, *Ag1* then sends a message to request room X from *Ag2*. After receiving the request from *Ag1*, *Ag2* frees room X and switches to book room Y instead. However, if *Ag2* is unable to find a room Y as a replacement, then it has a right to refuse *Ag1*'s request. In case that *Ag1* receives rejections from all other *YPAs*; it will then choose next timeslot in the list and re-broadcast. Such processes continue repeatedly until the constraint-mismatched problem is solved or the prioritized-timeslot list is empty. The

detail of negotiation design is described in section 5.4 and the interaction diagram is presented in Figure 5.10.

4.4 Processes in Organization

In terms of organizing, there are two distinct procedures that have been defined to organize how the *Year-Programme Agents* and the *Rooms Agent* work together for setting up a set of timetables. These two forms are:

4.4.1 Sequential

This procedure starts by running the *initial allocation phase* (*S1*) until this phase terminates, and then running the *negotiation phase* (*S2*) subsequently, if any constraint-mismatched elements still remain, as shown in Figure 4.4.

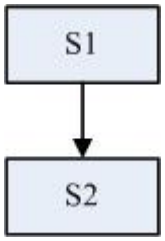

Figure 4.4: Sequential scheduling

4.4.2 Interleaved

This procedure starts by running the *initial allocation phase* (*S1*) on a step-by-step basis, interleaving this with the *negotiation phase* (*S2*) if any *YPA* faces a constraint-mismatched element problem. After the problem has been resolved, the allocation phase will continue, as shown in Figure 4.5.

From the above descriptions of allocating and scheduling rooms, we have two ways of allocating rooms and another two different ways of implementing the negotiation process. That means we have four possible principles for implementing the university course timetables in this study, as shown in Table 4.1. These are:

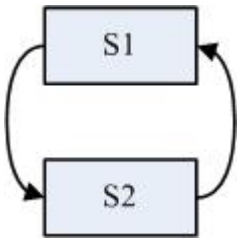

Figure 4.5: Interleaved scheduling

Organizing Form	**Allocation Form**	
	FIFO	**RR**
Sequential	*FIFOSeq*	*RRSeq*
Interleaved	*FIFOInt*	*RRInt*

Table 4.1: Models of interaction between agents

- *First-In-First-Out & Sequential (FIFOSeq)*
- *Round-Robin & Sequential (RRSeq)*
- *First-In-First-Out & Interleaved (FIFOInt)*
- *Round-Robin & Interleaved (RRInt)*

To investigate these in more detail, a set of hypotheses have been chosen in order to determine whether these four different principles give different results or not for constraint achievement and fairness issues.

Some questions which are used for developing this set of hypotheses are:

- Can these four principles achieve the defined problem goals?
- Can use of a negotiation phase solve constraint-mismatched resource allocation problem?
- Do different initial sequences lead to different results?
- Among the four principles, which one generally gives the smallest/largest penalty value? Which one will produce the smallest/largest standard deviation of satisfied values?
- Does increasing the number of agents in the system lead to larger penalty values?

- Is our proposed model better than other techniques in term of constraint achievement?
- Among the four principles, which one gives the best result in terms of constraint achievement and meeting the aim of fairness?

The following hypotheses have been investigated:

- All four proposed multiagent principles are able to achieve the university course timetabling formula and constraints which were defined for the *International Timetabling Competition 2007*.
- The number of successfully organized lectures will be increased after applying the *negotiation phase* following the *initial allocation phase* of both *FIFO* and *RR* algorithms.
- Different initial sequences of *YPAs* will produce different results.
- Among the four principles, the *First-In-First-Out & Sequential* principle will always give the smallest penalty value when compared with the other principles.
- Among the four principles, the *First-In-First-Out & Interleaved* principle will always give the largest penalty value when compared with the other principles.
- Among the four principles, the *Round-Robin & Interleaved* principle will always give the smallest standard deviation of satisfied values when compared with the other principles.
- Among the four principles, the *First-In-First-Out & Sequential* principle will always give the largest standard deviation of satisfied values when compared with the other principles.
- Increasing the number of agents will lead to larger penalty values.
- The proposed principles generally produce smaller penalty values when compared with the *ITC-2007* reference results.
- The *Round Robin & Interleaved* principle is most likely to produce better solutions in terms of both smallest penalty value and best degree of fairness when comparing to the other principles.

4.5 University Structural Scenario

This research aims to mimic the real world university timetabling process. Most universities define levels of organization as follows:

- *University Level*: a university is composed of many faculties.
- *Faculty Level*: any faculty is joined up from many departments.
- *Department Level*: any department has one or more curriculum(s)/programme(s) for which it needs to provide teaching activities.
- *Curriculum/Programme Level*: any curriculum/programme is composed of many modules/courses that are grouped by a number of years for which that curriculum/programme's students have to study. Here, a Set of Modules/Courses of the Year Programme means a list of modules/courses for which that Year Programme students have to study.
- *A Module/Course* composes of one or more lecture(s). It is taught by a lecturer who is good at the module/course. Normally, the lecturer and the module/course belong to the same department.
- *A Shared Module/Course* is a module/course that is taken by students from several different curricula/programmes, who attend in the same room at the same time.

The resources involved in the university course timetabling problems are: *students, lecturers, rooms, courses and times.* Each type of resource must belong to one specific level of a university's organization.

- *Students*: any student must belong to one specific curriculum/programme under the department and faculty to which that curriculum/programme belongs.
- *Lecturers*: any lecturer must be a member of one specific department under the faculty to which that department belongs.
- *Rooms*: any room must be a possession of the university which is managed by the *administrative room resource section* which belongs directly to the university level. Any department that needs to use any room resources must book the required resource from this section. For this research, every room has the same characteristics

apart from room-name and room-capacity, which are different form one to another. Each room has been expressed in terms of its name and its fixed number of available seats.

- *Courses/Modules*: any course/module must belong to one programme/curriculum which the department owning it must take responsibility for organizing the course/module by providing a lecturer and scheduling that course/module timetable. For the shared course/module, the one who takes a responsibility to organize this shared course/module should be the department, which provides a lecturer to teach on, to take a role to set up the timetable, while the other shared departments take a role in avoiding clashes with the shared slots when they are organizing their own timetables.

- *Times*: normally the number of teaching days per week is the same as the working days. Each day is split into a fixed number of timeslots, which is the same for each day of the week. A period is a pair composed of a day and a timeslot. The total number of scheduling periods is the product of days multiplies by the day timeslots. In this research, we assume that scheduling periods of each *YPA* in the system are the same.

4.6 Methodology

In this study, there were two research methods that could be adapted from as a list. These were: 1) *Experiments research method* 2) *Case study research method*

4.6.1 Experiments

An *experiment* is a research method to investigate the relationships between *cause* and *effect*. It aims to *verify* or *falsify* or *establish the validity* of the *hypothesis* which is a statement in the form of "Factor A causes B". The experiment is run in a controlled environment and careful measurements of the outcomes are taken (Oates, 2006). Some typical characteristics of research based on experiments are:

- *Formulate hypothesis*

- Carry on the *experiment* repeatedly

  - Observation or measurement of a factor
  - Manipulate circumstances
  - Re-observation or re-measurement of the factor to identify any changes

- *Prove or disprove the hypothesis*
- *Identify causal factors* by discovering which factor is the cause (independent variable) and which is the effect (dependent variable)
- *Explain the casual link* between cause and effect

In summary, experimental approach is characterized by having control over the research environment. Some variables are then manipulated to observe their effect on other variables.

4.6.2 Case study

Case study is a research method that focuses on one instance and provides an intensive investigation into as much as possible about the phenomenon of interest. The main aim of this type of research is to obtain insight into details in a situation where the instance normally involves complex relationships and processes. It is used when the boundaries between phenomenon and context are not evidently clear (Yin, 2009). The typical characteristics of research that is based on the use of one or more case studies are:

- *Focus on depth rather than breadth*: obtain as much as possible detail for the one instance of the phenomenon under investigation.
- *Natural setting*: examine the instance in its natural setting; not in a laboratory or any other artificial situation.
- *Holistic study*: focus on the complex relationships and processes to see how they are interconnected and inter-related; rather than trying to isolate individual factors.
- *Multiple sources and methods*: quantitative and qualitative data obtained from multiple sources are used.

In summary, the case-study method is an in depth approach. It is generally used when researchers need to find or explain '*how*' and '*why*' this outcome occurs in some particular cases or situations.

In this research, we need to demonstrate that the proposed principles have the potential to solve the university course timetabling formula and achieve the set of constraints which are defined for the *International Timetable Competition 2007*. The datasets for this, which are provided on the competition website, were used to test our four different principles to determine how well each can solve the problem, when compared with the reference results obtained from the website. In other words, we might say "the proposed principle X causes ? result". The validating value is the outcome which we need for this study.

Our study is empirical in nature, coming up with conclusions which are capable of being verified by observation or experiment by using datasets from the single source. It seems straightforward and is not a complicated process. Therefore, the most suitable research method is to use controlled experiments rather than the case-study method, which focuses on intensive investigation of one instance. An experiment starts from formulating hypotheses about the probable results, then conducts on experiment in order to get enough facts (data) to prove or disprove the hypothesis. The experimental design must manipulate cause factors so as to bring forth the effect by controlling over the variables under the study. An Experimental approach is appropriate when proof is sought that certain variables affect other variables in some way, which seems to match with the needs of this research.

4.7 Summary

This chapter provides the reasons for addressing the university course timetabling through a distributed intelligent agents system and how this will be evaluated via problem formula, constraints and datasets which were defined for *ITC-2007*. It describes the proposed intelligent agent architecture, resource allocation protocols and organizing forms. The last two sections present the university structural scenario which is defined corresponding with *ITC-2007* problem formula; and two research methodologies are reviewed.

Chapter 5

Agent Design

The previous chapter presented an overview of the system, while this chapter describes the detailed designs of the agents. It starts with the principles of design; and presents the designs for both a *Year-Program Agent* and the *Rooms Agent*. We describe the protocol for the *initial allocation phase* and the protocol for the *negotiation phase*. The last two sections explain about how the date-time slots preferences are prioritized and how the quality of allocated resources is assessed by defining a set of satisfied values.

5.1 Principles of Design

By aiming to imitate how the human planner works, the model has been designed around the architecture and characteristics of intelligent agents, in which all properties are innate inside the agent itself. An agent makes decisions based on the *incomplete information* available to it, its own *selfish* manner, its own *capability* and the *circumstances* that it meets at any given moment. We define each characteristic of the agents in an appropriate part of the protocol design in order to ensure some outcomes such as:

- Exploiting an agent's *selfish* characteristic through the allocation protocol, by which each agent aims to choose the resources (timeslots and rooms) that will best match its requirements at any given moment.
- Applying an agent's *collaboration* property for the reallocation of room resources. When some agents in the system need help to sort out their allocation problem, the remaining agents collaborate to assist the agents which now are facing constraint-mismatched resource allocation problems.

- Applying an agent's *negotiation* property to resolve conflicts when desired resources are limited. The one who currently owns a resource requested by another will hold it until it finds a new resource which meets its needs and that gives a satisfaction value not less than the current one. It will then free the resource; making it possible to achieve agreement in negotiation. Otherwise, the resource's owner has the option of refusing the request.

- Applying an agent's *coordination* property in order to avoid clashes among shared courses. We assign the *RA* to take a role to be a coordinator; a set of agents who need to share the same course must then ask the *RA* to retrieve information about the booked slots of the other agents within the same group and try to avoid using those booked slots when scheduling their own subjects.

Every agent in the system works toward the same goal of achieving an organized set of courses under its responsibility, which these meet all defined essential constraints and satisfy as many desirable constraints as possible. Every decision about reasoning is made by the agent itself, and is determined from the present state that the system is in at any given time, by trying to choose the best-fit resource to occupy first. If any agent cannot obtain resources that meet its needs, the other agents must participate in the task of relieving the problem. This model is designed on the base of individual heuristic decision-making by each agent and the timetabling mechanisms are based on *First-In-First-Out*, *Round Robin* and *extended Contract Net Protocol*. Each agent operates to implement its own timetable with a number of finite loops and the system will eventually be terminated. The distributed university course timetables are then the result of participation from every member in the system, with each one undertaking its own timetable organization, and cooperating with others to relieve problems.

5.2 Agent Design

Here we examine the designs for the different types of agent.

5.2.1 *Year-Program Agent (YPA)*

The goal of each *YPA* is to find an appropriate timetable for the set of courses that form its responsibility. The range of main functions which is embedded in each *YPA* are:

- *Create table*: create its own empty timetable

- *Select course*: select a course by choosing the most complex course's constraints to organize first; in the case that there are many courses which have the same number of constraints; the course which has the maximum number of students will be organized first.
- *Organizing a lecture*: a set of lectures of each course will be organized a single lecture at a time.
- *Searching timeslots*: search available timeslots by excluding 'must avoid' timeslots such as the 'booked' timeslots, the timeslots from the 'enforced constraints', the 'unavailable teacher' timeslots, and the 'shared course' timeslots.
- *Prioritizing timeslots*: prioritize available timeslots according to the requirements of the problem formula. The available timeslots which have properties to achieve minimal working days and adjacent to the booked timeslots will be put on the top of the other available timeslots.
- *Searching rooms*: search for available rooms according to the prioritized timeslots; one timeslot at a time.
- *Prioritizing rooms*: prioritize the set of rooms according to their properties from small to large sizes; the smallest room size still can contain the number of students for that lecture by putting it on the top of the list.
- *Matching*: match lecture(L), timeslot(T) and room(R) together by choosing the combination returning the maximum satisfy value. Whenever the satisfy value does not meet the expectation value, the second next timeslot in the timeslot list will be considered and search for available rooms of that second next timeslot. Then, the matching of these three components will do repeatedly.
- *Booking*: book timeslot and room for that lecture will be done if the satisfy value (Sv) is greater than the expectation value(ExpV).
- *Updating in timetable*: update the booked timeslot, room, and lecture in its own timetable and report the totally satisfy value to *RA*.

The hierarchical structure diagram for a *YPA* is presented in Figure 5.1 and a flowchart of processes of *YPA* while organizing a lecture in the *initial allocation phase* is presented in Figure 5.2.

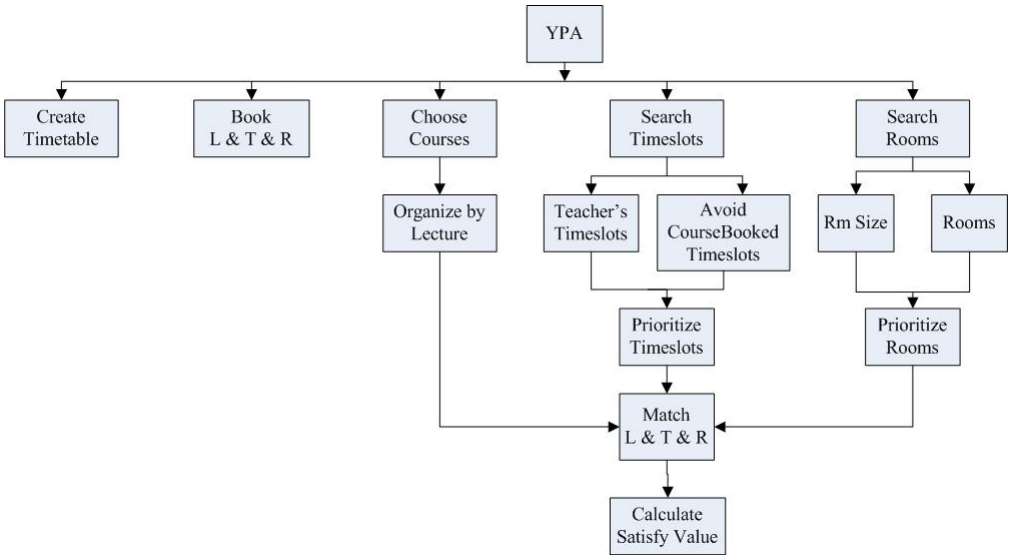

Figure 5.1: The hierarchical structure diagram of a YPA
L = Lecture; T = Timeslot; R = Room

Initial knowledge for each YPA's belief part has been determined from the information that was provided in the input file of *ITC-2007*. An example set of information which is necessary for timetabling is presented in Figure 5.3.

5.2.2 *Rooms Agent (RA)*

The goals of the *RA* are to allocate room to each *YPA* according to the requests of that *YPA*, and to maintain information that is necessary for organizing the timetables of the *YPAs*. The main functions which are embedded in the *RA* are:

- *Assigning room*: assign a vacant room in response to the request of *YPA*
- *Searching information*:
  - search available room-size information for a *YPA* for the aim of sorting the rooms order
  - Search available rooms information so that a *YPA* can prioritize the resources of rooms

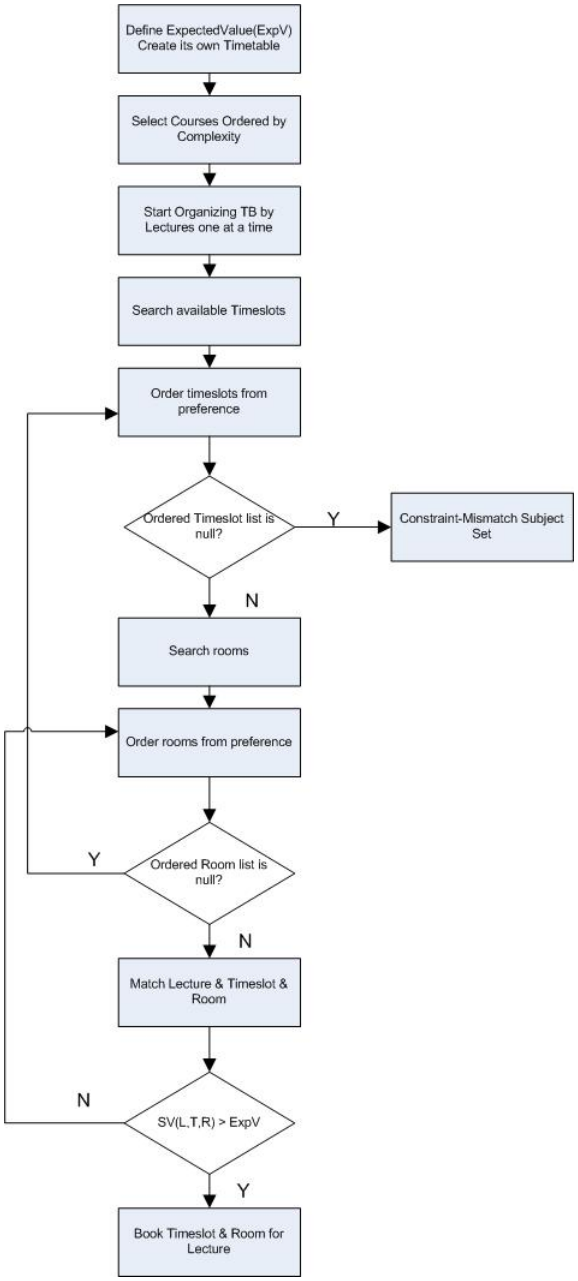

Figure 5.2: A lecture organizing process in the initial allocation phase
L = Lecture; T = Timeslot; R = Room; ExpV = Expectation Value; Sv = Satisfy Value

```
thisCurCompOf(["C1","C2",... "Cn"]). //a list of courses in this curriculum
thisCosInCurs("C1",["Q1",...,"Qn"]). //a list of curricula which course C1 is
shared
course(1,"C1","Teacher","#Lecture","MinWorkingDays","#Student").
...                                          // number of lectures for course C1
course(n,"C1","Teacher","#Lecture","MinWorkingDays","#Student").
avoid("C1","D1","T1").
...                                  //avoid course C1 from this specific date-time slot
avoid("C1","Dx","Ty").
avcnflcourse("C1",["C2",...,"Cn"]).//a list of courses which course C1 must avoid
```

Figure 5.3: Initial knowledge in *YPA*

  - Search unavailable timeslots for a teacher who teaches on the course that the *YPA* is organizing, in order to avoid a conflict
  - Search for information about booked timeslots of courses which the *YPA* needs to avoid such as the booked timeslots of the shared courses, the timeslots which are unavailable teacher. All this information is needed for the *YPA* to avoid any conflicts

- *Releasing YPAs*: release the *YPAs* to organize timetables in sequence according to the *First-In-First-Out* or *Round-Robin* algorithm
- *Reordering sequence*: reorder the sequence of *YPAs* under "*the last getting first*" rule when applying the *Round-Robin* algorithm in the initial allocation phase.

The hierarchical structure diagram of the *RA* is presented in Figure 5.4

Initiating knowledge in the RA's belief part has been defined. A set of information that is necessary for timetabling is presented in Figure 5.5

5.2.2.1 *Controlled sequence in RA*

One of the *RA*'s main roles is to take responsibility for managing a set of *YPAs* to work as an ordered sequence.
As there are two possible models employed for the initial allocating resources, the *RA* controls the sequence of *YPAs* in two different ways. These are:

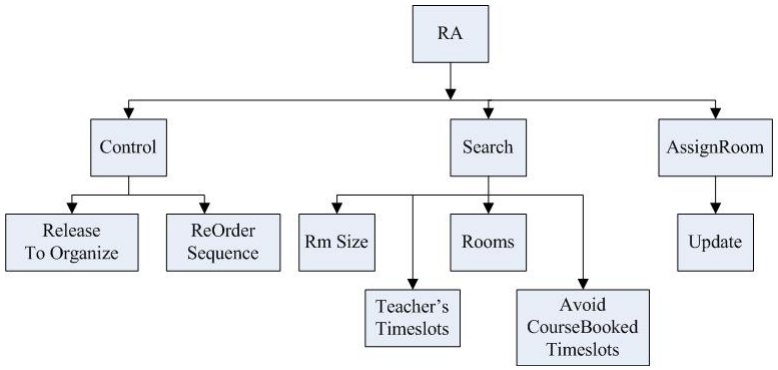

Figure 5.4: The hierarchical structure diagram of the *RA*

```
tableSz(rows,columns). //The size of timetables which is used in whole system
room("RoomID", "Capacity","D1","T1","AVAILABLE").
...                                          //All available rooms information
room("RoomID", "Capacity","Dx","Ty","AVAILABLE").
```

Figure 5.5: Initiating knowledge in *RA*

- ***FIFO controlled loop*** is shown as a diagram in Figure 5.6(a). The *RA* starts by ordering *YPAs* into a list at random. Then, the *RA* releases the topmost YPA to organize its courses. During the organizing process, the *RA* assists that *YPA* until all courses of the *YPA* have been organized. After the *YPA* completed its tasks, the *RA* releases the next *YPA* in the list to organize its courses. This process continues until the list of *YPAs* is empty, and the *RA* then terminates the organizing loops.

- ***RR controlled loop*** is shown as a diagram in Figure 5.6(b). As this allocation type is designed to work as a round robin, the *RA* starts by ordering *YPAs* into a randomly ordered list for the first round. Then, the *RA* releases the top *YPA* to organize one lecture. During the organizing process, the *RA* also assists that *YPA* by providing the necessary information for organizing the single lecture. After the *YPA* has completed this, the *RA* releases the second next *YPA* of the list to organize one lecture. This process continues until the *YPA* list in the *RA* is empty which means every *YPA* has had an equal chance to organize one lecture in that round. Then, the *RA* reorders the sequence of *YPAs* according to "*the last getting first*" rule, that means that the *YPA* that got the highest satisfied values of resources from the former rounds will be the last one of the list to request resources in the next

round, and then puts the *YPAs* back in the list if each one still has some courses left for organizing. After that the next round will be performed if the reordered list is not empty. Otherwise, the *RA* terminates the organizing loops. The detail of how it calculates the satisfied value for each round is presented in section 5.6.

As we know, there are two main elements in each YPA—the *initial allocation* part (*S1*) and the *negotiation* part (*S2*). The detailed designs are in section 5.3 and 5.4.

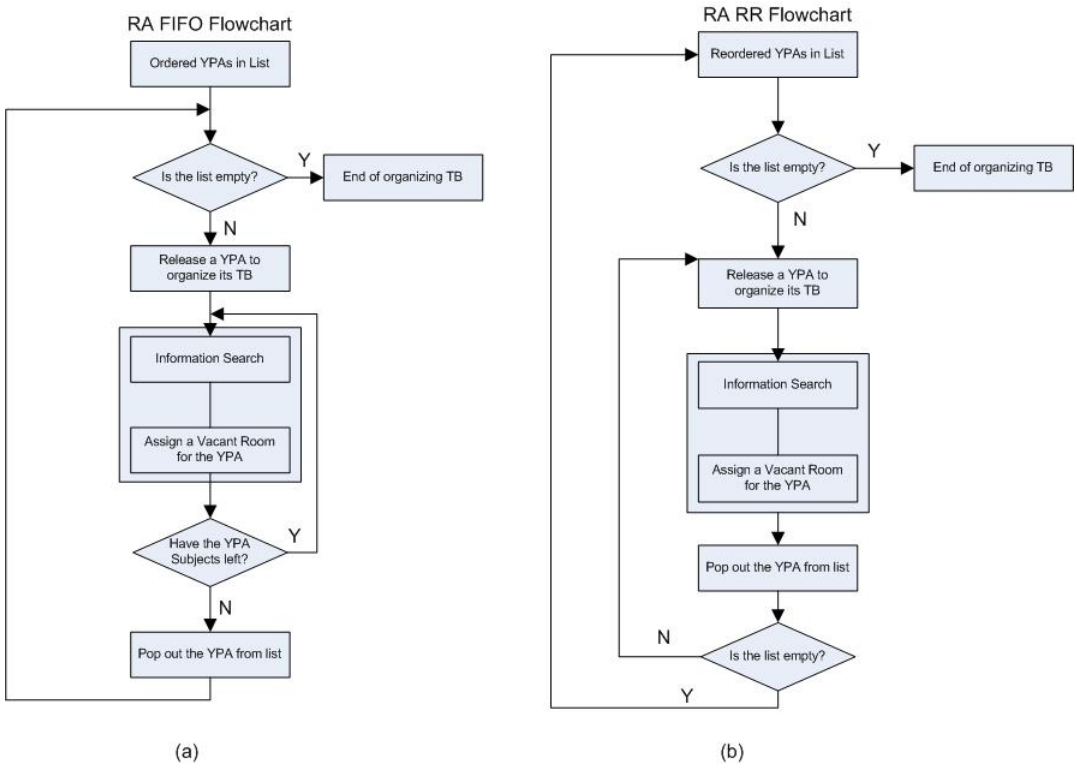

Figure 5.6: (a) *First-In-First-Out* controlled loop (b) *Round-Robin* controlled loop

5.3 Detailed Design for Initial Allocation between *YPA* & *RA*

For this section, we present the design of the protocol for the *initial allocation* part when a *YPA* is organizing the scheduling of a lecture. The sequence of processes is similar

for either the *First-In-First-Out* model or the *Round-Robin* model. For organizing any lecture, the sequence is composed of three nested loops; as shown in Figure 5.7.

Outermost Loop: Loop1

1. Search for the most complex courses to organize
2. Search all booked lectures which belong under the same curriculum (shared course lectures)
3. Search booked lectures from its own timetable.
4. Search date-time slots which must be avoided for that course, according to the constraint requirement.
5. Search date-time slots for which that course's teacher is unavailable.
6. Search date-time slots of courses under other curricula which have at least one course shared with this organizing course. The host of organizing course needs to know the date-time slots information, so that the organizer is able to avoid the conflicts.
7. Search the days of the week which have been booked for this course.

At this point, the *YPA* has information about date-time slots which have already been booked and so needs to avoid when it is organizing the current course's lectures. From the above information, the *YPA* is able to identify which date-time slots are available and suitable to assign for a lecture. Moreover, consideration of some information—such as avoiding isolated date-time slots or trying to reach the minimal working days requirement—leads the *YPA* to classify or prioritize timeslots according to definition of the cardinal preference structure in Chapter 2. The defined utility function maps between the set of available date-time slots and the number of constraints that each available date-time slot can be achieved. The details of the mapping will be presented in more detail in section 5.5. When there are prioritized date-time slots available, then *YPA* enters Loop2.

Outer Loop: Loop2

1. The available date-time slots are ordered by ranging from highest preference to lowest preference.

2. The *YPA* searches for available rooms which are large enough to contain the number of students who are taking this course.

When there are available rooms of sufficient sizes, then the *YPA* enters Loop3. Otherwise, the next date-time slot and available rooms will be searched. However, in the case in which the last date-time slot in the list has been considered and the *YPA* is unable to match that date-time slot with any room, then this lecture will be moved to be a member of the constraint mismatch subjects set.

Innermost Loop: Loop3

1. The available rooms are ordered by ranging from smallest to largest size.

2. The *YPA* matches the chosen date-time slot and the smallest room size and then asks the *RA* for searching whether that date-time slot of the smallest rooms is available.

3. After retriving the returned result from the *RA*.

   3.1 If the date-time slot of the smallest room is also available, then it sends the *RA* a request to book that date-time slot; and updates the information in the *YPA*'s timetable.

   3.2 Otherwise, it matches that date-time slot against the second next smallest room size of the list, then asks the *RA* for searching whether that date-time slot of that second next smallest rooms is available (Loop3 step 3).

   3.3 This process continues repeatedly until the date-time slot matches any available room; or in the worst case, the *YPA* is unable to match any available room at the needed date-time slot, then the next date-time slot (Loop 2 step 1) will be considered. The sequences of matching the prioritized resources are presented in Figure 5.8.

   The utility value that returns from the *YPA*'s chosen resources should then be maximum; as the set of resources has already been prioritized from high utility values to low utility values. So, the set of matched resources will be the best at that given moment.

The above protocol is designed based on the rules of good mechanism design that an individual agent should reach his/her maximum value of the sum of the utility functions, as described in (Rosenschein and Zlotkin, 1994,Sandholm, 1999) in section 2.4.4.

5.4 Detailed Design for Negotiation between a *YPA* & other *YPAs* and the *RA*

For this section, we present the details of the protocol for the negotiation part. It is designed to refine the allocation of rooms. After the initial allocation part, if the constraints mismatch subject set is not empty, then the *RA* releases any *YPA* which is now facing constraints mismatch problem to solve the problem by itself. The negotiation protocol we have designed here is adapted from the *Contract-Net Protocol (CNP)* which was proposed by Smith (1980). The *YPA* that is facing the constraint mismatch problem takes the role of the '*manager*', and the '*contractors*' in our protocol are the *YPAs* who identify themselves as being able to help relieve the problem. The process is presented in Figure 5.9 and follows these steps:

1. Broadcasting for help
2. Resolving the mismatch
3. Awarding to the proposer

5.4.1 Broadcasting for help

1. From Figure 5.10, assuming that *YPA1* is unable to find a resource (room) which matches its course's constraints, it then moves that lecture to be a member of the constraint-mismatched subject set.
2. When the *RA* finds that the constraint-mismatched subject set is not empty, it releases the *YPA1* that is now in trouble, beginning the negotiation phase by sending it a message to start negotiation.
3. After the *YPA1* has got the message to begin negotiation, it searches for available date-time slots in order to organize the lecture that is in trouble. Some information needs to be gathered for this stage, which is:

   3.1 Search all booked lectures which belong under the same curriculum (shared course lectures)

3.2 Search booked lectures from its own timetable.

3.3 Search date-time slots which must be avoided according to the constraint requirement.

3.4 Search date-time slots which that course's teacher is unavailable.

3.5 Search date-time slots of courses under other curricula which have at least one course that is shared with this course. If *YPA1* is the host that is organizing the course, then it needs to know the information about date-time slots for the others that share the same course, so that it is able to avoid conflicts in date-time slots.

At this point, *YPA1* has information about the date-time slots that have been booked and have to be avoided. From the above information, *YPA1* is able to infer which date-time slots are available and suitable for being assigned for this lecture. Moreover, some information such as avoiding isolated date-time slots or trying to reach the minimal working days requirement may lead *YPA1* to classify or prioritize timeslots according to the cardinal preference structure. By defining a utility function, it maps between the set of date-time slots and the number of constraints which each date-time slot can be achieved. The details of the mapping will be presented in more detail in section 5.5. Then, it chooses the most preferable date-time slot and information about the number of students that take this course are provided in its broadcast for help; then *YPA1* waits for every participant replying with its answer before moving onwards to the next stage.

5.4.2 Resolving the mismatch

After the other participants receive *YPA1*'s request, which asks for room resources for that specific date-time slot, every participant examines its own resources to see whether it has any room occupied on that particular date-time slot, and if this is large enough to contain the specified number of students or not. The two possible results here are: "*Found*"or "*Not Found*".

- If the result is "*Not Found*", the participant replies instantly with the message "*Refusing Help*"
- If the answer is "*Found*", the participant will seek to find replacement resources for its own lecture which currently occupies the date-time slot and room which match the needs of *YPA1*. Some information is needed for this stage which the participant needs to consider, including

1. Search for information about the current booked lecture
2. Search for all booked lectures which belong under the same curriculum (shared course lectures)
3. Search for booked lectures from its own timetable.
4. Search for date-time slots which must be avoided according to the constraint requirement.
5. Search for date-time slots which that course's teacher is unavailable.
6. Search for date-time slots of courses under other curricula which have at least one course there that is shared with this course. If the participant is the host that is organizing the course, then it needs to know the information about date-time slots for the others that share the same course, so that it is able to avoid conflicts in the date-time slots.
7. Search available rooms which are large enough to contain the number of students who are taking this course.
8. Search for the days of the week which have been booked for this course.
9. The participant uses above information to determine and chooses the best-fit resources to organize its own lecture. If such resources exist, it means that the resources are able to meet all essential constraints and the number of desirable constraints they reached is not less than the currently occupied resources reached. Then, the participant will reply with the message "*Proposing Relief*"; otherwise, the participant *YPA* has a privilege to reply with the message "*Refusing Help*".

5.4.3 Awarding switching task to the proposer

1. After *YPA1* has received replies from every participant in the system, it chooses one participant from set of participants which replied with a"*Proposing Relief*" answer. The chosen participant would be the one who now occupied the resource which gives the highest satisfied value to *YPA1*. It then sends a '*revoke room request*' to that participant *YPA* to ask for the room that it needs.

2. After that participant receives the '*revoke room request*', it sends a request to the *RA* to book the new room which has been identified, and organizes its lecture in

the new date-time slot and the new room. Then, it replies to *YPA1* with the '*room revoked*' response.

3. When *YPA1* receives the '*room revoked*' response, it sends a request to the *RA* to book the revoked room which is now available for *YPA1* to occupy. So here, the conflict has been sorted out by the *negotiation* protocol. However, it may also be the case that *YPA1* receives "*Refusing Help*" replies from all of the participants in system. The second next date-time slot will then be chosen for broadcasting next. This process will continue repeatedly until either *YPA1* has found any participant that is willing to switch to the replace resources and frees the occupied resources for *YPA1*; or in the worst case there is no one can help after *YPA1* has tried until the preferable date-time slot list is empty. Then, the unorganized lecture will be left for a human planner to resolve.

5.5 Date-Time slots Preference

In this section, we provide more details about how to classify or prioritize timeslots according to the cardinal preference structure which consists of an evaluation (*utility*) function $u : X \rightarrow Val$, when Val is either a set of numerical values (*quantitative*) or a scale of qualitative values. After the *YPA* receives information about the date-time slots which have been booked and those which need to avoid, the *YPA* creates a "*virtual table*", which has the same dimensions as timetable it is currently implementing; and marks both the booked and the needed to avoid slots. Here, it is assumed that the marked virtual table is presented as in Figure 5.11(a). The letter "*B*" means the booked slot and the letter "*A*" means the necessary avoiding slots. Prioritization of available slots also depends on the constraints of the problem formula such as avoiding isolated date-time slots, or trying to reach the requirement from minimal working days. If the current organizing course has not been booked in the *YPA* timetable, then the prioritized available date-time slots arrangement will be run from slot number 1 to slot number 25 as shown in Figure 5.11 (b). Otherwise, the prioritized available data-time slots arrangement will be run from slot number 1 to slot number 25 as shown in Figure 5.11 (c), if the current organizing lecture of this course has already been booked in the *YPA* timetable. The prioritized results are calculated from an evaluation function—termed *achievement(date,time)*—that maps between the set of date-time slots and the total number of constraints which that slot

can be achieved.

$$achievement(date, time) = \sum_{i=1}^{n} AchvedCon_i(date, time)$$

Given

$$AchvedCon_i(date, time) =$$
$$\begin{cases} 0\, when\, Constraint\, i\, at\, date - time\, slot\, has\, not\, been\, acieved \\ \\ 1 \;\; when\, Constraint\, i\, at\, date - time\, slot\, has\, been\, achieved \end{cases}$$

When

- Constraint$_1$ = all booked lectures under the same curriculum (shared course lectures) have been avoided.
- Constraint$_2$ = all booked lectures from the organizer *YPA*'s timetable have been avoided.
- Constraint$_3$ = all (date-time slots)' which defined from problem formula have been avoided.
- Constraint$_4$ = all (date-time slots)' which that course's teacher is unavailable have been avoided.
- Constraint$_5$ = all (date-time slots)' under other curriculums which have at least one course there shared with this organizing course have been avoided.
- Constraint$_6$ = the date is different from the days of the week which have been booked for this course.
- Constraint$_7$ = this date-time slot is adjacent to the other booked slots

After calculating the quantitative achievements of each available date-time slot through achievement function, a set of *achievement(date,time)* values will be returned and then prioritized them ranging from the maximum value to the minimum value. In the case that many date-time slots have the same *achievement(date,time)* value, the slots will be prioritized from left to right and high to low. An example of prioritized date-time slots ranging *achievement(date,time)* value from high to low is presented as an ordering of values from 1 to 25 in Figure 5.11 (b) and 5.11 (c).

Unused Seats					
%	**0-10**	**11-20**	**21-30**	**31-40**	**41-50**
SatisfiedValue	10	9	8	7	6
%	**51-60**	**61-70**	**71-80**	**81-90**	**91-100**
SatisfiedValue	5	4	3	2	1

Table 5.1: The definition of satisfied values
10 = the highest level of satisfaction; 1 = the lowest level of satisfaction

5.6 Calculating the Satisfied Value

After the resources have been allocated, every YPA has to calculate a "*satisfied value*" for the resources it has obtained. Here, the satisfied value is determined from the number of unused seats in the allocated room. The unused seats is the underuse result of room capacity and student numbers. The value of the satisfied function *Satisfiedvalue(%ofUnusedSeats)* is the result of the percentage of the left seats which is defined in Table 5.1. The fewer unused seats that there are, the higher the satisfed value will be.

5.7 Summary

This chapter has described the details of the inter-agent protocols which are necessary for this research. It provides a description of the hierarchical structure of both a *Year-Program Agent* and the *Rooms Agent* by describing all the functions they perform. Then, the details of the interaction protocol design have been presented. The first is the interaction between *a YPA* and *the RA* used in the initial resource allocation phase, while the second is the interaction between a *YPA* and the *other YPAs* required for the negotiation phase.

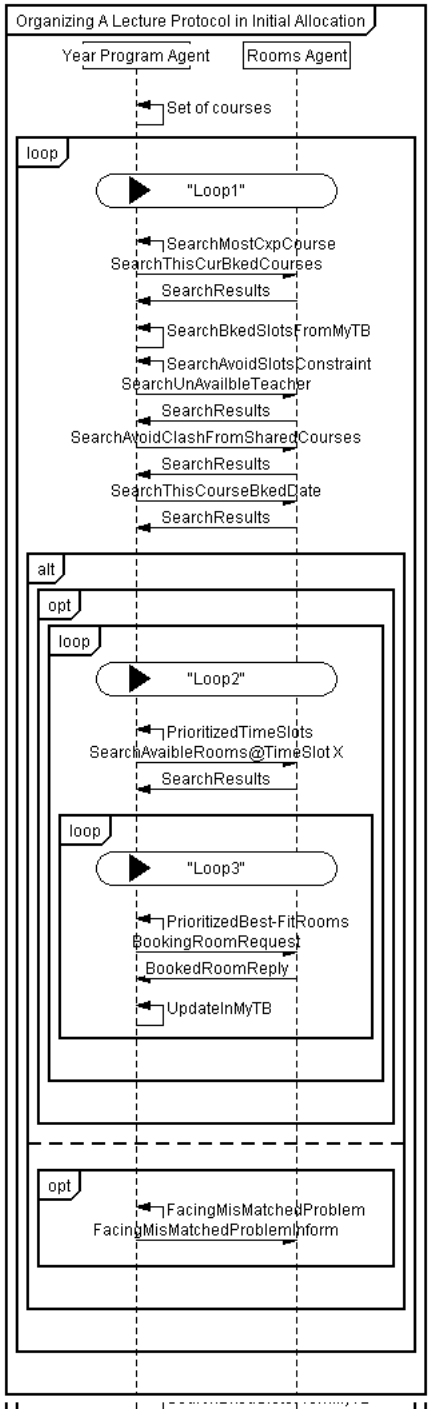

Figure 5.7: Interaction between *YPA* & *RA* in the initial allocation phase

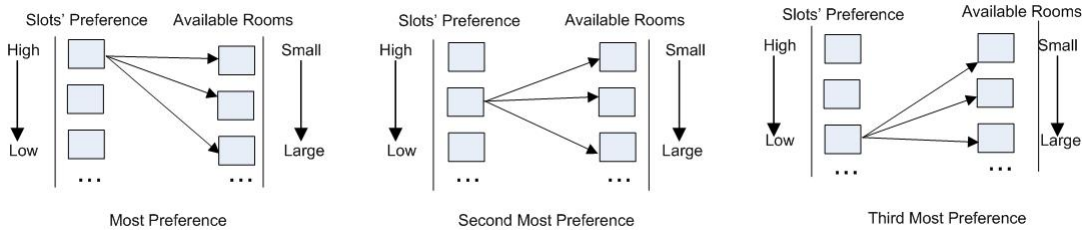

Figure 5.8: Prioritized resource matching

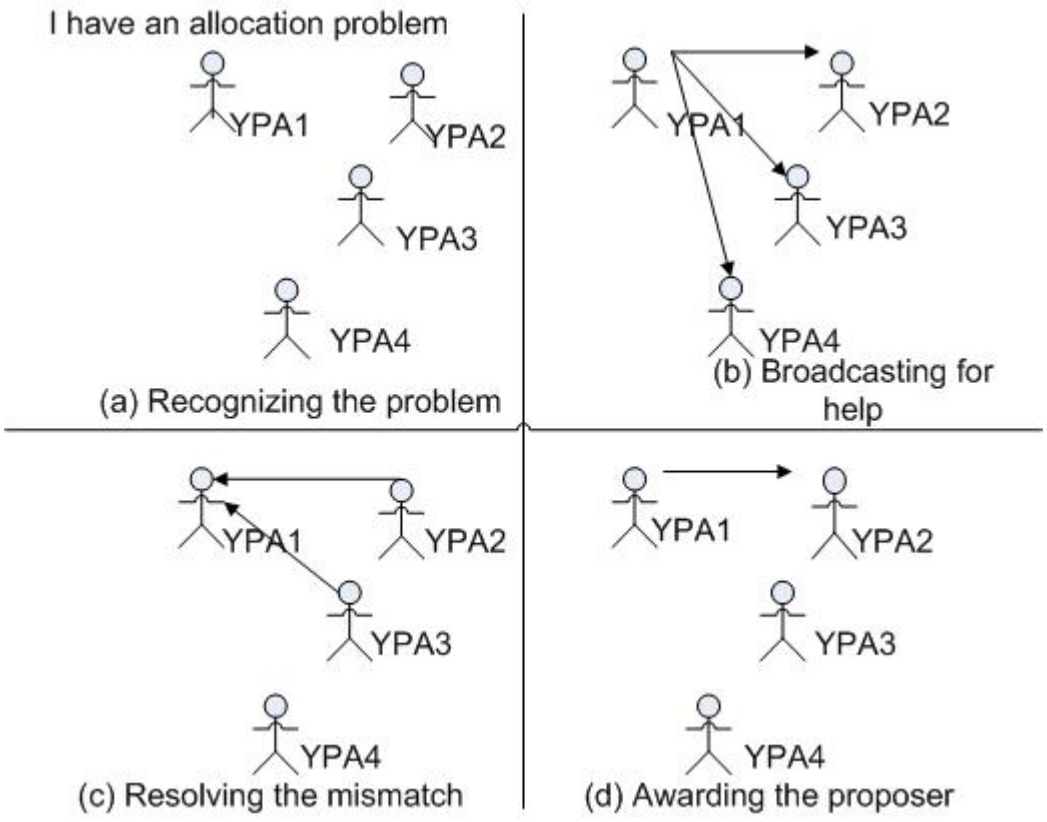

Figure 5.9: Extended Contract Net Protocol

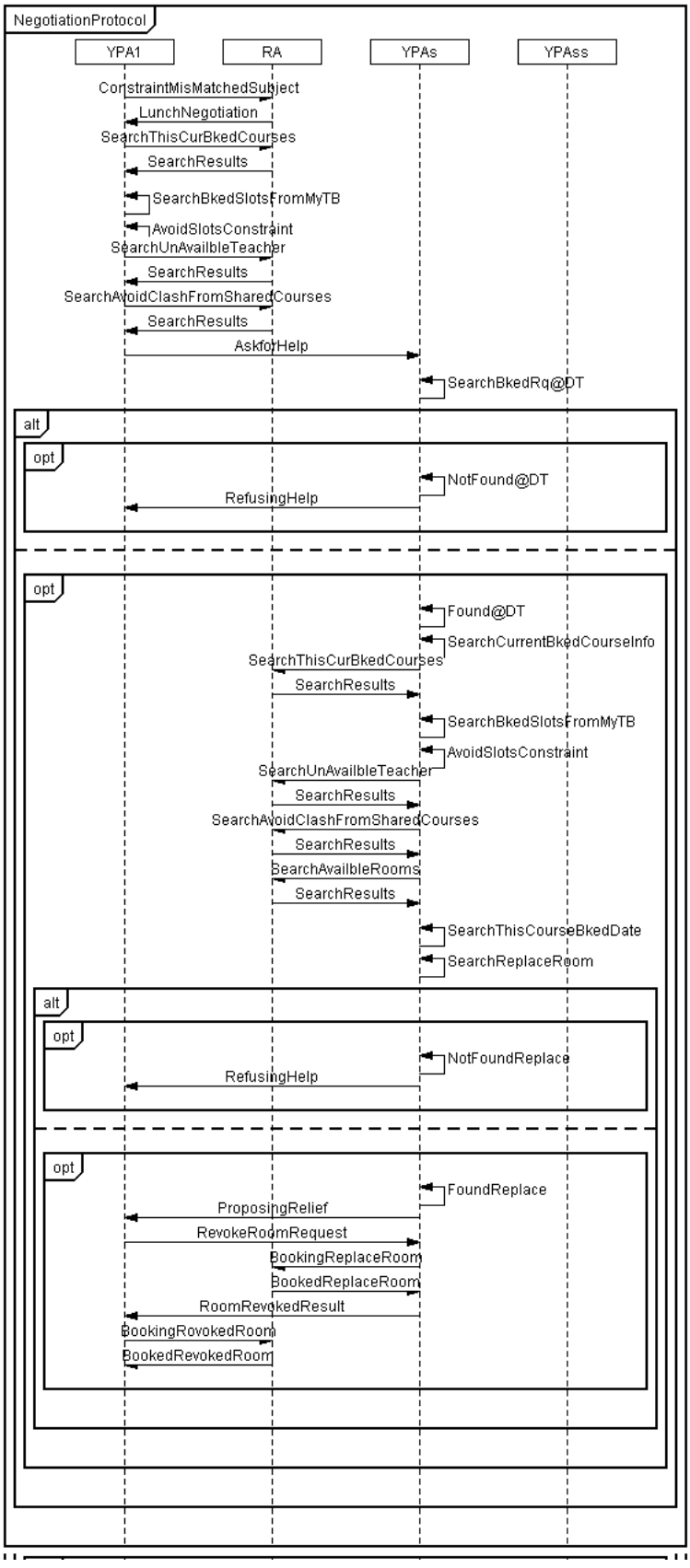

Figure 5.10: Interaction between *YPA* & *YPAs* and *RA* in the negotiation phase

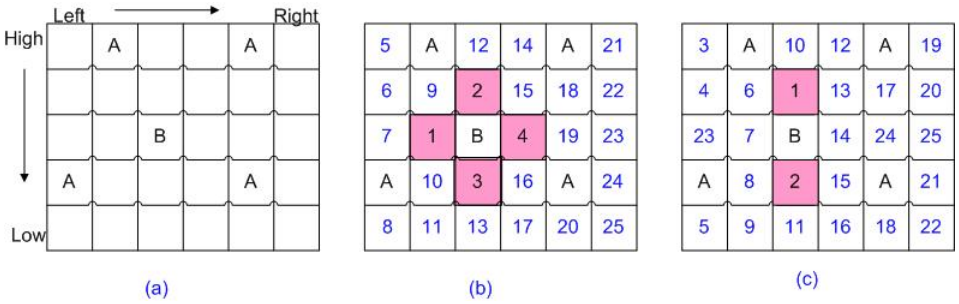

Figure 5.11: (a) Virtual table with avoided and booked slots (b) 1-25 prioritized slots when the organizing lecture has not been booked (c) 1-25 prioritized slots when the organizing lecture has been booked

"A" = the necessary avoiding slots; "B" = the booked slot;
1 = the highest priority; 25 = the lowest priority

Chapter 6

Benchmark Data Sets

This chapter describes the datasets which have been defined for the *International Timetabling Competition*; these have been adapted as a benchmark for this work. The means of validating the results and a set of reference results are also provided.

6.1 The International Timetabling Competition (ITC)

For this research we have employed as a benchmark the set of problem formula and data instances which were used in the *Curriculum-based Course Timetabling* Competition (Gaspero et al., 2007). It is one of the three tracks in *the Second International Timetabling Competition (ITC-2007)*, for which the other two competitions are *Examination Timetabling* and *Post Enrolment Timetabling*. The aim of this competition was to bridge the gap between research and practice by introducing a significant degree of complexity in the tracks, so that the employed formulations are closer to the real world needs (not in all aspects, but a degree of generality is kept) and the datasets are also taken from the real world. Under the *Curriculum-based Course Timetabling* Competition, all instances have been selected from a large set of interesting cases from a real-world institute—the University of Udine in Italy. More competition information can be obtained at http://www.cs.qub.ac.uk/itc2007 and http://tabu.diegm.uniud.it/ctt. The following sections provide brief details of the *Curriculum based Course Timetabling* Competition.

6.2 Problem Definition

The curriculum-based timetabling involves scheduling a set of lectures for several university courses to a given number of rooms, time periods nomally on a weekday. The conflicts among courses are provided by the curricula published by the University and not the enrolment data. The problem is specified using the following entities:

- *Days*, *Timeslots* and *Periods*: *Days* means the number of teaching days in a week; *Timeslots* means a number of timeslots available in each day; A *Period* means a pair of a day and a timeslot. From the definition, the total number of scheduling periods is the products of days times the day timeslots. Note that day and timeslot numbers are defined to start from 0.
- *Courses and Teachers*: Each course consists of a fixed number of *lectures*, *a number of students* who attain in that course, *the name of teacher* who is responsible to teach in that course and *the minimum number of days* that the lectures should be spread across in the week.
- *Rooms*: Each room is described in terms of *name* and *capacity* (the number of available seats); all rooms have the same properties other than different capacities.
- *Curricula*: A curriculum consists of a set of courses. The conflicts between courses among curricula are set.

As order to address this problem definition, we need to assign responsibilities to YPAs, by defining each YPA to organize a set of courses under a specific curriculum.

6.2.1 Constraints

The *essential* and *desirable* constraints which are defined for this problem formulation are presented in Table 6.1; and the *costs of violation* and *penalty values* are given in following section:

6.2.2 Violation and Penalty Values

Violation and *penalty values* are necessary, since they are needed for weighted-sum single-objective function. For this competition, multi-objective formulations and Pareto optimality issues are out of scope. The cost component *UD1(Udine1)* is used for this research.

Essential Constraints	***Desirable Constraints***
• ***Lectures***: All lectures must be scheduled and assigned to distinct periods. • ***RoomOccupancy***: Two lectures cannot take place in the same room at the same time. • ***Conflicts***: Lectures belonging to the same curriculum or taught by the same teacher must not conflict. • ***Availabilities***: the teacher must be available when lecture is scheduled.	• ***RoomCapacity***: the assigned room must be able to contain the number of students which attend that course. • ***MinimumWorkingDays***: the number of lectures of each course must reach the minimum number of days. • ***CurriculumCompactness***:whole lectures belonging to the same programme (called a curriculum) should be adjacent.

Table 6.1: The defined *essential & desirable* constraints (Gaspero et al., 2007)

- *Lectures constraint*: a violation will be recorded if a lecture is not scheduled.
- *RoomOccupancy constraint*: a violation will be recorded and be counted as one if an extra lecture is scheduled in the same period and room.
- *Conflicts constraint*: a violation will be recorded if there are any two conflicting lectures in the same period. If there are four conflicting lectures, then six violations are counted.
- *Availabilities constraint*: a violation will be recorded if a lecture is scheduled in a period for which the course teacher is unavailable.
- *RoomCapacity constraint*: one penalty point will be counted for each student above the room capacity.
- *MinimumWorkingDays constraint*: five penalty points will be counted for each day below the minimum working days requirement.
- *CurriculumCompactness constraint*: one penalty point will be counted if for a given curriculum, there is one lecture that is not adjacent to any other lecture within the same day.

6.3 Instances and File Formats

6.3.1 Input File Format

An instance contains in one text file which is able to divide into four sections: *course* information, *room* information, *curricula* information and *constraint* information, as details which is presented in following Figure 6.1.

The data in each section is always provided in a precise order and can be interpreted as following:

- In the *courses* section, each line contains information about that course in one line, which is comprised of course name, teacher who teaches on that course, the number of lectures per week, the minimum number working days for those lectures, and the number of students who are taking that course.

Name: Instance_name
Courses: Number of courses
Rooms: Number of rooms
Days: Number of days
Periods per day: Number of periods
Curricula: Number of curricula
Constraints: Number of constraints
COURSES:
<CourseID> <Teacher> <#Lectures> <MinWorkingDays>
<#Students>
<CourseID> <Teacher> <#Lectures> <MinWorkingDays>
<#Students>
<CourseID> <Teacher> <#Lectures> <MinWorkingDays>
<#Students>
...
ROOMS:
<RoomID> <Capacity>
<RoomID> <Capacity>
<RoomID> <Capacity>
...
CURRICULA:
<CurriculumID> <#Courses> <CourseID>... <CourseID>
<CurriculumID> <#Courses> <CourseID>... <CourseID>
<CurriculumID> <#Courses> <CourseID>... <CourseID>
...
UNAVAILABILITY_CONSTRAINTS:
<CourseID> <Day> <Time>
<CourseID> <Day> <Time>
<CourseID> <Day> <Time>
...

Figure 6.1: Input file format (Gaspero et al., 2007)

<CourseID> <RoomID> <Day> <Time>
<CourseID> <RoomID> <Day> <Time>
<CourseID> <RoomID> <Day> <Time>
...

Figure 6.2: Output file format (Gaspero et al., 2007)

- In the *rooms* section, each line maintains information about that room in one line, which is comprised of room name, and room capacity.
- In the *curricula* section, each line provides information about that curriculum in one line, which is comprised of curriculum name, the number of courses in that curriculum, and a list of names of course that belongs to the same curriculum.
- In the *unavailability_ constraints* section, there is information about course names and the day-time slots which these courses need to be avoided.

6.3.2 Output File Format

The result of the timetabling process must be provided in a single text file, for which each line is composed of *course id*, *room id*, and *day-time slot* as presented in Figure 6.2.

6.4 Benchmarking

6.4.1 Instances

For this competition, there are 21 instances of timetable requirements that have been specified, called Comp01...Comp21. All instances are downloadable from http://tabu.diegm.uniud.it/ctt. All of them have been taken form real cases arising in the *University of Udine*. The main features of these instances are summarized in Table 6.2.

Table 6.2 shows the following features for each instance, comprising the number of courses (C), total lectures (L), rooms (R), periods per day (PpD), days (D), curricula (Cu), min and max lectures per day per curriculum (MML), average number of conflicts (Co), Average teacher availability (TA), average number of lectures per curriculum per day (CL), average room occupation (RO).

instance	C	L	R	PpD	D	Cu	MML	Co	TA	CL	RO
comp01	30	160	6	6	5	14	2-5	13.2	93.1	3.24	88.9
comp02	82	283	16	5	5	70	2-4	7.97	76.9	2.62	70.8
comp03	72	251	16	5	5	68	2-4	8.17	78.4	2.36	62.8
comp04	79	286	18	5	5	57	2-4	5.42	81.9	2.05	63.6
comp05	54	152	9	6	6	139	2-4	21.7	59.6	1.8	46.9
comp06	108	361	18	5	5	70	2-4	5.24	78.3	2.42	80.2
comp07	131	434	20	5	5	77	2-4	4.48	80.8	2.51	86.8
comp08	86	324	18	5	5	61	2-4	4.52	81.7	2	72.0
comp09	76	279	18	5	5	75	2-4	6.64	81	2.11	62.0
comp10	115	370	18	5	5	67	2-4	5.3	77.4	2.54	82.2
comp11	30	162	5	9	5	13	2-6	13.8	94.2	3.94	72.0
comp12	88	218	11	6	6	150	2-4	13.9	57	1.74	55.1
comp13	82	308	19	5	5	66	2-3	5.16	79.6	2.01	64.8
comp14	85	275	17	5	5	60	2-4	6.87	75	2.34	64.7
comp15	72	251	16	5	5	68	2-4	8.17	78.4	2.36	62.8
comp16	108	366	20	5	5	71	2-4	5.12	81.5	2.39	73.2
comp17	99	339	17	5	5	70	2-4	5.49	79.2	2.33	79.8
comp18	47	138	9	6	6	52	2-3	13.3	64.6	1.53	42.6
comp19	74	277	16	5	5	66	2-4	7.45	76.4	2.42	69.2
comp20	121	390	19	5	5	78	2-4	5.06	78.7	2.5	82.1
comp21	94	327	18	5	5	78	2-4	6.09	82.4	2.25	72.7

Table 6.2: Main features of instances (Gaspero et al., 2007)

The average number of conflicts (Co) is the result of dividing the total number of the pair of lectures that cannot be scheduled at the same time by the total number of distinct pairs of lectures. The values of TA, CL, and RO are calculated in the same way.

6.4.2 Validators

To help encourage others to use the problem formula and instance datasets, with the aim of having these become a standard within the course timetabling research field, a web based validator has been set up. It allows the user to upload a solution file, select an existing instance and then obtain a validation report. Figure 6.3 presents the validator and the location of the validating link is "http://tabu.diegm.uniud.it/ctt/index.php?page=valid".

Figure 6.3: Validate solution website

6.4.3 Reference Results

The reference solutions have been achieved by several competitors from different techniques i.e. *tabu search*, *mathematical programming* and *other technique*; running with a 360–second timeout on PC and taking the best out of 40 repetitions. Table 6.3 presents the reference results, which all are feasible solutions, and the best soft penalty values for these instances.

6.5 Summary

The main contents of this chapter provide information about the I*nternational Timetabling Competition (ITC)*. We particularly focus on the third track information—*Curriculum-Based Course Timetabling*. The details of problem definition, data instances and input-output files format have been described. The last section of this chapter describes the means of validating the timetabling results, and a set of reference results is also provided.

instance	Best Results		Technique	Author
	Hard Penalty	Soft Penalty		
comp01	0	4	Tabu Search	Andrea Schaerf
comp02	0	12	Other	S. Abdullah and H. Turabieh
comp03	0	38	Tabu Search	Lu and Hao
comp04	0	18	Tabu Search	Andrea Schaerf
comp05	0	219	Tabu Search	Lu and Hao
comp06	0	14	Other	S. Abdullah and H. Turabieh
comp07	0	3	Mathematical Programming	Gerald Lach
comp08	0	20	Mathematical Programming	Gerald Lach
comp09	0	54	Tabu Search	Lu and Hao
comp10	0	2	Mathematical Programming	Gerald Lach
comp11	0	0	Tabu Search	Andrea Schaerf
comp12	0	239	Tabu Search	Lu and Hao
comp13	0	32	Tabu Search	Lu and Hao
comp14	0	27	Tabu Search	Lu and Hao
comp15	0	38	Tabu Search	Lu and Hao
comp16	0	11	Other	S. Abdullah and H. Turabieh
comp17	0	30	Other	S. Abdullah and H. Turabieh
comp18	0	34	Tabu Search	Lu and Hao
comp19	0	32	Tabu Search	Lu and Hao
comp20	0	2	Other	S. Abdullah and H. Turabieh
comp21	0	34	Other	S. Abdullah and H. Turabieh

Table 6.3: Reference results

Chapter 7

Experimental Results

This chapter presents the results from the experiments. It starts by reviewing the research questions and then identifies how the different principles can be investigated. The research method is organized as a set of experiments to address each defined hypothesis; collecting the experimental results; and finally concluding the experiment by verifying, falsifying or establishing the validity of that hypothesis.

7.1 Introduction

In this research we aim to investigate the following two research questions:

1. How to apply agent-based technology to allocate the university resources so as to satisfy the essential constraints for serving the teaching activities and to address as many as possible of the desirable constraints that correspond to real world needs?

2. How to ensure that each agent gets a fair chance to acquire the resources that it needs?

To do so, we need to analyse the results which came from using the *First-In-First-Out & Sequential (FIFOSeq)*, *First-In-First-Out & Interleaved (FIFOInt)*, *Round-Robin & Sequential (RRSeq)* and *Round-Robin & Interleaved (RRInt)* prnciples. We do this by comparing the timetabling results in terms of how well the constraints were addressed and the degree of fairness achieved when allocating rooms to *YPAs*. In this research, we address the above questions by restructuring them as a set of hypotheses, then using an experimental design in order to test each causal relationship. All experiments under this

study are run on laptop computer—Intel Core 2 Duo T7100 @ 1.8GHz 1.8GHz Ram 1 GB 32 bit Windows Vista.

7.2 The Hypotheses

In order to address the issues and questions identified in section 4.4, the follwing ten hypotheses have been identified:

H1) All four proposed multiagent principles are able to achieve the university course timetabling formula and constraints which were defined for the *International Timetabling Competition 2007*.

H2) The number of successfully organized lectures has been increased after applying the *negotiation phase* following the *initial allocation phase* of both *FIFO* and *RR* algorithms.

H3) Different initial sequences of *YPAs* will produce different results.

H4) Among the four models, the *First-In-First-Out & Sequential* principle will always give the smallest penalty value when compared with the other principles.

H5) Among the four principles, the *First-In-First-Out & Interleaved* principle will always give the largest penalty value when compared with the other principles.

H6) Among the four principles, the *Round-Robin & Interleaved* principle will always give the smallest standard deviation of satisfied values when compared with the other principles.

H7) Among the four principles, the *First-In-First-Out & Sequential* principle will always give the largest standard deviation of satisfied values when compared with the other principles.

H8) Increasing the number of agents (curricula) gives larger penalty values. (An agent takes a responsibility to organize a set of courses in a curriculum.)

H9) The proposed principles generally produce smaller penalty values when compared with the reference results.

H10) The *Round Robin & Interleaved* principle is most likely to produce better solutions in terms of both smallest penalty value and best degree of fairness when comparing to the other principles.

7.3 Testing the Hypotheses

H1) All four proposed multiagent principles are able to achieve the university course timetabling formula and constraints which were defined for the *International Timetabling Competition 2007*.
In order to test hypothesis *H1*, the experimental design uses the following

- *independent variables*: instance datasets and the initial *YPAs* sequences of each dataset.
- *dependent variable*s: integer counts for the number of *success(S)* or *fail(F)* results (range 0-15)

Five instances have been randomly selected to be samples of this experiment. Each sample is run fifteen times through the four different models, as this amount is sufficient to produce a conclusion. For each time of running, an initial *YPAs* sequence set is randomly chosen and four sets of organized timetables are returned for success; or treat as fail if models cannot produce the timetables. The experimental process is summarised below.

0. set instanceCount = 0.
1. randomly select an instance X; and set instanceCount = instanceCount +1
2. randomly select a set of initial sequence YPAs Y; and set trial_num = 0.
3. four treatments have been established for the four different principles and use the set of initial sequence YPAs Y as the input for each of these four treatments; the number of experiment is counted: trial_num = trial_num +1
4. four sets of organized timetables are returned for success; or treat as fail if the treatments cannot produce the timetables
5. information about the instance, the initial sequence of YPAs Y and the results of this experiment is recorded
6. randomly select a new set of initial YPAs Y and go to step 2, if trial_num≤ 15; otherwise go to step 1, if instanceCount ≤ 5; else experiment is terminatd

After the experiment is terminated, the number of successes(S) / fails(F) will be counted for each treatment and presented in the form of a percentage of achievements

- If percentage of achievements is 100, then hypothesis *H1* is supported.
- If percentage of achievements is 0, then hypothesis *H1* is not supported.
- If the percentage of achievements is between 0 and 100, then hypothesis *H1* will be accepted at the validity value which is equal to the value of the percentage of that achievement.

In this *H1* hypothesis test, the chosen instances are *Comp04*, *Comp11*, *Comp13*, *Comp15* and *Comp18* in *Appendix D* . The information of initial *YPAs* sequences is in *Appendix A*. The organized timetable results are in *Appendix B*. The information about the number of successes(S) or fails(F) for this experiment is presented in Table 7.1

Information in Table 7.1 and the conditions which are designed for this experiment are indicated that hypothesis *H1* is supported. The four proposed multiagent principles are able to achieve the university course timetabling formula and constraints which were defined for the *International Timetabling Competition 2007*.

instance	FIFOSeq			RRSeq			FIFOInt			RRInt		
	S	**F**	**%**	**S**	**F**	**%**	**S**	**F**	**%**	**S**	**F**	**%**
Comp04	15	0	100	15	0	100	15	0	100	15	0	100
Comp11	15	0	100	15	0	100	15	0	100	15	0	100
Comp13	15	0	100	15	0	100	15	0	100	15	0	100
Comp15	15	0	100	15	0	100	15	0	100	15	0	100
Comp18	15	0	100	15	0	100	15	0	100	15	0	100

Table 7.1: Numbers of successes or fails of *H1* hypothesis test

H2) The number of successfully organized lectures has been increased after applying the *negotiation phase* following the *initial allocation phase* of both *FIFO* and *RR* algorithms. In order to test hypothesis *H2*, the experimental design uses the following

- *independent variables*: instance datasets and the initial *YPAs* sequences of each dataset.
- *dependent variable*: change occurring in the total number of organized lectures after the initial allocation phase and the total number of organized lectures after the negotiation phase.

Five instances have been randomly selected to be samples of this experiment. Each sample is run fifteen times through the four different treatments, as this amount is sufficient to produce a conclusion. For each time of running, an initial *YPAs* sequence set is randomly chosen and four sets of organized timetables are returned. The experimental process is summarised below.

0. set instanceCount = 0
1. randomly select an instance X; and set instanceCount = instanceCount +1
2. randomly select a set of initial sequence YPAs Y; and set trial_num = 0.
3. four treatments have been established for the first-in-first-out, first-in-first-out & negotiation; and round robin, round robin & negotiation by applying the set of initial sequence YPAs Y as the input for each of these four treatments; the number of the experiment is counted: trial_num = trial_num +1
4. four sets of organized timetables are returned
5. information about the instance, the initial sequence of YPAs Y and the results of these four treatments are recorded
6. randomly select a new set of initial YPAs Y and go to step 2, if trial_num $\leq$ 15; otherwise go to step 1, if instanceCount $\leq$ 5; else experiment is terminatd

After the experiment is terminated, the total number of organized lectures of each instance from the four treatments is counted and is compared between treatments which adding up the negotiation part and the one which has not been added.
If the number of organized lectures of the treatment which added negotiation part is greater than the one which has not been added, then we can conclude that negotiation is able to increase the number of organized lectures; as some *YPAs* in system are glad to switch rooms and hypothesis *H2* is supported. Otherwise, If the number of organized lectures of the treatment which added negotiation part is equal the one which has not been added, then we can conclude that negotiation is able to increase the number of organized lectures, but some *YPAs* in system can not help to switch rooms and hypothesis *H2* is also supported; otherwise, the hypothesis *H2* will not be supported.
In this *H2* hypothesis test, the chosen instances are *Comp04, Comp11, Comp13, Comp15* and *Comp18* in *Appendix D*. The information of initial *YPAs* sequences is in *Appendix A*. The organized timetables from four treatments are in *Appendix B*. Information about the number of organized lectures from *First-In-First-Out, First-In-First-Out & Negotiation, Round Robin, Round Robin & Negotiation* is shown in Table 7.2. We expect that negotiation phase is able to solve constraint-mismatched resource allocation problem.

instance	Total Lectures	First-In-First-Out Organized Lectures			Round Robin Organized Lectures		
		FIFO	FIFO &Nego	change	RR	RR &Nego	change
Comp04	4290	4222	4239	17	4267	4270	3
Comp11	2430	2360	2430	70	2367	2430	63
Comp13	4620	4603	4607	4	4619	4619	0
Comp15	3765	3694	3703	9	3683	3697	14
Comp18	2070	2070	2070	0	2066	2070	4

Table 7.2: Numbers of organized lectures before & after including the negotiation part

The information in Table 7.2 shows that hypothesis *H2* is supported. The number of organized lectures has been increased in the majority of cases, after applying the *negotiation phase* following the *initial allocation phase* of both *FIFO* and *RR* algorithms.
H3) Different initial sequences of *YPAs* will produce different results.
In order to test hypothesis *H3*, the experimental design uses the following

- *independent variables*: instance datasets and the two different initial *YPAs* sequences of each dataset.
- *dependent variables*: integer counts for the number of boolean value indicating end results is *identical (I)* or *different(D)* (range 0-15)

Five instances have been randomly selected to be samples of this experiment. Each sample is run fifteen times using each of the four different principles, as this amount is sufficient to produce a conclusion. For each time of running, two initial *YPAs* sequence sets are randomly chosen and eight sets of organized timetables are returned. The experimental process is summarised below.

0. set instanceCount = 0
1. randomly select an instance X; and set instanceCount = instanceCount +1
2. randomly select two different sets of initial sequence YPAs Y1 & Y2; and set trial_num = 0.
3. four treatments have been established for the four different principles and use the two sets of initial sequences YPAs Y1 & Y2 as the input for each of these four treatments; the number of the experiment is counted: trial_num = trial_num +1
4. eight sets of organized timetables are returned
5. any two organized timetables which are products of the same treatment are compared and judged; the result will be identical (I)if those two timetables are the same; otherwise the result will be different (D)
6. information about the instance X, the initial sequences of YPAs Y1&Y2 and the results of this experiment is recorded
7. randomly select new sets of initial sequences YPAs Y1 & Y2 and go to step 2, if trial_num ≤ 15; otherwise go to step 1, if instanceCount ≤ 5; else experiment is terminated .

After the experiment is terminated, the number of identical(I)/different (D) results will be counted for each treatment and presented in the form of a percentage of different results

- If percentage of different results is 100, then hypothesis *H3* is supported;
- If percentage of different results is 0, then hypothesis *H3* is not supported;
- If the percentage of different results is between 0 and 100, then hypothesis *H3* will be accepted at the validity value which is equal to the value of the percentage of different results.

In this *H3* hypothesis test, the chosen instances are *Comp04, Comp11, Comp13, Comp15* and *Comp18* in *Appendix D*. The information of initial *YPAs* Y1&Y2 sequences is in *Appendix A*. The organized timetable results are in *Appendix B*. The information about the number of identical (I) or different (D) results of this experiment is presented in Table 7.3

instance	FIFOSeq			RRSeq			FIFOInt			RRInt		
	I	D	%	I	D	%	I	D	%	I	D	%
Comp04	0	15	100	0	15	100	0	15	100	0	15	100
Comp11	0	15	100	0	15	100	0	15	100	0	15	100
Comp13	0	15	100	0	15	100	0	15	100	0	15	100
Comp15	0	15	100	0	15	100	0	15	100	0	15	100
Comp18	0	15	100	0	15	100	0	15	100	0	15	100

Table 7.3: Numbers of identical or different results of two different initial *YPAs* sets

Information in Table 7.3 and the conditions under which this experiment occur indicate that hypothesis *H3* is supported. Different initial sequences of *YPAs* produce different results.

H4) Among the four principles, the *First-In-First-Out & Sequential* principle will always give the smallest penalty value when compared with the other principles.

H5) Among the four principles, the *First-In-First-Out & Interleaved* principle will always give the largest penalty value when compared with the other principles.

H6) Among the four principles, the *Round-Robin & Interleaved* principle will always give the smallest standard deviation of satisfied values when compared with the other principles.

H7) Among the four principles, the *First-In-First-Out & Sequential* principle will always give the largest standard deviation of satisfied values when compared with the other principles.

H8) Increasing the number of agents (curricula) gives larger penalty values.

H9) The proposed principles generally produce smaller penalty values when compared with the *ITC-2007* reference results.

H10) The *Round Robin & Interleaved* principle is most likely to produce better solutions in terms of both lowest penalty value and best degree of fairness when comparing to the other principles.

In order to test hypotheses *H4-H10*, the experimental design uses the following

- *independent variables*: instance datasets and the initial *YPAs* sequence of each dataset.
- *dependent variables*: successfully organized timetables, timetable validation results, sets of satisfied values, standard deviation of the satisfied values.

Five instances have been randomly selected to be samples of this experiment. Each sample is run fifteen times using the four different principles, as this amount is sufficient to produce a conclusion. For each time of running, an initial *YPAs* sequence set is randomly chosen and four sets of organized timetables are returned. The experimantal process is summarised below.

0. set instanceCount = 0
1. randomly select an instance X; and set instanceCount = instanceCount +1
2. randomly select a set of initial sequence YPAs Y; and set trial_num = 0.
3. four treatments have been established for the four different principles and use the set of initial sequence YPAs Y as the input for each of these four treatments; the number of the experiment is counted: trial_num = trial_num +1
4. four sets of organized timetables are returned
5. information about the instance X, the initial sequence of YPAs Y, organized timetables, timetable validating results, sets of satisfied values, standard deviation of the satisfied values are calculated and recorded
6. judge the sequence of YPAs Y which gives the smallest/the largest penalty value, the smallest/the largest of the standard deviation of the satisfied values; then record the results.
7. randomly select a new set of initial YPAs Y and go to step 2, if trial_num $\leq$ 15; otherwise go to step 1, if instanceCount $\leq$ 5; else experiment is terminated.

After the experiment being terminated, the results of the smallest/ the largest penalty values, the smallest/the largest standard deviation of the satisfied values of each instance will be reported.

- For all instances, if the *First-In-First-Out & Sequential* principle gives the smallest penalty value of organized timetable results when compared with the others, then hypothesis *H4* will be supported; otherwise, hypothesis *H4* will not be supported.
- For all instances, if the *First-In-First-Out & Interleaved* principle gives the largest penalty value of organized timetable results when compared with the others, then hypothesis *H5* will be supported; otherwise, hypothesis *H5* will not be supported.

- For all instance, if the *Round-Robin & Interleaved* principle gives the smallest standard deviation of the satisfied values when compared with the others, then hypothesis *H6* will be supported; otherwise, hypothesis *H6* will not be supported.

- For all instance, if the *First-In-First-Out & Sequential* principle gives the largest standard deviation of the satisfied values when compared with the others, then hypothesis *H7* will be supported; otherwise, hypothesis *H7* will not be supported.

In these hypotheses *H4-H7* test, the random instances are *Comp04*, *Comp11*, *Comp13*, *Comp15* and *Comp18* in *Appendix D*. The information of randomly initial *YPAs* sequences is in *Appendix A*. The organized timetable results are in *Appendix B*. The timetable validating results (penalty values) and the standard deviation of satisfied values are in *Appendix C*.

A penalty function has been defined for the purpose of comparing the outcomes from the different principles.

The first part counts the number of violations of the essential constrants, while the second part (fractional) part counts the number of violations of the desirable constraints.

$$\text{penalty(E,D)} = \text{E} + 0.001*\text{D}$$

when E represents the number of violations of essential constraints.

D represents the number of violations of desirable constraints.

As we found that the number of returned penalty values of the desirable constraints from all experiments are fewer than 1000; we therefore took 0.001 to use as the weighting for the number of violations of the desirable constraints.

Instance Comp04

The penalty values of instance *Comp04* from this experiment are presented by ranging from low to high values ordered by results of the *FIFOSeq* principle in Table 7.4 and the graph is plotted as shown in Figure 7.1.

Initial	FIFOSeq	RRSeq	FIFOInt	RRInt
Seq02	1.160	0.144	1.141	0.140
Seq15	2.140	2.153	1.155	2.153
Seq08	2.148	1.138	2.121	1.138
Seq06	2.156	3.168	2.141	2.168
Seq05	2.173	0.145	2.145	0.152
Seq13	3.154	2.171	3.129	2.171
Seq11	3.178	2.153	2.142	2.153
Seq10	3.195	2.166	3.164	2.166
Seq03	4.138	0.160	4.110	0.160
Seq01	4.170	2.155	4.147	2.155
Seq12	4.181	1.154	4.147	1.157
Seq07	4.193	1.155	2.143	1.155
Seq09	5.148	1.147	5.131	1.147
Seq14	5.149	1.152	5.119	1.152
Seq04	7.178	2.167	5.135	2.167

Table 7.4: Penalty values for instance Comp04

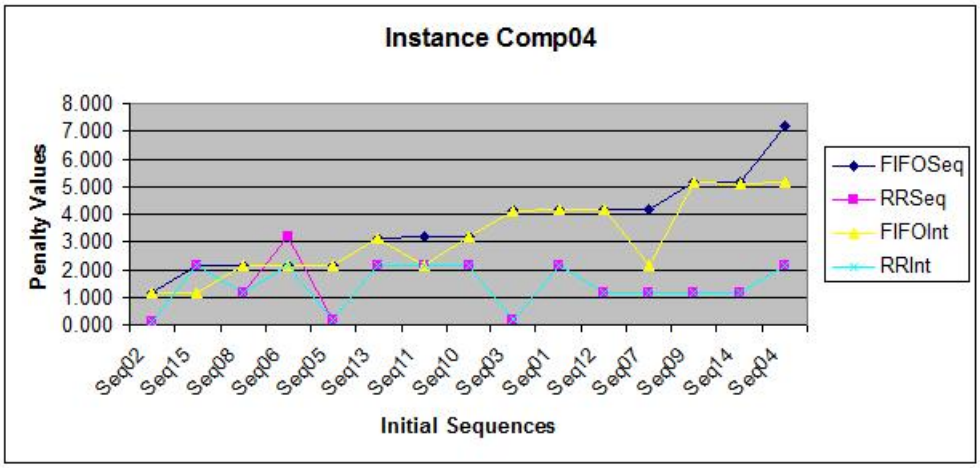

Figure 7.1: Penalty values for instance Comp04

For this instance, the best solution of this experiment is an outcome of the *RRInt* principle which it is a result of the 2nd initial sequence.
The standard deviation of the satisfied values of instance *Comp04* from this experiment is presented in Table 7.5 and is plotted in graph as shown in Figure 7.2. The sorted standard deviation values from low to high of each principle are plotted in graph as shown in Figure 7.3.

Initial	**FIFOSeq**	**RRSeq**	**FIFOInt**	**RRInt**
Seq02	14.06694	16.08371	14.22331	16.35595
Seq15	15.59036	13.22198	16.29037	13.22198
Seq08	14.45919	11.86092	14.66142	11.86092
Seq06	13.69332	13.12047	13.69332	13.12047
Seq05	14.28662	14.17115	14.28662	13.64701
Seq13	15.52213	14.09711	15.52213	14.09711
Seq11	13.92347	12.6507	13.14233	12.6507
Seq10	13.75726	14.61066	13.75726	14.61066
Seq03	15.54935	13.28079	15.54935	13.28079
Seq01	15.29507	13.76547	15.28273	13.76547
Seq12	14.1297	11.53095	14.2973	12.15016
Seq07	15.37339	13.77067	15.40687	13.60648
Seq09	13.08457	14.48969	13.08457	14.48969
Seq14	14.79538	14.55757	14.79538	14.49087
Seq04	15.79531	13.17073	15.64867	13.17073

Table 7.5: Standard deviation values for instance Comp04

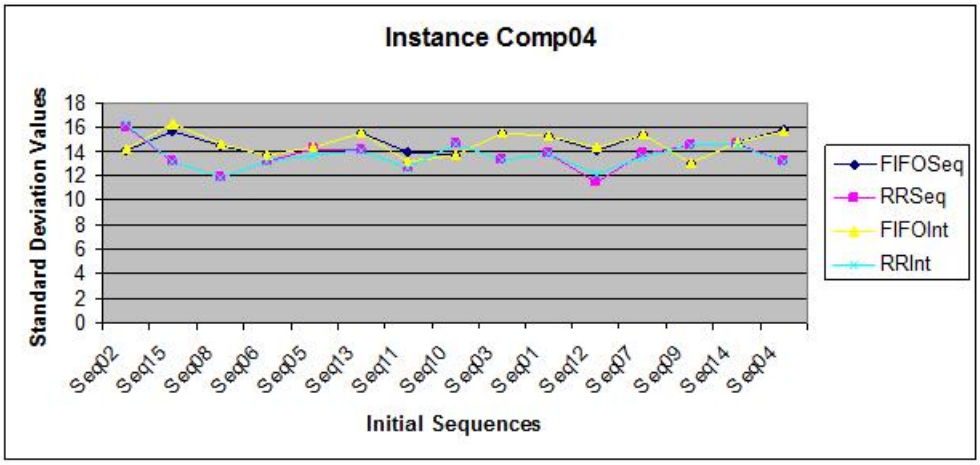

Figure 7.2: Standard deviation values for instance Comp04

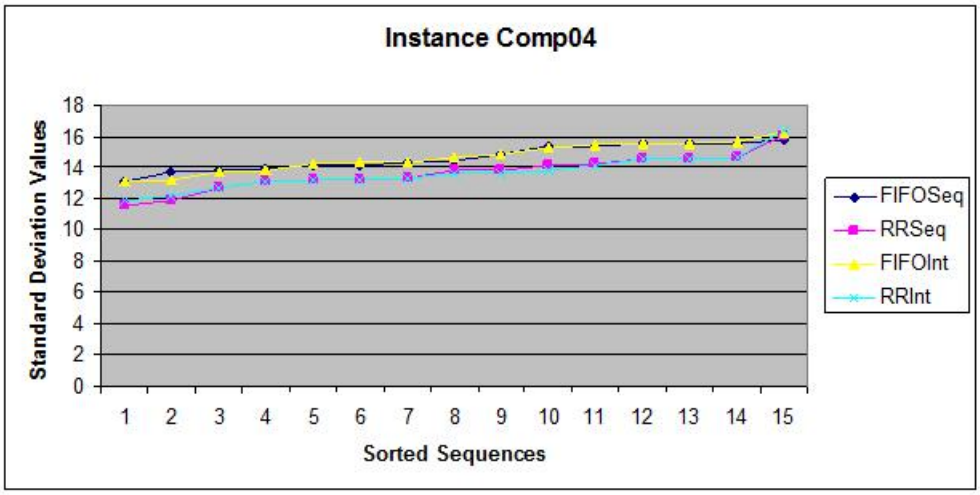

Figure 7.3: Sorted standard deviation values for instance Comp04

For this instance, the smallest standard deviation of satisfied values of resources of this experiment is an outcome of the *RRSeq* principle which it is a result of the 12nd initial sequence. The sequence which gives the smallest penalty value and the smallest standard deviation of satisfied value of resources is the 8th initial sequence. The results are products of the *RRSeq* and *RRInt* principles.

Instance Comp11

The penalty values of instance *Comp11* from this experiment are presented by ranging from low to high values ordered by results of the *FIFOSeq* principle in Table 7.6 and the graph is plotted as shown in Figure 7.4.

Initial	FIFOSeq	RRSeq	FIFOInt	RRInt
Seq02	0.007	0.053	0.007	0.052
Seq14	0.007	0.047	0.007	0.047
Seq06	0.012	0.033	0.012	0.031
Seq07	0.013	0.052	0.013	0.051
Seq11	0.014	0.052	0.014	0.059
Seq12	0.016	0.040	0.014	0.042
Seq04	0.017	0.028	0.017	0.028
Seq08	0.017	0.033	0.017	0.033
Seq13	0.018	0.051	0.018	0.050
Seq03	0.020	0.023	0.020	0.022
Seq05	0.021	0.043	0.021	0.043
Seq09	0.028	0.045	0.028	0.057
Seq15	0.040	0.054	4.033	0.054
Seq01	0.041	0.041	3.045	0.043
Seq10	0.055	0.035	0.053	0.033

Table 7.6: Penalty values for instance Comp11

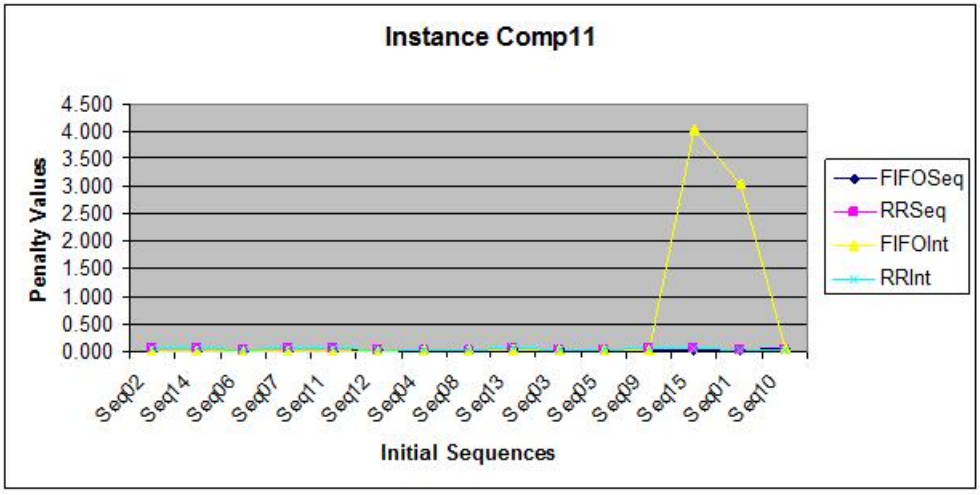

Figure 7.4: Penalty values for instance Comp11

For this instance, the best solution of this experiment is an outcome of the *FIFOSeq* and *FIFOInt* principles which it is a result of the 2nd and the 14th initial sequences. The standard deviation of the satisfied values of instance *Comp11* from this experiment is presented in Table 7.7 and is plotted in graph as shown in Figure 7.5. The sorted standard deviation values from low to high of each principle are plotted in graph as shown in Figure 7.6.

Initial	**FIFOSeq**	**RRSeq**	**FIFOInt**	**RRInt**
Seq02	9.40911	13.4968	9.40911	13.3909
Seq14	15.4233	15.7263	15.4233	15.7263
Seq06	14.4968	13.9205	14.4557	13.9205
Seq07	13.2065	12.2871	13.2065	12.2871
Seq11	15.0629	13.4878	15.0629	13.5298
Seq12	17.3104	14.0879	17.3104	14.0879
Seq04	12.4348	14.6947	12.4348	14.6947
Seq08	13.5644	11.7611	13.5644	12.1546
Seq13	13.7477	15.1386	13.7477	15.1386
Seq03	11.8713	11.544	11.8713	11.4799
Seq05	13.3215	13.5529	13.3215	12.3034
Seq09	12.3106	11.9539	12.3106	13.8873
Seq15	12.9499	11.6611	12.9259	11.7612
Seq01	11.542	13.7333	10.5142	13.5487
Seq10	16.9415	14.3182	17.0095	14.3587

Table 7.7: Standard deviation values for instance Comp11

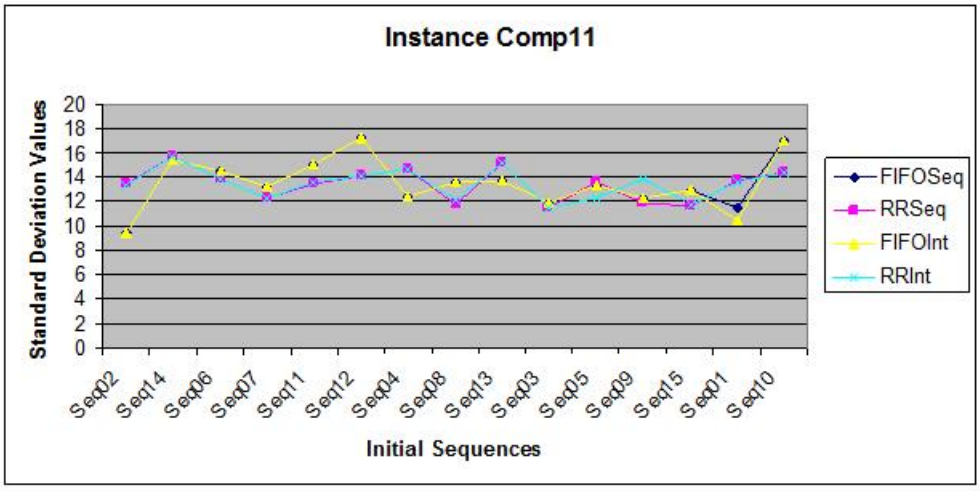

Figure 7.5: Standard deviation values for instance Comp11

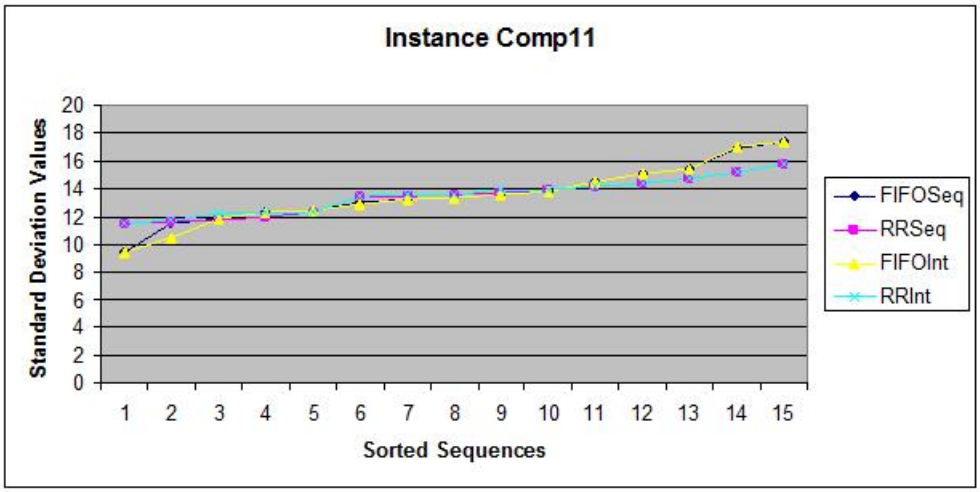

Figure 7.6: Sorted standard deviation values for instance Comp11

For this instance, the smallest standard deviation of satisfied values of resources of this experiment is an outcome of the *FIFOSeq* and *FIFOInt* principles which it is a result of the 2nd initial sequence.

The sequence which gives the smallest penalty value and the smallest standard deviation of satisfied value of resources is the 2nd initial sequence. The results are products of the *FIFOSeq* and *FIFOInt* principles.

Instance Comp13

The penalty values of instance *Comp13* from this experiment are presented by ranging from low to high values ordered by results of the *FIFOSeq* principle in Table 7.8 and the graph is plotted as shown in Figure 7.7.

Initial	FIFOSeq	RRSeq	FIFOInt	RRInt
Seq12	0.170	0.165	0.170	0.165
Seq09	0.178	0.167	0.178	0.167
Seq03	0.186	0.201	0.186	0.201
Seq08	0.186	0.190	0.186	0.190
Seq05	0.188	0.184	0.188	0.184
Seq11	0.192	0.185	0.192	0.185
Seq10	0.201	0.190	0.201	0.190
Seq01	0.210	0.198	0.210	0.198
Seq07	1.163	1.196	1.163	1.196
Seq14	1.170	0.187	1.170	0.187
Seq06	1.177	0.178	1.177	0.178
Seq04	1.206	0.163	1.206	0.163
Seq13	2.200	0.189	2.200	0.189
Seq15	3.189	0.193	3.189	0.193
Seq02	4.195	0.173	4.195	0.173

Table 7.8: Penalty values for instance Comp13

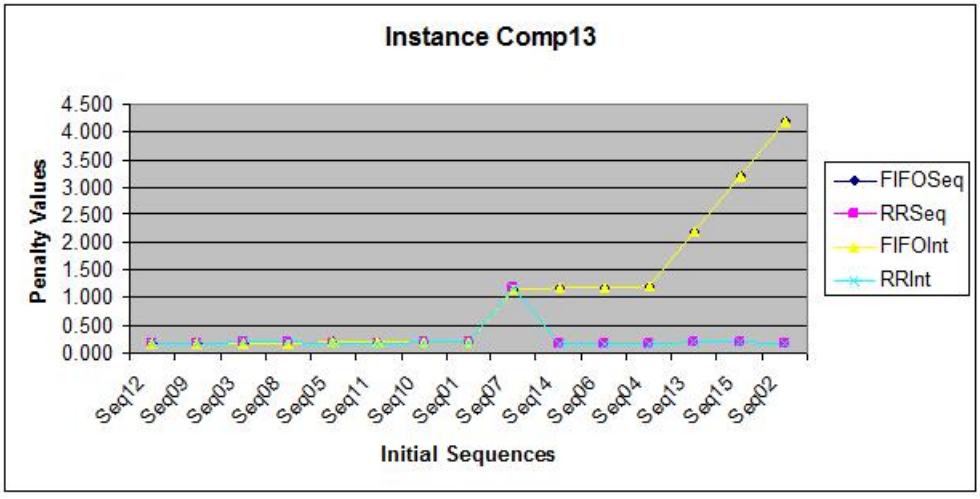

Figure 7.7: Penalty values for instance Comp13

For this instance, the best solution of this experiment is an outcome of the *RRSeq* and *RRInt* principles which it is a result of the 12nd initial sequence.
The standard deviation of the satisfied values of instance *Comp13* from this experiment is presented in Table 7.9 and is plotted in graph as shown in Figure 7.8. The sorted standard deviation values from low to high of each principle are plotted in graph as shown in Figure 7.9.

Initial	**FIFOSeq**	**RRSeq**	**FIFOInt**	**RRInt**
Seq12	13.78233	13.31006	13.78233	13.31006
Seq09	15.83617	14.85988	15.83617	14.85988
Seq03	14.67921	13.76559	14.67921	13.76559
Seq08	15.51432	13.42455	15.51432	13.42455
Seq05	14.52807	13.00819	14.52807	13.00819
Seq11	17.45976	14.03313	17.45976	14.03313
Seq10	14.31493	13.1725	14.31493	13.1725
Seq01	13.85319	13.61357	13.85319	13.61357
Seq07	15.61258	13.58496	15.61258	13.58496
Seq14	14.51541	13.60103	14.51541	13.60103
Seq06	14.34402	14.25821	14.34402	14.25821
Seq04	16.76861	13.07493	16.76861	13.07493
Seq13	15.42377	14.30181	15.42377	14.30181
Seq15	15.41013	12.79509	15.41013	12.79509
Seq02	14.34918	13.6524	14.34918	13.6524

Table 7.9: Standard deviation values for instance Comp13

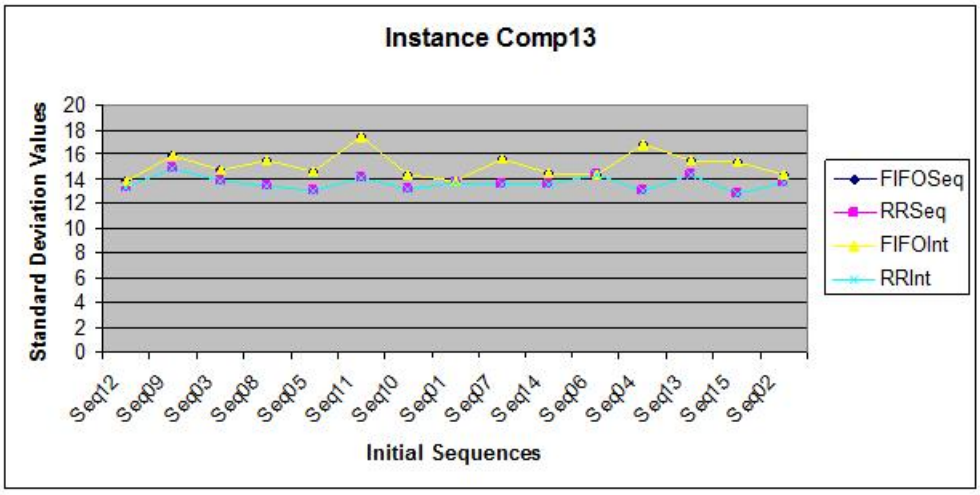

Figure 7.8: Standard deviation values for instance Comp13

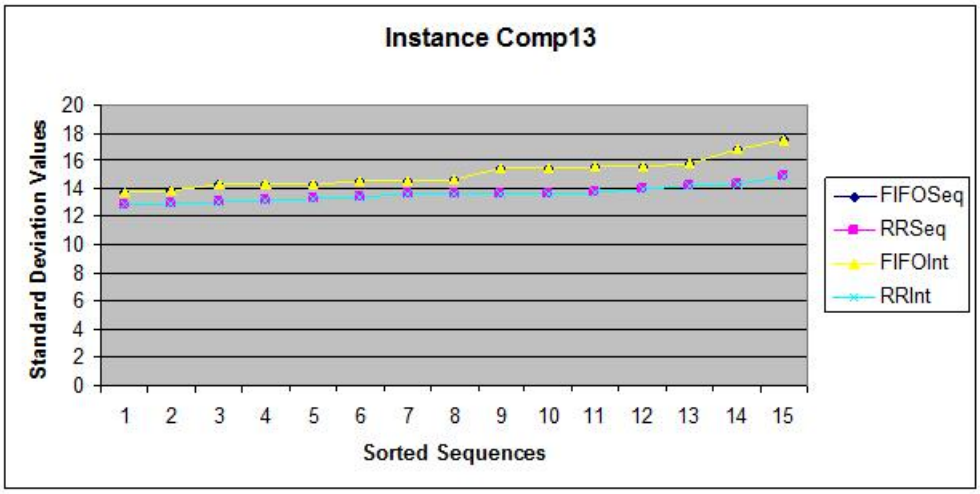

Figure 7.9: Sorted standard deviation values for instance Comp13

For this instance, the smallest standard deviation of satisfied values of resources of this experiment is an outcome of the *RRSeq* and *RRInt* principles which it is a result of the 15th initial sequence.

The sequences which give the smallest penalty value and the smallest standard deviation of satisfied values of resources are the 12nd or the 5th initial sequences. The results are products of the *RRSeq* and *RRInt* principles.

Instance Comp15

The penalty values of instance *Comp15* from this experiment are presented by ranging from low to high values ordered by results of the *FIFOSeq* principle in Table 7.10 and the graph is plotted as shown in Figure 7.10.

Initial	FIFOSeq	RRSeq	FIFOInt	RRInt
Seq10	1.206	4.261	1.206	4.252
Seq11	2.176	2.213	2.176	2.213
Seq05	2.187	4.201	2.187	4.203
Seq15	2.223	4.240	1.222	4.240
Seq14	3.230	3.245	3.230	3.241
Seq04	3.234	5.218	3.234	5.218
Seq06	3.246	7.223	3.242	7.223
Seq02	5.214	5.223	5.218	5.223
Seq13	5.226	4.232	5.226	4.242
Seq09	5.238	4.221	5.241	4.223
Seq01	6.200	3.243	6.205	3.243
Seq08	6.240	4.214	6.240	4.212
Seq03	6.244	8.241	6.247	8.241
Seq12	6.259	4.224	6.259	4.221
Seq07	7.201	7.235	7.197	7.235

Table 7.10: Penalty values for instance Comp15

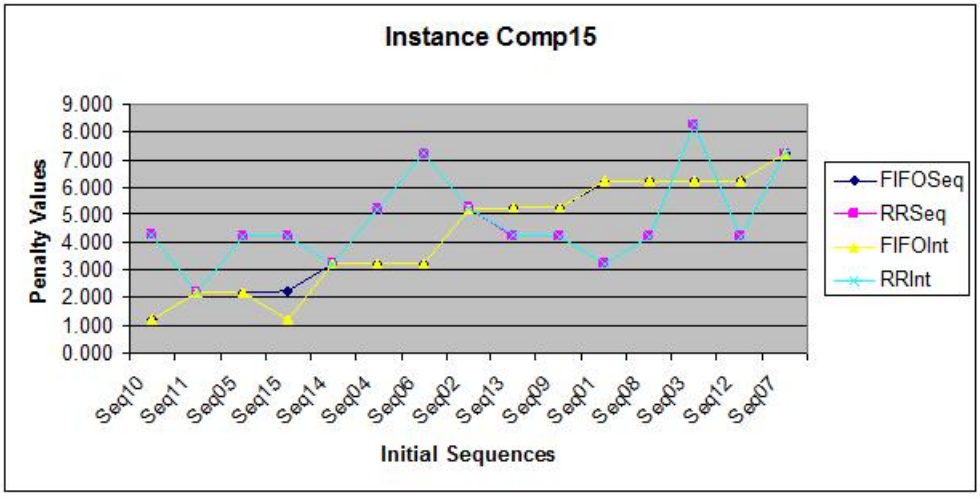

Figure 7.10: Penalty values for instance Comp15

For this instance, the best solution of this experiment is an outcome of the *FIFOSeq* and *FIFOInt* principles which it is a result of the 10th initial sequence.
The standard deviation of the satisfied values of instance *Comp15* from this experiment is presented in Table 7.11 and is plotted in graph as shown in Figure 7.11. The sorted standard deviation values from low to high of each principle are plotted in graph as shown in Figure 7.12.

Initial	**FIFOSeq**	**RRSeq**	**FIFOInt**	**RRInt**
Seq10	14.63138	14.18005	14.63138	12.60612
Seq11	14.72634	14.48279	14.72634	14.48279
Seq05	13.17876	17.92704	13.17876	17.93245
Seq15	14.7887	14.98562	14.00974	14.98562
Seq14	15.53601	16.01299	15.53601	16.08626
Seq04	14.51639	15.21514	14.51639	15.21514
Seq06	15.37644	14.1883	15.37644	14.1883
Seq02	13.58475	16.6845	13.58475	16.6845
Seq13	15.74166	15.39225	15.74166	15.49215
Seq09	14.47914	13.21881	14.67802	13.17881
Seq01	15.60737	13.98737	15.51995	13.98737
Seq08	16.36932	15.12801	16.36932	15.13316
Seq03	14.60386	15.76612	14.49255	15.76612
Seq12	13.47947	16.46453	13.47947	16.46177
Seq07	15.32802	15.08888	15.32183	15.08888

Table 7.11: Standard deviation values for instance Comp15

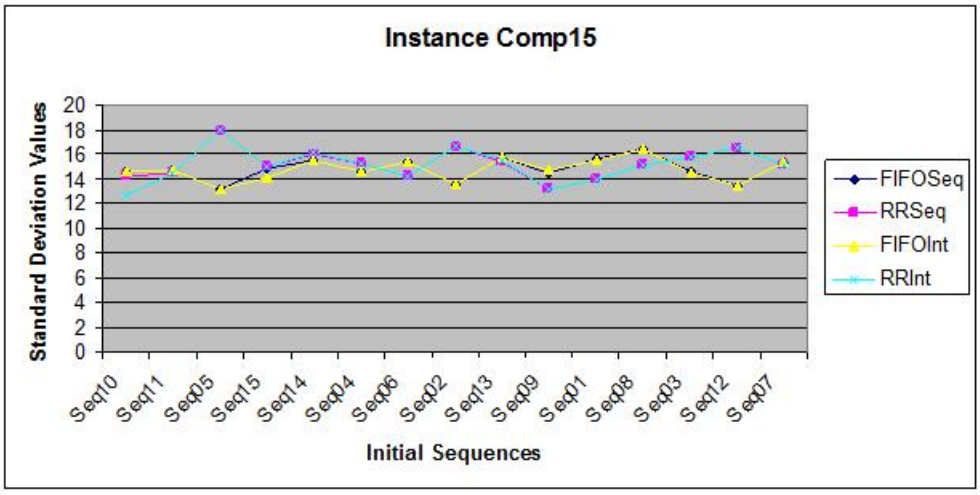

Figure 7.11: Standard deviation values for instance Comp15

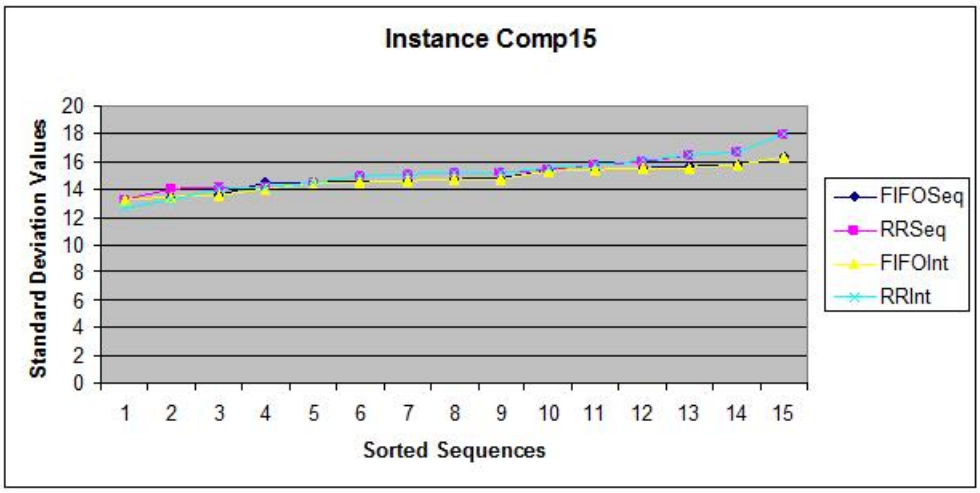

Figure 7.12: Sorted standard deviation values for instance Comp15

For this instance, the smallest standard deviation of satisfied values of resources of this experiment is an outcome of the *RRInt* principle which it is a result of the 10th initial sequence.

The sequence which give the smallest penalty value and the smallest standard deviation of satisfied values of resources is the 5th initial sequence. The results are products of the *FIFOSeq* and *FIFOInt* principles.

Instance Comp18

The penalty values of instance *Comp18* from this experiment are presented by ranging from low to high values ordered by results of the *FIFOSeq* principle in Table 7.12 and the graph is plotted as shown in Figure 7.13.

Initial	FIFOSeq	RRSeq	FIFOInt	RRInt
Seq10	0.197	0.185	0.197	0.185
Seq14	0.200	0.199	0.200	0.199
Seq09	0.202	0.195	0.202	0.195
Seq04	0.208	0.212	0.208	0.212
Seq11	0.208	0.195	0.208	0.185
Seq13	0.208	0.208	0.208	0.208
Seq15	0.213	0.206	0.213	0.200
Seq08	0.215	0.199	0.205	0.199
Seq07	0.216	0.203	0.216	0.203
Seq05	0.220	0.189	0.220	0.189
Seq06	0.221	0.197	2.220	0.197
Seq03	0.222	0.187	0.222	0.187
Seq12	0.229	0.186	0.229	0.186
Seq01	0.230	0.231	0.230	0.231
Seq02	0.250	0.182	0.250	0.182

Table 7.12: Penalty values for instance Comp18

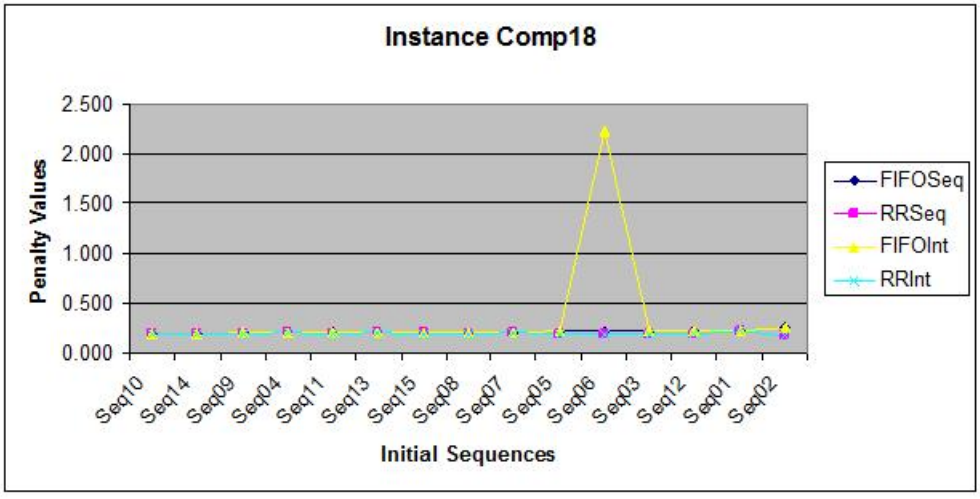

Figure 7.13: Penalty values for instance Comp18

For this instance, the best solution of this experiment is an outcome of the *RRSeq* and *RRInt* principles which it is a result of the 2nd initial sequence.
The standard deviation of the satisfied values of instance *Comp18* from this experiment is presented in Table 7.13 and is plotted in graph as shown in Figure 7.14. The sorted standard deviation values from low to high of each principle are plotted in graph as shown in Figure 7.15.

Initial	**FIFOSeq**	**RRSeq**	**FIFOInt**	**RRInt**
Seq10	15.47606	11.32257	15.47606	11.32257
Seq14	16.07973	16.86567	16.07973	16.86567
Seq09	14.92435	13.98611	14.92435	13.98611
Seq04	13.83098	13.75904	13.83098	13.75904
Seq11	15.57751	13.21952	15.57751	13.21952
Seq13	17.39632	15.02549	17.39632	15.02549
Seq15	13.13509	15.49012	13.13509	13.24001
Seq08	13.76339	12.58447	13.76339	12.58447
Seq07	15.90522	13.82034	15.90522	13.82034
Seq05	13.61634	16.75782	13.61634	16.75782
Seq06	13.81225	12.93513	15.46944	12.93513
Seq03	16.66444	13.25634	16.66444	13.25634
Seq12	16.82012	13.81071	16.82012	13.81071
Seq01	16.79025	13.76634	16.79025	13.76634
Seq02	17.73978	12.99924	17.73978	12.99924

Table 7.13: Standard deviation values for instance Comp18

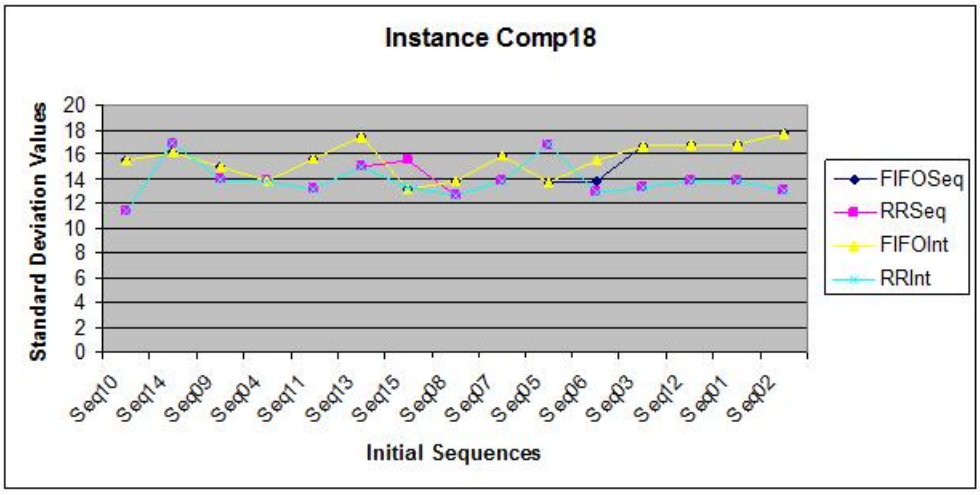

Figure 7.14: Standard deviation values for instance Comp18

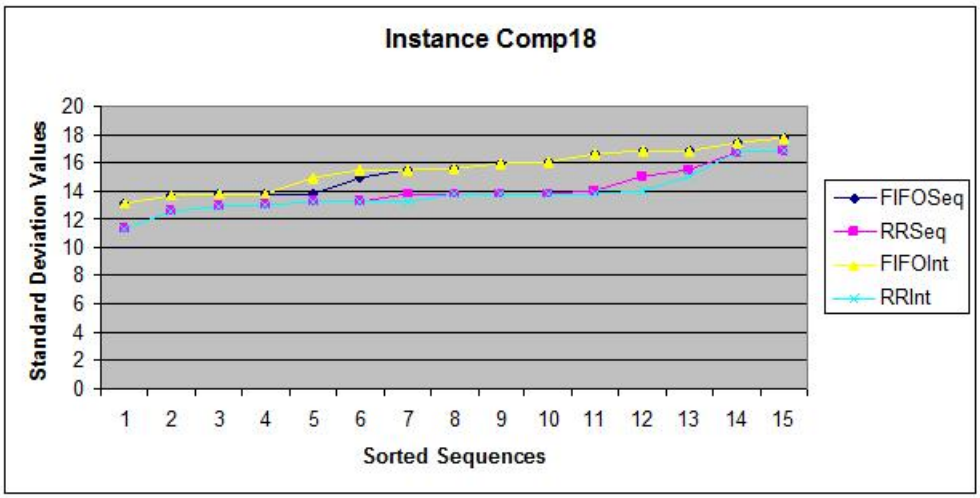

Figure 7.15: Sorted standard deviation values for instance Comp18

For this instance, the smallest standard deviation of satisfied values of resources of this experiment is an outcome of the *RRSeq* and *RRInt* prnciples which it is a result of the 10th initial sequence.

The sequence which gives the smallest penalty value and the smallest standard deviation of satisfied values of resources is the 10th initial sequence. The results are products of the *RRSeq* and *RRInt* principles.

H4) The summarized results of the above experiment are shown in Table 7.14 by presenting the principle which gives the smallest value of each instance.

instance	**The Smallest Penalty Value**			
	FIFOSeq	**RRSeq**	**FIFOInt**	**RRInt**
Comp04				√
Comp11	√		√	
Comp13		√		√
Comp15	√		√	
Comp18		√		√
Frequency	**2**	**2**	**2**	**3**

Table 7.14: The smallest penalty value of each instance

Information in Table 7.14 and the conditions which are designed for this experiment are indicated that hypothesis *H4* is not supported. The *First-In-First-Out & Sequential* principle does not always give the smallest penalty value even though it expected that it provides more chances for some *YPAs,* especially those located on the top of the list, to pick good resources which meet their needs first. We found some principles such as *Round Robin & Sequential, Round Robin & Interleaved and First-In-First-Out & Interleaved* also give the smallest penalty value for some instances.

H5) The summarized results of the above experiment are shown in Table 7.15 by presenting the principle which gives the largest penalty value of each instance.

instance	**The Largest Penalty Value**			
	FIFOSeq	**RRSeq**	**FIFOInt**	**RRInt**
Comp04	√			
Comp11			√	
Comp13	√		√	
Comp15		√		√
Comp18			√	
Frequency	**2**	**1**	**3**	**1**

Table 7.15: The largest penalty value of each instance

Information in Table 7.15 and the conditions which are designed for this experiment are indicated that hypothesis *H5* is not supported. The *First-In-First-Out & Interleave*d principle does not always give the largest penalty value, even though we believe that some *YPAs*, especially those located on the tail of the list, will face high penalty values

as their desired resources have been occupied by the previous *YPAs*. However, it tends to be so. Occasionally, some principles such as *First-In-First-Out & Sequential, Round Robin & Sequential and Round Robin & Interleaved* also give the largest penalty value for some instances.

H6) The summarized results of the above experiment are shown in Table 7.16 by presenting the principle which gives the smallest standard deviation value of each instance.

instance	**The Smallest Standard Deviation Value**			
	FIFOSeq	**RRSeq**	**FIFOInt**	**RRInt**
Comp04		√		
Comp11	√		√	
Comp13		√		√
Comp15				√
Comp18		√		√
Frequency	**1**	**3**	**1**	**3**

Table 7.16: The smallest standard deviation value of each instance

Information in Table 7.16 and the conditions which are designed for this experiment are indicated that hypothesis *H6* is not supported.The *Round-Robin & Interleaved* principle does not always give the smallest standard deviation of satisfied values, even though it is expected to provide more chances to reach the desired resources equally among *YPAs* in the system. However, *Round-Robin & Sequential* still gives the smallest standard deviation for some instances. Both *Round-Robin & Interleaved* and *Round-Robin & Sequential* tend to give more sucessful in fairness among *YPAs* than both *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved* principles.

H7) The summarized results of the above experiment are shown in Table 7.17 by presenting the principle which gives the largest standard deviation value of each instance.

instance	The Largest Standard Deviation Value			
	FIFOSeq	**RRSeq**	**FIFOInt**	**RRInt**
Comp04				√
Comp11	√		√	
Comp13	√		√	
Comp15		√		√
Comp18	√		√	
Frequency	**3**	**1**	**3**	**2**

Table 7.17: The largest standard deviation value of each instance

The information in Table 7.17 and the conditions used for this experiment indicate that hypothesis *H7* is not supported. The *First-In-First-Out & Sequential* principle does not always give the largest standard deviation of satisfied values, even though we expected that First-In-First-Out & Sequential would give a big difference between satisfied values among *YPAs* in the system. However, both *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved* principles seem to give unfair allocations among *YPAs*, whereas some instances *of Round Robin & Sequential* and *Round Robin & Interleaved* principles also give the largest standard deviation value .
A link between *standard deviation value* and *fairness* can be explained, as follows:

A set of satisfied values which has been returned after a set of timetables has been organized indicates how much each YPA in the system is satisfied by the set of resources it occupies. The standard deviation value of the set of satisfied values indicates the difference of the average for the satisfied values amongs YPAs in the system. That means that the smaller the standard deviation value is, the closer the satisfied values tend to be to the average. When each YPA in system gets nearly the same satisfied value, it can be assumed that each YPA got a fair chance to reach its desired resources.

H8) Increasing the number of agents(curricula) gives larger penalty values.(An agent takes a responsibility to organize a set of courses in a curriculum.)
In order to test hypothesis *H8*, we need to compare the results which produce from different numbers of agents by the type of principles. For each of the random samples, there are different numbers of curricula, which mean there are different numbers of YPAs in each instance; as we defined in section 6.2 that one YPA takes the responsibility for organizing a set of courses as defined in a curriculum. We then take information which was recorded from above experiment to rearrange by ranging the validated penalty values from low to high by running the number of *YPAs* in system from low to high by principle

types.

- ***FIFOSeq***: the ordered results of the *FIFOSeq* principle are presented in table and graph forms, as shown in Table 7.18 and plotted in Figure 7.16

Comp11 13YPAs	**Comp18 52YPAs**	**Comp04 57YPAs**	**Comp13 66YPAs**	**Comp15 68YPAs**
0.007	0.197	1.160	0.170	1.206
0.007	0.200	2.140	0.178	2.176
0.012	0.202	2.148	0.186	2.187
0.013	0.208	2.156	0.186	2.223
0.014	0.208	2.173	0.188	3.230
0.016	0.208	3.154	0.192	3.234
0.017	0.213	3.178	0.201	3.246
0.017	0.215	3.195	0.210	5.214
0.018	0.216	4.138	1.163	5.226
0.020	0.220	4.170	1.170	5.238
0.021	0.221	4.181	1.177	6.200
0.028	0.222	4.193	1.206	6.240
0.040	0.229	5.148	2.200	6.244
0.041	0.230	5.149	3.189	6.259
0.055	0.250	7.178	4.195	7.201

Table 7.18: *FIFOSeq*'s ordered penalty values when the number of *YPAs* in system is increased

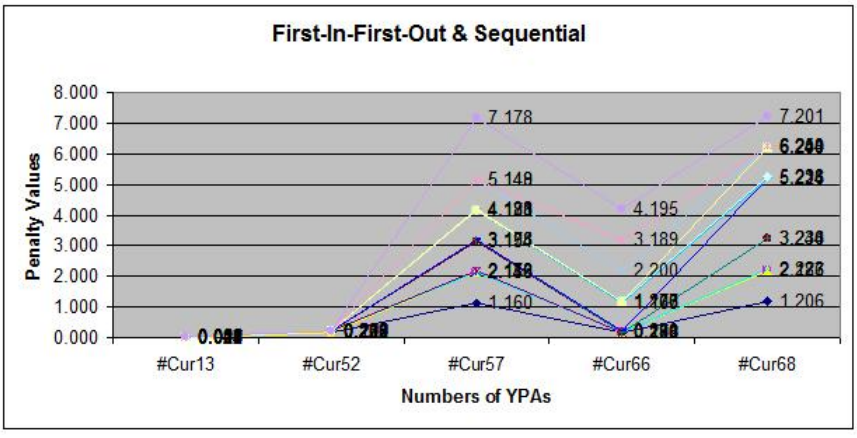

Figure 7.16: *FIFOSeq*'s ordered penalty values when the number of *YPAs* in system is increased

The information in Table 7.18 and the graph in Figure 7.16 show that increasing the number of agents does not always give larger penalty values.

- ***RRSeq***: the ordered results of the *RRSeq* principle are presented in table and graph forms, as shown in Table 7.19 and plotted in Figure 7.17

Comp11 13YPAs	**Comp18 52YPAs**	**Comp04 57YPAs**	**Comp13 66YPAs**	**Comp15 68YPAs**
0.023	0.182	0.144	0.163	2.213
0.028	0.185	0.145	0.165	3.243
0.033	0.186	0.160	0.167	3.245
0.033	0.187	1.138	0.173	4.201
0.035	0.189	1.147	0.178	4.214
0.040	0.195	1.152	0.184	4.221
0.041	0.195	1.154	0.185	4.224
0.043	0.197	1.155	0.187	4.232
0.045	0.199	2.153	0.189	4.240
0.047	0.199	2.153	0.190	4.261
0.051	0.203	2.155	0.190	5.218
0.052	0.206	2.166	0.193	5.223
0.052	0.208	2.167	0.198	7.223
0.053	0.212	2.171	0.201	7.235
0.054	0.231	3.168	1.196	8.241

Table 7.19: *RRSeq*'s ordered penalty values when the number of *YPAs* in system is increased

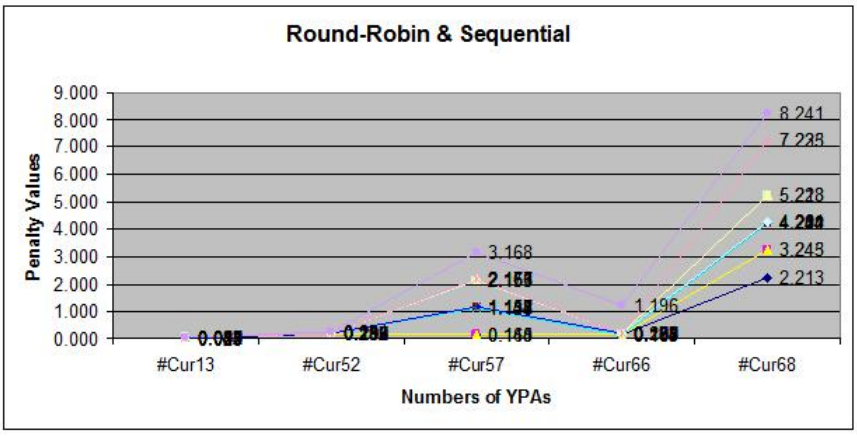

Figure 7.17: *RRSeq*'s ordered penalty values when the number of *YPAs* in system is increased

The information in Table 7.19 and the graph in Figure 7.17 show that increasing the number of agents does not always give larger penalty values.

- ***FIFOInt***: the ordered results of the *FIFOInt* principle are presented in table and graph forms, as shown in Table 7.20 and plotted in Figure 7.18

Comp11 13YPAs	Comp18 52YPAs	Comp04 57YPAs	Comp13 66YPAs	Comp15 68YPAs
0.007	0.197	1.141	0.170	1.206
0.007	0.200	1.155	0.178	1.222
0.012	0.202	2.121	0.186	2.176
0.013	0.205	2.141	0.186	2.187
0.014	0.208	2.142	0.188	3.230
0.014	0.208	2.143	0.192	3.234
0.017	0.208	2.145	0.201	3.242
0.017	0.213	3.129	0.210	5.218
0.018	0.216	3.164	1.163	5.226
0.020	0.220	4.110	1.170	5.241
0.021	0.222	4.147	1.177	6.205
0.028	0.229	4.147	1.206	6.240
0.053	0.230	5.119	2.200	6.247
3.045	0.250	5.131	3.189	6.259
4.033	2.220	5.135	4.195	7.197

Table 7.20: *FIFOInt*'s ordered penalty values when the number of *YPAs* in system is increased

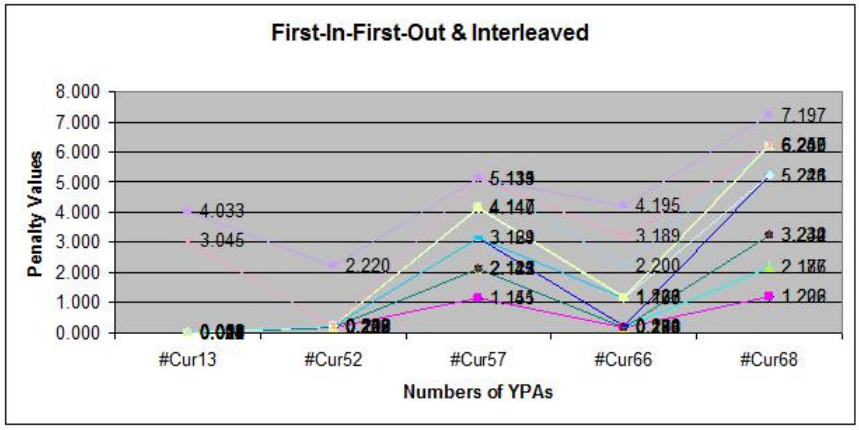

Figure 7.18: *FIFOInt*'s ordered penalty values when the number of *YPAs* in system is increased

The information in Table 7.20 and the graph in Figure 7.18 show that increasing the number of agents does not always give larger penalty values.

- ***RRInt***: the ordered results of the *RRInt* principle are presented in table and graph

forms, as shown in Table 7.21 and plotted in Figure 7.19

Comp11 13YPAs	**Comp18 52YPAs**	**Comp04 57YPAs**	**Comp13 66YPAs**	**Comp15 68YPAs**
0.022	0.182	0.140	0.163	2.213
0.028	0.185	0.152	0.165	3.241
0.031	0.185	0.160	0.167	3.243
0.033	0.186	1.138	0.173	4.203
0.033	0.187	1.147	0.178	4.212
0.042	0.189	1.152	0.184	4.221
0.043	0.195	1.155	0.185	4.223
0.043	0.197	1.157	0.187	4.240
0.047	0.199	2.153	0.189	4.242
0.050	0.199	2.153	0.190	4.252
0.051	0.200	2.155	0.190	5.218
0.052	0.203	2.166	0.193	5.223
0.054	0.208	2.167	0.198	7.223
0.057	0.212	2.168	0.201	7.235
0.059	0.231	2.171	1.196	8.241

Table 7.21: *RRInt*'s ordered penalty values when the number of *YPAs* in system is increased

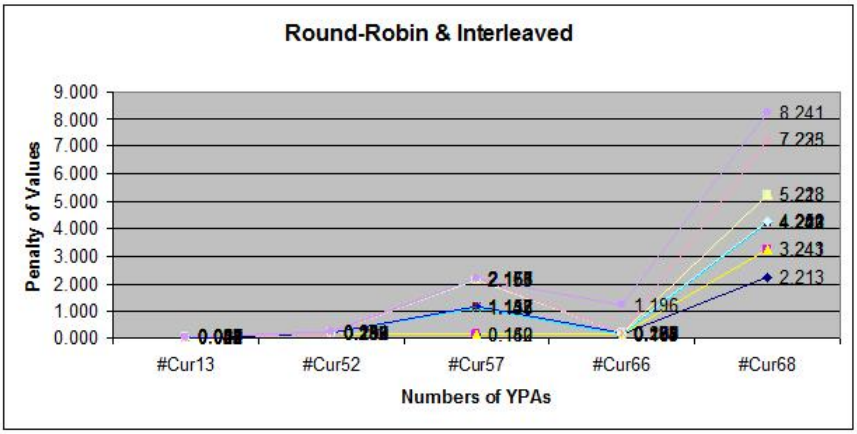

Figure 7.19: *RRInt*'s ordered penalty values when the number of *YPA*s in system is increased

The information in Table 7.21 and the graph in Figure 7.19 show that increasing the

number of agents does not always give larger penalty values.

instance	**The best results from Our Model**		**The best results from the Reference**	
	Number of Essential Constraints Violated	**Number of Desirable Constraints Violated**	**Number of Essential Cnstraints Violated**	**Number of Desirable Constraints Violated**
Comp04	0	140	0	18
Comp11	0	7	0	0
Comp13	0	163	0	32
Comp15	1	206	0	38
Comp18	0	182	0	34

Table 7.22: Comparing the best of ours with the reference results

H9) The proposed principles generally produce smaller penalty values when compared with the *ITC-2007* reference results.

In order to test hypothesis *H9*, we compare the smallest penalty results of each instance that we have got from the above experiment with the reference results which were provided by *ITC-2007*in Table 6.3. The corresponding results are compared and presented in Table 7.22.

The information in Table 7.22 indicates that hypothesis *H9* is not supported. The proposed principles give larger penalty values for all chosen instances when compared with the reference results. However, if we drill down into the penalty values and classify them according to the considered constraints, then we find that the vast majority of penalty values are caused by isolated-lecture violations as shown in Table 7.23. If considered in terms of achievements, the percentage of achievements for each constraint is presented in Table 7.24. For clarification,

1) In order to identify how well our model is able to achieve each constraint, this information is useful for improving the weaker parts of the model.

2) As the random samples have different numbers of curricula, courses, lectures, students, rooms and so on, we need to compare the success of constraints between samples; so the percentage of achievements are needed to calculate for the purpose of comparing.

instance	principle	Essential Constraints				Desirable Constraints		
		Lectures	Conflicts	Availability	Room Occupancy	Room Capacity	MinWork Days	Isolated Lectures
Comp04	RRInt	0	0	0	0	0	10	130
Comp11	FIFO Seq/Int	0	0	0	0	0	0	7
Comp13	RR Seq/Int	0	0	0	0	0	5	158
Comp15	FIFO Seq/Int	1	0	0	0	0	20	186
Comp18	RR Seq/Int	0	0	0	0	0	0	177

Table 7.23: Classified penalty values into constraints

instance	principle	Essential Constraints				Desirable Constraints		
		Lectures	Conflicts	Availability	Room Occupancy	Room Capacity	MinWork Days	Isolated Lectures
Comp04	RRInt	100	100	100	100	100	99.07	76.16
Comp11	FIFO Seq/Int	100	100	100	100	100	100	97.27
Comp13	RR Seq/Int	100	100	100	100	100	99.57	76.13
Comp15	FIFO Seq/Int	99.88	100	100	100	100	98.15	76.87
Comp18	RR Seq/Int	100	100	100	100	100	100	62.89

Table 7.24: Percentage of achievements from the best results of our models

H10) The *Round Robin & Interleaved* principle is most likely to produce better solutions in terms of both smallest penalty value and best degree of fairness when compared to the other principles.

In order to test hypothesis *H10*, we define better solutions in the meaning of that solution giving the smallest penalty value/the smallest standard deviation of satisfied values/the smallest in both penalty value and standard deviation of satisfied values. From the results

of above experiment, for each instance we determine type of principles which is able to produce timetables with either smallest penalty value or smallest standard deviation or both; and then count the number of success. The results of this investigation are presented in Table 7.25.

Instance	**smallest**	**FIFOSeq**	**RRSeq**	**FIFOInt**	**RRInt**
Comp04	Penalty				√
	STDEV		√		
	Penalty & STDEV		√		√
Comp11	Penalty	√		√	
	STDEV	√		√	
	Penalty & STDEV	√		√	
Comp13	Penalty		√		√
	STDEV		√		√
	Penalty & STDEV		√		√
Comp15	Penalty	√		√	
	STDEV				√
	Penalty & STDEV	√		√	
Comp18	Penalty		√		√
	STDEV		√		√
	Penalty & STDEV		√		√
Frequency		**5**	**8**	**5**	**9**

Table 7.25: Frequency of achievement/fairness/achievement & fairness by models

Information in Table 7.25 indicates that hypothesis *H10* is supported, as the *Round Robin & Interleaved* principle is most likely to produce preferable solutions when compared to the others. However, from the counted information, it indicates that the *Round Robin & Sequential* principle also tends to give more preferable solutions than *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved* principles.

7.4 Summary

This chapter has presented the experimental research method. It reviews the research questions and describes how a set of hypotheses has been identified. There are ten hypotheses used for this study. Each hypothesis has been described, and the experimental process used to address each one defined. The experimental results for each are recorded and the conclusion of each is summarized.

Chapter 8

Discussion

The previous chapter presents a number of sets of experimental results. This chapter will interpret and discuss the results in terms of the original research questions. It starts by summarizing and interpreting the results of testing the hypotheses; then addresses the research questions, clarifying the quality of results from each principle and identifying the threats which influence the quality of results.

8.1 Analysis of the Experimental Results

The experimental results from previous chapter have demonstrated that agent-based technology has the potential to address distributed university course timetabling problems that not only include a set of essential constraints but also a set of desirable constraints that correspond to real world needs. All test cases used have been taken from real world situations.

H1) All four proposed multiagent principles are able to achieve the university course timetabling formula and constraints which were defined for the *International Timetabling Competition 2007*.

The results of testing hypothesis *H1* indicate that the four different proposed principles— *FIFO&Seq, RR&Seq, FIFO&Int* and *RR&Int* — have potential to achieve successful results for the university course formula and constraints that were defined for *the International Timetabling Competition 2007*. The agent-based system was able to schedule all lectures in different periods with no instances of two lectures occupying the same room at the same time. The lectures belonging to the same curriculum or taught by the same

teacher do not conflict. Every lecture has been assigned a teacher. Every assigned room has a capacity that is greater than or equal to the number of students in that course. The lectures of each course have been spread across the minimum number of working days, and the lectures under the same curriculum have been tried to avoid isolated as many as possible.

H2) The number of successfully organized lectures have been increased after applying the *negotiation phase* following the *initial allocation phase* of both *FIFO* and *RR* algorithms.

The results of testing hypothesis *H2* demonstrate that the number of organized lectures have been increased after applying the *negotiation phase* following the *initial allocation phase* of both *FIFO* and *RR* algorithms. That means that the constraint mismatched problem could be solved by negotiation, as we expected. However, there are many factors that influence the success of negotiation, such as the resources which are held by other *YPAs*, the rooms that are free at that moment, the constraints that accompany the courses from both sides. These might cause the number of organized lectures has not been increased after applying the *negotiation phase* in some instances.

H3) Different initial sequences of *YPAs* will produce different results.

The results of testing hypothesis *H3* evidently show that any two initial sequences of *YPAs* produce different results when running them on the same principle. That means the initial ordering of access to the *RA* by the *YPAs* can have a significant effect upon the outcomes. Some orderings of *YPAs* can lead to an outcome that achieves all the essential constraints and most of the defined desirable constraints, while some orderings are unable to meet even all the essential constraints. As all possible initial *YPAs* orderings are $n!$ when n represents numbers of *YPAs* in system, the set of sequences which have been chosen randomly for the experiments are very few when compared with the total numbers. Therefore, the initial ordering of *YPAs* is an external threat which can have influence on the achievements of the system.

H4) Among the four models, the *First-In-First-Out & Sequential* principle will always give the smallest penalty value when compared with the other principles.

The results of testing hypothesis *H4* show that the *First-In-First-Out & Sequential* model does not always give the smallest penalty value, even though it was expected that it would provide more chances for *YPAs* on the top of the list to pick good resources which meet their needs. We found some principles such as *Round Robin & Sequential, Round Robin & Interleaved* and *First-In-First-Out & Interleaved* also give the smallest penalty value for some instances. Moreover, the experiment results do not conclusively

show that the *Round-Robin & Interleaved* principle is better than the other principles in achieving the best results and will always return the smallest penalty values even if it tends to be so. In a timetabling survey paper, Lewis (2007) observed that a comparison between one new algorithm and the others which has been proposed for the same problem may reveal that the new approach is not always able to perform well in some instances. And even for the reference set, we also have not found any algorithm or technique which always returns the smallest penalty values for all instances, as is shown in Table 6.3. Therefore, we can conclude that we have not found any one strategy to be generally most successful.

H5) Among the four principles, the *First-In-First-Out & Interleaved* principle will always give the largest penalty value when compared with the other principles.

The results of testing hypothesis *H5* show that the *First-In-First-Out & Interleaved* principle does not always give the largest penalty value, even though we believe that some *YPAs*, especially those located on the tail of the list, will face high penalty values as their desired resources have been occupied by previous *YPAs*. However, it tends to be so. On occasion, some principles such as *First-In-First-Out & Sequential, Round Robin & Sequential* and *Round Robin & Interleaved* also give the largest penalty value for some instances.

H6) Among the four principles, the *Round-Robin & Interleaved* principle will always give the smallest standard deviation of satisfied values when compared with the other principles.

The results of testing hypothesis *H6* indicate that the *Round-Robin & Interleaved* principle does not always give the smallest standard deviation of satisfied values, even though it was expected that it would provide more chances to reach the desired resources equally. However, *Round Robin & Sequential* also seems to give the smallest standard deviation of satisfied values for some instances. They both tend to be more fair than the *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved* principles. The standard deviation graphs from Figures 7.3, 7.9 and 7.15 obviously show that *Round-Robin & Interleaved* and *Round-Robin & Sequential* principles generally lead to more fair resource allocation among *YPAs* in the system than *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved*, whereas the standard deviation values of instances *Comp11* and *Comp15* show that all four principles give nearly the same fair resource allocation among *YPAs*, as shown in Figures 7.6 and 7.12.

H7) Among the four principles, the *First-In-First-Out & Sequential* principle will always

give the largest standard deviation of satisfied values when compared with the other principles.

The results of testing hypothesis *H7* indicated that the *First-In-First-Out & Sequential* principle does not always give the largest standard deviation of satisfied values, even though we expected that First-In-First-Out & Sequential would give a big difference between satisfied values among *YPAs* in system. However, both *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved* seem to give unfair allocations among *YPAs*, whereas *Round Robin & Sequential* and *Round Robin & Interleaved* principles also give the largest standard deviation value in a few instances.

Each *YPA* determines its satisfied value using information about the number of students in each class and the capacities of rooms which had been defined in each scenario; therefore, such values can form an internal threat as they directly influence the quality of the results, especially when these are few students on courses, and mostly big rooms are provided in the scenario.

H8) Increasing the number of agents (curricula) gives larger penalty values. (An agent takes a responsibility to organize a set of courses in a curriculum.)

The results of testing hypothesis *H8* show that increasing the number of agents does not always give larger penalty values, even it tends to be so. From the plotted graphs in Figures 7.16, 7.17, 7.18 and 7.19, we found that *Round-Robin & Sequential* and *Round-Robin & Interleaved* seem to be able to achieve the constraints better than the *First-In-First-Out & Sequential* principle is able to achieve under the same number of *YPAs*, while *First-In-First-Out & Interleaved* principle tends to give high penalty values even when the number of *YPAs* in the system is not many.

H9) The proposed principles generally produce smaller penalty values when compared with the reference results.

The results of testing hypothesis *H9* indicate that our proposed principles do not generally produce smaller penalty values when compared with the reference results. However, our algorithm is able to solve the university course timetabling problem as well as other techniques have done. Currently, the proposed algorithm can achieve all of the essential constraints and most of the needs of the desirable constraints except for the isolated-lecture constraint, for which our principles still return quite high penalty values when compared with the other constraints, as shown in Table 7.23. If considered in terms of the percentage of success, the isolated lectures have been avoided for more than 75% on average, while the minimum working days requirement also has been achieved for more

than 98% and the other constraints have already been one hundred percent achieved.
H10) The *Round Robin & Interleaved* principle is most likely to produce better solutions in terms of both smallest penalty value and best degree of fairness when comparing to the other principles.

The results of testing hypothesis *H10* confirm that the *Round-Robin & Interleaved* and *Round-Robin & Sequential* principles tend to produce better solutions when compared with the other principles, in terms of both the lowest penalty value and also best degree of fairness under the set of instances that we have randomly chosen for this experiment. Nevertheless, some instances such as *Comp11* and *Comp15* have demonstrated that successful results can also be produced by the *First-In-First-Out Sequential* and *First-In-First-Out Interleaved* principles. So, we need to do more intensive analysis into the set of instances in order to find out what are the factors or characteristics of the instances which produce different results when applying different principles.

8.2 Addressing the Research Questions

In summary, the results of testing the hypotheses in Chapter 7 show that

- Agent-based technology has the ability to allocate university resources so as to satisfy the essential constraints for serving the teaching activities and to address those desirable constraints that correspond to real world needs, according to the formula which is defined from the *International Timetabling Competition 2007*.

- Each agent gets a fair chance to acquire the resources that it needs when a *Round-Robin* algorithm has been applied over the controlled loop of the *RA*. These results are derived by considering the standard deviation of the satisfied values. The standard deviation values which are produced from using *Round-Robin* are less than the standard deviation values which are produced from *First-In-First-Out*. That means the variation in the average of satisfied values among *YPAs* in systems that applied *Round-Robin* is less than the difference of averages of satisfied values among *YPAs* which applied *First-In-First-Out*. Therefore, after applying the *Round-Robin* algorithm over the *RA* controlled loop gives each agent more chance to satisfy its own resource needs.

- The different qualities of results that are products of *First-In-First-Out & Sequential (FIFOSeq)*, *First-In-First-Out & Interleaved (FIFOInt)*, *Round-Robin & Sequential*

(RRSeq) and *Round-Robin & Interleaved (RRInt)* principles have been explored by comparing the timetabling results in terms of

- Constraint achievements: so far we found that every principle has the potential to sort out this university course timetabling problem and meets the needs of the constraints. From the hypotheses tested, we found that preferable solutions (least penalty & least standard deviation) for some instances are produced by the *Round-Robin & Sequential (RRSeq)* and *Round-Robin & Interleaved (RRInt)* principles, while some are produced by the *First-In-First-Out & Sequential (FIFOSeq)*, *First-In-First-Out & Interleaved (FIFOInt)* principles. However, in terms of frequency the *Round-Robin & Interleaved (RRInt)* and *Round-Robin & Sequential (RRSeq)* principles tend to be more successful than *First-In-First-Out & Sequential (FIFOSeq)*, *First-In-First-Out & Interleaved (FIFOInt),* as shown from the tests. Moreover, when the numbers of *YPAs* in system is increased, the constraint achievements from every principle tend to decrease (penalty values increase). Nevertheless, the *Round-Robin & Sequential* and *Round-Robin & Interleaved* principles seem to be able to achieve the constraints better than the *First-In-First-Out & Sequential* principle for the same number of *YPAs*, while *First-In-First-Out & Interleaved* principle tends to give high penalty values even when the number of *YPAs* in system is small. So far, we have not found any strategy which is generally the most successful strategy, giving both the smallest penalty value and the smallest standard deviation value for all instances. Moreover, more analysis of the set of instances is needed in order to figure out what are the factors or characteristics of the instances which lead to different results when applying different principles.
- Fairness issue: the sorted standard deviation values of each sample in Figures 7.3, 7.6, 7.9, 7.12 and 7.15 show that the standard deviation values produced by *Round-Robin & Interleaved* and *Round-Robin & Sequential* principles are smaller than the standard deviation values of *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved* principles. That means the differences in the average of satisfied values among *YPAs* in systems which applied the *Round-Robin* algorithm are less than the differences in the average of satisfied values among *YPAs* which applied *First-In-First-Out* algorithm. By applying

the *Round-Robin* algorithm therefore, each agent gets more chances to satify its own resource needs.

8.3 Summary

In this chapter, we have discussed the experimental outcomes which are produced from the work of the previous chapter. The relation between *causes* and *effects* for each hypothesis has been *stated*, *analysed* and *interpreted* respectively. And then the research question has been answered from these outcomes.

Chapter 9

Conclusion and Future Work

This final chapter presents the conclusion of the thesis, summarizes the research results, iterates the research contributions and identifies some work and some questions which need to be undertaken in order to improve the model and investigate its behaviour further.

9.1 Summary

This research has investigated the use of an agent-based model for solving the university course timetabling problem by generating resource allocation solutions for all participants in a system, considering not only a set of essential constraints for serving the teaching activities, but also a set of desirable constraints which correspond to the real world needs, and seeking to provide fair allocation among them. In this model, two types of agents have been defined to work together in the roles of *Year-Programme Agent* and a *Rooms Agent*.

- A *Year-Programme Agent (YPA)* has responsibility for generating the timetable for one level of a particular programme. It organizes a set of courses which that programme's students have to take in one particular term. The organized results must meet the essential constraints and reach as many as possible of the requirements of the desirable constraints. Another crucial role of the *Year-Programme Agent* is to collaborate with other *YPAs* to refine room allocation.
- The *Rooms Agent (RA)* manages the resources of rooms and books a requested room when that room is vacant. It coordinates the YPAs to work together in order

to avoid overlaps across the shared modules and takes responsibility for ordering the access sequence of YPAs to allocate the resources as fairly as possible.

The total number of agents in the system is equal the number of *YPAs*, which are required for the different degree programmes and levels, plus one *RA*.
The capabilities of the agents are harnessed to aid decision making, collaboration, cooperation and negotiation. The interactions defined for each principle imitate timetabling activities from the real world. The design is set to mimic a human scheduler's behaviour, by performing the following two phases:

- *Initial allocation phase*: to provide an initial allocation of resources between *YPAs* and the *RA*. As the need is to differentiate the chances to acquire the desired set of resources in YPAs, with two types of initial allocation being defined as:
  - *First-In-First-Out*: each *YPA* in turn makes a complete set of requests to the *RA*.
  - *Round-Robin*: in each round, each *YPA* presents one request to the *RA* in turn, with the requests from the set of *YPAs* being re-ordered dynamically for the next round by using the degree of satisfaction achieved in the last round.
- *Negotiation phase*: used to refine the allocation of rooms between *YPAs*

The overall steps of organizing timetables involve two distinct forms; these are:
• *Sequential*: start by running the initial allocation phase until this phase terminates, and then running the negotiation phase subsequently if any subjects still remain in the constraint-mismatch subject set.
• *Interleaved*: start by running the initial allocation phase and interleave this with the negotiation phase if any *YPA* faces a constraint-mismatched problem when organizing that subject.After the problem is resolved, the allocation phase will continue.
This research has investigated four different principles to implement the distributed university course timetables. These are: *First-In-First-Out & Sequential (FIFOSeq)*, *Fist-In-First-Out & Interleaved (FIFOInt)*, *Round-Robin & Sequential (RRSeq)* and *Round-Robin & Interleaved (RRInt)*. The two research questions were:

1. how to apply agent-based technology to allocate the university resources so as to satisfy the essential constraints and address the desirable constraints?

2. how to ensure that each agent gets a fair chance to acquire the resources that it needs?

this study has also investigated the different qualities resulting from above four principles, by comparing the timetabling results in terms of how well the constraints were addressed and the degree of fairness achieved when allocating rooms to *YPAs*. The problem formula and data instances of the third track of *the Second International Timetabling Competition (ITC-2007)* were used as benchmarks for validating these principles.

The results indicate that agent-based technology has the potential to address university course timetabling problems by including both the set of essential constraints and the set of desirable constraints which correspond to real world needs. The proposed model is able to schedule all lectures in different periods and we have not found any instances of two lectures occupying the same room at the same time. The lectures belonging to the same curriculum or taught by the same teacher do not conflict. Every lecture has been assigned a teacher before scheduling. Every assigned room is greater than or equals the number of students in that course. The lectures of each course have been spread into the minimum number of working days, and the isolated lectures under the same curriculum have been avoided as few as possible.

The negotiation phase is able to refine the room resources' allocation. However, how many times the *YPAs* have to try negotiation; or how well the negotiation succeed also depends on many factors, such as the resources which held by other *YPAs*, the available room resources which are free at that moment, the constraints which accompany with the courses from both sides.

Five instances from the *ITC2007*'s instances have been randomized to be the samples for the experiment. Each instance was run for fifteen times with different initial sequences of *YPAs* through each type of proposed principles. The best results from each instance that we have found do not obviously conclude that *Round-Robin & Interleaved* principle is better than the other principles in term of achieving the best results and always returns the least penalty values, even it tends to be so. Some best results are produced by *First-In-First-Out & Sequential*, *First-In-First-Out & Interleaved* and *Round-Robin & Sequential* principles for some instances. For the fairness issue, the sorted standard deviation values of each instance show that the standard deviation values of *Round-Robin & Interleaved* and *Round-Robin & Sequential* principles are less than the standard deviation values of *First-In-First-Out & Sequential* and *First-In-First-Out & Interleaved*. That means the different of average of satisfied values among *YPAs* in system which

applied the *Round-Robin* algorithm over the controlled loop of *RA* is less than the different of average of satisfied values among *YPAs* which applied *First-In-First-Out* algorithm. Therefore, by appling the *Round-Robin* algorithm, each agent gets more chances to satisfy its needs. Moreover, the increasing numbers of YPAs in system lead to a decrease in the effectiveness of constraint achievement in every principle. *Round-Robin & Sequential* and *Round-Robin & Interleaved* seem to be able to achieve the constraints better than the *First-In-First-Out & Sequential* principle for the same number of *YPAs*, while the *First-In-First-Out & Interleaved* principle tends to be less effective in solving the problem, even if the number of *YPAs* in the system is few.

Although the penalty values of the best results from our proposed principles are higher than the most successful reference results, our model is able to solve the university course timetabling problem. The proposed model can achieve all essential constraints and most of the needs of desirable constraints except for the isolated-lecture constraint, for which our model still produced quite high penalty values when comparing with the other constraints. If we consider this in term of the percentage of successes, the isolated lectures have been avoided for more than 75% of total lectures on average, while the minimum working days requirement also has been achieved for more than 98% and the other constraints have been one hundred percent achieved.

Importantly, there are two threats that could affect the quality of the result which is produced by the proposed model:

- *External threats* are results from randomly initial ordered *YPAs.*
- *Internal threats* are results from the defined information in the representative scenario.

9.2 Novel Contributions

While software agents have been previously proposed as a means of solving constraint-satisfaction problems such as timetabling, the major contribution of this research has been to investigate how an agent model can be most effectively organized for this task, and also to compare its effectiveness with both each other principles and other approaches.

We have investigated four principles that provide different ways of organizing the timetabling processes in the real world, with each participant making its decision influenced by its own needs as evident from the circumstances it meets at a given moment. Negotiation

and collaboration are also taken into account in protocols designed for solving conflicts and collaborating among the participants in order to reach the best solution. The results should help narrow the gap between theory and practice. When the timetable is being organized, the agent strategy is based on the principle of matching a course with the most preferred timeslots and allocation to the best-fit rooms whenever the agent gets the opportunity to organize its subjects. This design structure therefore supports a set of complicated constraints. The overlap problem over shared subjects has also been considered while each subject is being organized; therefore, a procedure of inter-agent negotiation to solve related conflicts should be unnecessary.
A novel feature of the model is that it seeks to allocate rooms fairly by applying a *Round-Robin* algorithm in the automated timetabling mechanism in order to ensure that each agent gets an equal chance to access the resources. Hence it has explored a new model for managing university course timetabling, by considering not only constraints but also the issue of fairness of allocation when the timetable is implemented.

9.3 Future Work

Further work is needed on these principles, and some questions need to be explored in order to improve the use of agents for university course timetabling research field. We suggest investigations of the following issues are needed.

1. The experimental results show that as the penalty value of the isolated-lecture constraint used in our principles is evidently high, that means the algorithms for avoiding isolated date-time slots needs to be improved.

2. There is a need to do more intensive analysis into the differences in output across the set of instances, in order to find out which factors or characteristics of the instances cause these differences. This should aid with matching principles to particular timetable characteristics.

3. To add more real world constraints to the model. One idea is to organize desirable constraints by applying an ordinal scale (must, should, could, would like) seems to increase more flexibility and avoid the clash over the shared courses that means the students are able to take more classes. The preferable date-time slots from the need of lecturers and the shortest pathways (distance between any two consecutive lectures) within the university could also be extended and implemented in the

model. The more constraints the model can achieve, the much closer it comes to providing a response to the needs from the real world.

4. Universities in the real world always face problems when there are increasing/decreasing numbers of students in courses, or with situations where some courses need to be cancelled after the timetable for those courses have been organized. Using an agents model, such adjustments might be achieved by focusing on the appropriate YPAs when re-running the model.

Bibliography

Aarts E H L and Korst J 1989 *Simulated Annealing and Boltzmann Machines* John Wiley & Sons.

Abdennadher S and Marte M 2000 *University Course Timetabling Using Constraint Handling Rules, Applied Artificial Intelligence* **14**, 311–326.

Aknine S, Pinson S and Shakun M F 2004 *An Extended Multi-Agent Negotiation Protocol, Autonomous Agents and Multi-Agent Systems* **8**, 5–45.

Al-Maqtari S, Abdulrab H and Babkin E 2009 *in* 'A new model for solution of complex distributed constrained problems' IEEE/ACS International Conference on Computer Systems and Applications Rabat, Morocco.

Al-Maqtari S, Abdulrab H and Nosary A 2006 Emergent Properties in Natural and Artificial Dynamical Systems, Understanding Complex Systems Springer Berlin / Heidelberg chapter Constraint Programming and Multi-Agent System Mixing Approach for Agricultural Decision Support System.

Al-Yakoob S M and Sherali H D 2005 *Mathematical programming models and algorithms for a class–faculty assignment problem, European Journal of Operational Research* **173**, 488–507.

Apt K R 2003 *Principles of Constraint Programming* Cambridge University Press.

Arntzen H and Lokketangen A 2003 *in* 'A tabu search heuristic for a university timetabling problem' the 5th Metaheuristics International Conference Kyoto Japan.

Bellifemine F L, Caire G and Greenwood D 2007 *Developing Multi-agent Systems with JADE* John Wilely & Sons.

Berwick B 2008 'Solving CSPs'. Date accessed:April 13,2010.
URL: *http://web.mit.edu/6.034/wwwbob/constraint.pdf*

Binmore K 1991 *Fun and Games: A Text on Game Theory* D.C. Heath.

Bordini R H, Hubner J F and Wooldridge M 2007 *Programming multi-agent systems in AgentSpeak using Jason* John Wiley and Sons Ltd.

Boyd S and Mattingley J 2007 Branch and Bound Methods Technical report Stanford University.

Burke E, de Werra D, de Lausanne and Kingston J 2004 *Applications to timetabling* Chapman Hall/CRC Press.

Burke E, Elliman D and Weare R 1994 *in* 'A Genetic Algorithm Based University Timetabling System' the 2nd East-West International Conference on Computer Technologies in Education Crimea Ukraine.

Burke E, Jackson K, Kingston J and Weare R 1997 *Automated University Timetabling: The State of the Art, The Computer Journal* **40**, 565–571.

Burke E K and Petrovic S 2002 *Recent Research Directions in Automated Timetabling, European Journal of Operational Research* **140**, 266–280.

Busetta P, Ronnquist R, Hodgson A and Lucas A 1999 'Jack Intelligent Agents - Components for Intelligent Agents in Java'. Date Accessed:Oct 2011.
URL: *http://www.agentlink.org/newsletter/2/newsletter2.pdf*

Carter M 2000 *in* E Burke and W Erben, eds, 'A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo' The 3rd International Conference of Practice and Theory of Automated Timetabling Springer.

Carter M W and Laporte G 1998 Vol. 1408 of *Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science* Springer chapter Recent Developments in Practical Course Timetabling, pp. 9–19.

Causmaecker P D, Demeester P and Berghe G V 2006 *in* 'Evaluation of the University Course Timetabling Problem with the Linear Numberings Method' The 25th Workshop of the UK PLANNING AND SCHEDULING Nottingham, UK.

Causmaecker P D, Demeester P, Lu Y and Berghe G V 2002 *in* 'Agent Technology for Timetabling' the 4th International Conference on Practice and Theory of Automated Timetabling.

Causmaecker P D, Ouelhadj D and Berghe G V 2003 *in* 'Agents in Timetabling Problems' The 1st Multidisciplinary International Conference on Scheduling pp. 67–71.

Cernuzzi L, Juan T, Sterling L and Zambonelli F 2004 Methodologies and Software Engineering for Agent Systems chapter The Gaia Methodology Basic Concepts and Extensions, pp. 69–88.

Chevaleyre Y, Dunne P E, Lang U E J, Lemaitre M, Maudet N, Padget J, Phelps S, Aguilar J A R and Sousa P 2006 *Multiagent Resource Allocation, Informatica* **30**, 3–31.

Clausen J 1999 Branch and Bound Algorithms - by Principles and Examples Technical report Department of Computer Science, University of Copenhagen Universitetsparken 1, DK2100 Copenhagen, Denmark.

Cowling P I, Ouelhadj D and Petrovic S 2003 *A multi-agent architecture for dynamic scheduling of steel hot rolling, Journal of Intelligent Manufacturing* **14**, 457–470.

Dastani M 2008 *2APL A Practical Agent Programming Language, International Journal of Autonomous Agents and Multi-Agent Systems* **16**(3), 214–248.

Davis L 1991 *Handbook of genetic algorithms* Van Nostrand Reinhold.

Elmohamed M S and Fox G 1997 *in* 'A Comparison of Annealing Techniques for Academic Course Scheduling' the Practice and Theory of Automated Timetabling Toronto Canada.

Endriss U, Maudet N, Sadri F and Toni F 2003 *in* 'On Optimal Outcomes of Negotiations over Resources' Autonomous Agents and Multiagent Systems ACM Press.

Ernst A, Singh G and Weiskircher R 2008 *in* 'Scheduling Meetings at Trade Events with Complex Preferences' the 18th International Conference on Automated Planning and Scheduling Sydney Australia.

Even S, Itai A and Shamir A 1975 *in* 'On the complexity of timetable and multi-commodity flow problems' the 16th Annual Symposium on Foundations of Computer Science IEEE Computer Society.

Fudenberg D and Tirole J 1991 *Game Theory* MIT Press.

Galitsky B 1999 *in* 'Agents with adjustable autonomy for scheduling in the competitive environment' Agents with Adjustable Autonomy AAAI Spring Symposium Technical Report pp. 41–49.

Garey M R and Johnson D S 1979 *Computers and Intractability- A guide to NP-completeness.* W.H. Freeman and Company.

Gaspero L D, McCollum B and Schaerf A 2007 The Second International Timetabling Competition (ITC-2007):Curriculum-Based Course Timetabling (Track 3) Technical report.

Gaspero L D, Mizzaro S and Schaerf A 2004 *in* 'A MultiAgent Architecture for Distributed Course Timetabling' The 5th International Conference on the Practice and Theory of Automated Timetabling Pittsburg, Pennsylvania USA pp. 471–474.

Gaspero L D and Schaerf A 2000 *in* L. D Gaspero and A Schaerf, eds, 'Tabu Search Techniques for Examination Timetabling' the Third International Conference on Practice and Theory of Automated Timetabling III Germany.

Georgeff M P and Lansky A L 1987 *in* 'Reactive Reasoning and Planning' the Sixth National Conference on Artificial Intelligence MIT Press.

Gibbons R 1958 *A primer in game theory* Prentice Hall.

Golfarelli M, Maio D and Rizzi S 1997 A Task–Swap Negotiation Protocol Based on the Contract Net Paradigm Technical report University of Bologna.

Hertz A 1992 *Finding a feasible course schedule using Tabu search, Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science* **35**, 255–270.

Hertz A, Taillard E and de Werra D 1993 'A Tutorial On Tabu Search'. 661–673.

Huber M J 2001 'JAM Agents in a Nutshell'. Date Accessed: Oct 2011.
URL: *www.marcush.net/IRS/Jam/Jam-man-01Nov01.doc*

Jennings N 2000 *On agent-base software engineering, Artificial Intelligence* **117**, 277–296.

Kaplansky E and Meisels A 2004 *in* 'Negotiation among scheduling agents for distributed timetabling' the 5th International Conference on the Practice and Theory of Automated Timetabling.

Kinny D 1993 The distributed multi-agent reasoning system architecture and language specification Technical report Australian Artificial Intelligence Institute Melbourne, Australia.

Kirkpatrick S, Gelatt C D, Jr and Vecchi M P 1983 *Optimization by simulated annealing, Science* **220**, 671–680.

Lewicki R, Barry B and Saunders D M 2010 *Negotiation* McGraw-Hill Higher Education. ISBN: 0073381209.

Lewis R 2007 *A Survey of Metaheuristic–based Techniques for University Timetabling Problems, OR Spectrum* **vol 30 (1)**, 167–190.

Liu J, Jing H and Tang Y Y 2002 *Multi-agent oriented constriant satisfaction,Artificial Intelligence* **136**, 101–144.

Matuszek 2009 'Backtracking'. April 12th, 2010.
URL: *www.cis.upenn.edu/ matuszek/cit594-2009/Lectures/35-backtracking.ppt*

McCollum B 2007 *in* 'A Perspective on Bridging the Gap between Research and Practice in University Timetabling, Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science' Vol. 3867 Springer pp. 3–23.

Meisels A, Kuflik T and Gudes E 1991 *Limited–resource scheduling by generalized rule–based system, Knowledge–Based System* **4**, 215–224.

Michalewicz Z 1994 *Genetic Algorithms + Data Structure = Evoluation Programs* second extended edition edn Springer-Verlag.

M.Nandhini and S.Kanmani 2009 *in* 'Implementation of class timetabling using multi agents' International Conference on Intelligent Agent & Multi-Agent Systems Chennai.

Montana D, Herrero J, Vidaver G and Bidwell G 2000 *A multiagent society for military transportation scheduling, Journal of Scheduling* **3**, 225–246.

Murray K and Rudova H 2007 *in* 'Modeling and solution of a complex university course timetabling problem' the 6th international conference on Practice and theory of automated timetabling VI pp. 189–210.

Murthy S, Akkiraju R, Rachlin J and Wu F 1997 *in* 'Agent-Based Cooperative Scheduling' AAAI Workshop on Constraints and Agents pp. 112–117.

Norberciak M 2006 *in* 'Universal Method for Solving Timetabling Problems Based on Evolutionary Approach' the International Multiconference on Computer Science and Information Technology Wisla, Poland.

Oates B J 2006 *Researching Information Systems and Computing* SAGE Publications.

Oon W C and Lim A 2002 *in* 'Multi-Player Game Approach to Solving Multi-Entity Problems' Eighteenth national conference on Artificial intelligence Edmonton, Alberta, Canada pp. 961–962.

Oprea M 2007 *MAS—UP–UCT: A Multi-Agent System for University Course Timetable Scheduling, International Journal of Computers, Communications&Control* **2**, 94–102.

Ozcan E and Alkan A 2002 *in* 'Timetabling using a Steady State Genetic Algorithm' the 4th International Conference on the Practice and Theory of Automated Timetabling Gent, Belgium.

Paechter B, Rankin R, Cumming A and Fogarty T C 1998 *in* 'Timetabling the Classes of an Entire University with an Evolutionary Algorithm' the 5th International Conference on Parallel Problem Solving from Nature Amsterdam, Netherlands.

Parkes D C and Ungar L H 2001 *in* 'An Auction-Based Method for Decentralized Train Scheduling' the 5th International Conference on Autonomous Agents pp. 43–50.

Pnueli A 1986 *in* 'Specification and Development of Reactive Systems' Information Processing Amsterdam Holland pp. 845–858.

Rao A S 1996 *in* 'AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language' Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-96).

Rao A S and Georgeff M P 1991 *in* 'Modeling rational agents within a BDI-architecture' the Second International Conference on Principles of Knowledge Representation and Reasoning Morgan Kaufmann Publishers San Mateo, Ca.

Rich D C 1995 *in* 'A smart genetic algorithm for university timetabling' Practice and Theory of Automated Timetabling Edinburgh Scotland.

Rosenschein J S and Zlotkin G 1994 *Rules of Encounter Designing Conventions for Automated Negotiation among Computers* The MIT Press Cambridge, Massachusetts London, England.

Rudova H and Matyska L 2000 *in* 'Constraint-based Timetabling with Student Schedules' the third International Conference on Practice and Theory of Automated Timetabling Constance Germany.

Sandholm T 1993 *in* 'An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations' The 11th National Conference on Artificial Intelligence AAAI Press.

Sandholm T 1998 *in* 'Contract types for satisfacing task allocation: I theoretical results, AAAI Spring Symposium: Satisficing Models'.

Sandholm T W 1999 Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence The MIT Press chapter Distributed Rational Decision Making.

Schaerf A 1999 *A Survey of Automated Timetabling,Artificial Intelligence Review* **13**, 87–127.

Selim S M 1988 *Split Vertices in Vertex Colouring and Their Application in Developing a Solution to the Faculty Timetable Problem, The Computer Journal* **31**, 76–82.

Sen S 1997 *Developing an Automated Distributed Meeting Scheduler, Intelligent Systems and Their Applications* **12**.

Sheau H, Safaai-Deris and Hashim S Z M 2009 *in* 'Investigating Constraint-Based Reasoning for University Timetabling Problem' Vol. 1 Proceedings of the International MultiConference of Engineers and Computer Scientists Hong Kong.

Sierra C, Faratin P and Jennings N R 1997 *in* 'A Service-Oriented Negotiation Model between Autonomous Agents' Vol. 1237 the eighth European Workshop on Modeling Autonomous Agents in a Multi-Agent World Springer pp. 17–35.

Smith K A, a Abramson D and Duke D 2003 *Hopfield neural networks for timetabling: formulations, methods, and comparative results, Computers and Industrial Engineering* **44**, 283–305.

Smith R G 1980 *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, IEEE Trans. Computers* **29**, 1104–1113.

Socha K, Knowles J and Sampels M 2002 *in* 'A MAX–MIN Ant System for the University Course Timetabling Problem' the third International Workshop on Ant Algorithms Brussels Belgium.

Sousa P, Ramos C and Neves J 2003 *The Fabricare scheduling prototype suite: Agent interaction and knowledge base, Journal of Intelligent Manufacturing* **14**, 441–455.

Strnad D and Guid N 2007 *A multi-agent system for university course timetabling, Applied Artificial Intelligence* **21**, 137–153.

Thompson J M and Dowsland K A 1998 *A robust simulated annealing based examination timetabling system, Computers & Operations Research* **25**, 637–648.

Torres L, Palacios O, Pacheco R and Cortes A 2006 Automated University Timetabling Technical report Department of Computer Science University of Texas at El Paso 500 W. University, El Paso, TX 79968, USA.

van Hoeve W J and Katriel I 2006 Handbook of Constraint Programming Elsevier chapter Global Constraints.

Vermeulen I, Bohte S, Elkhuizen S, Bakker P and Poutr H L 2008 *in* 'Decentralized Online Scheduling of Combination-Appointments in Hospitals' the 18th International Conference on Automated Planning and Scheduling Sydney Australia.

Weiss G 2000 *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* The MIT Press.

Werra D 1985 *An introduction to timetabling, European Journal of Operational Research* **19**, 151–162.

Woods D and Trenaman A 1999 *in* 'Simultaneous satisfaction of hard and soft timetable constraints for a university department using evolutionary timetabling' Artificial Intelligence & Cognitive Science 1999 Cork, Ireland pp. 1–7.

Wooldridge M 2002 *An Introduction to MultiAgent Systems* John Wiley & Sons, Ltd.

Wooldridge M and Jennings N 1995 *Intelligent Agents: Theory and Practice, Knowledge Engineering Review* **10**(2).

Yang Y and Paranjape R 2011 *A multi-agent system for course timetabling, Intelligent Decision Technology* **3**, 113–131.

Yang Y, Paranjape R and Benedicenti L 2006 *in* 'An agent based general solution model for the course timetabling problem' the fifth international joint conference on Autonomous agents and multiagent systems pp. 1430–1432.

Yin R K 2009 *Case Study Research Design and Methods* SAGE.