Meta-heuristic Techniques for Grid Resource Management

A Thesis submitted
in partial fulfillment of the requirements for the award of degree of
DOCTOR OF PHILOSOPHY

by
Bhupinder Singh
(9040352)

Under the guidance of

Dr. Seema Bawa
Professor
Computer Science and Engineering Department
Thapar University, Patiala -147004

Computer Science and Engineering Department
Thapar University, Patiala – 147 004, INDIA

September 2009

Contents

Contents ..... **i**
Certificate ..... **vii**
Acknowledgements ..... **viii**
List of Figures ..... **xi**
List of Tables ..... **xiii**
Abstract ..... **xiv**

1 Introduction **1**
1.1 Grid Computing 1
1.1.1 Benefits for Grid Computing 3
1.1.2 Types of Grid 5
1.2 Resource Management Systems for Computational Grids 8
1.2.1 Grid Resource Discovery 9
1.2.2 Grid Resource Discovery Models 11
1.2.3 Grid Resource Scheduling 11
1.2.4 Grid Resource Scheduling Models 13
1.3 Meta-Heuristic Techniques 15
1.3.1 Simulated Annealing (SA) 17
1.3.2 Genetic Algorithms (GA) 18
1.3.3 Ant Colony Optimization (ACO) 19

1.4 Thesis Objectives and Contributions 20
1.4.1 Objectives of the Thesis 21
1.4.2 Contributions 21
1.5 Thesis Organization 23
1.6 Summary 25

2 Literature Survey 26
2.1 Introduction 26
2.2 Grid Resource Management System (GRMS) 27
2.2.1 Grid Resource Management System Taxonomy 28
2.3 Grid Resource Management Systems Survey 31
2.3.1 Globus 31
2.3.2 Condor 33
2.3.3 Condor-G 34
2.3.4 Nimrod/G 35
2.3.5 Legion 37
2.3.6 Alchemi 38
2.3.7 GridWay 40
2.3.8 Gridbus Resource Broker 41
2.4 Comparison of Grid Resource Management Systems 42
2.5 Grid Scheduling Heuristics 43
2.5.1 Opportunistic Load Balancing (OLB) 43
2.5.2 Minimum Execution Time (MET) 44
2.5.3 Minimum Completion Time (MCT) 44
2.5.4 Min-min 44
2.5.5 Max-min 44

2.5.6 Duplex 45
2.5.7 Genetic Algorithms (GAs) 45
2.5.8 Simulated Annealing (SA) 45
2.6 Grid Resource Discovery Algorithms 45
2.6.1 Flooding Algorithm 46
2.6.2 Swamping Algorithm 46
2.6.3 Random Pointer Jump Algorithm 46
2.6.4 Name Dropper Algorithm 47
2.6.5 Time-To-Live based Reservation Algorithm 47
2.6.6 Discovering Intermittently Available Resources (DIAR) Algorithm 47
2.6.7 Absorption (Randomized Resource Discovery) Algorithm 48
2.7 Summary 48

3 Proposed Algorithms for Grid Resource Scheduling 49
3.1 Introduction 49
3.2 Problem Formulation 50
3.2.1 Objective / Fitness Function 51
3.3 Sexual GA based Resource Scheduling Algorithm (SGASchedule) 52
3.3.1 SexualGA 52
3.3.2 Chromosome Representation 53
3.3.3 Initial Population 53
3.3.4 Selection 54
3.3.5 Crossover and Mutation 54
3.3.6 Exit criteria 54
3.3.7 Building the new population 55
3.3.8 Algorithm (Pseudo Code) 55

3.4 ACO based Resource Scheduling Algorithm (ACOSchedule) 56

3.4.1 Pheromone Trail Definition 57

3.4.2 Heuristic Information 57

3.4.3 Pheromone Trail Updating 57

3.4.4 Building the solution 58

3.4.5 Optimizing ACO by Genetic Operators 58

3.4.6 Algorithm (Pseudo Code) 60

3.5 Hybrid Genetic Simulated Annealing based Grid Scheduling Algorithm (HybridGSA) 62

3.5.1 Initial Temperature and Annealing Schedule 63

3.5.2 Crossover 63

3.5.3 Mutation 64

3.5.4 Algorithm (Pseudo Code) 64

3.6 Summary 66

4 Proposed Algorithm for Grid Resource Discovery 67

4.1 Introduction 67

4.2 Problem formulation 68

4.3 Mobile Agents 70

4.4 Proposed Algorithm for Resource Discovery based on ACO guided Mobile Agents (ACORD) 70

4.4.1 Fitness function 71

4.4.2 Pheromone Trail Definition 71

4.4.3 Heuristic Information 72

4.4.4 Pheromone Trail Updating 72

4.4.5 Building the Solution 72

4.4.6 Exit Criteria 73

4.5 Algorithm to be executed by Query Agent 73

4.5.1 Algorithms (Pseudo Code) 75

4.6 Advantages of proposed ACO based Resource Discovery Algorithm (ACORD) 76

4.7 Summary 77

5 Design and Implementation of Proposed Algorithms 78

5.1 Introduction 78

5.2 GridSim Toolkit 78

5.2.1 Resource modeling 80

5.2.2 Application Modeling 80

5.2.3 Steps for Simulating Application Scheduling 80

5.3 Implementation details of Grid Simulator based on GridSim Tool kit 81

5.3.1 Components of Grid Simulator 82

5.3.2 Class Diagram 82

5.3.3 Communication Mechanism 84

5.3.4 Pseudo Code for Submission System Flow 86

5.3.5 Pseudo Code for Scheduler Flow 86

5.3.6 Extensibility 86

5.4 Grid Test Bed 87

5.4.1 Web based User Interface 88

5.4.2 Information flow in Grid Test Bed 90

5.4.3 Simulation Model used in Grid Test Bed 91

5.4.4 Mobile Agent based Framework 93

5.5 Summary 93

6 Experimental Details and Results.....94
6.1 Introduction.....94
6.2 Simulation Model.....95
6.2.1 Constructing CT matrix.....95
6.2.2 Pseudo Code.....96
6.2.3 Achieving Heterogeneity.....97
6.2.4 Algorithm parameters used in experiments.....98
6.3 Experimental Results for proposed scheduling algorithms using Grid Test Bed.....99
6.3.1 Minimizing makespan.....100
6.3.2 Minimizing Cumulative Time delay (T_{delay}).....102
6.4 Experimental Results on GridSim based Grid Simulator.....105
6.5 Experimental results for Proposed Resource Discovery Algorithm.....107
6.6 Summary.....110

7 Conclusions & Future Scope.....111
7.1 Conclusions.....111
7.2 Future Scope of work.....113
7.2.1 Supporting Parallelization of Grid Scheduling Algorithms.....113
7.2.2 Data intensive Programming and Scheduling Framework.....114
7.2.3 Scheduling Framework for Real Time applications.....114
7.2.4 Grid based Generic Mathematical / Software libraries.....115
7.2.5 Grid based DWDM Network Design and Planning Tool.....115

References.....116

Publications.....129

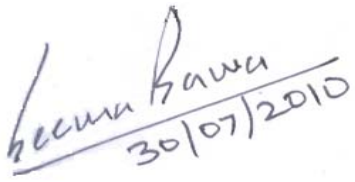
Certificate

I hereby certify that the work which is being presented in this thesis entitled "***Meta-heuristic Techniques for Grid Resource Management***", submitted in partial fulfillment of the requirements for the award of degree of ***Doctor of Philosophy*** in ***Computer Science and Engineering Department*** of ***Thapar University, Patiala,*** is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa and refers to the work done by other researchers which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Bhupinder Singh)
Reg. No. 9040352

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.



(Dr. Seema Bawa)
Professor,
Computer Science & Engineering Department (CSED),
Thapar University, Patiala 147 004, Punjab, INDIA

Acknowledgements

First of all, I express my gratitude to that Almighty, Who blessed me with the zeal and enthusiasm to complete this work successfully in spite of hard commitments at my workplace.

This thesis arose in part out of years of research and I have interacted with a great number of people whose contribution in assorted ways to the research and the making of the thesis deserved special mention. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

I would like to start by conveying my sincere thanks to my supervisor Dr. Seema Bawa, Professor, Computer Science and Engineering Department, Thapar University, Patiala, for her supervision, advice, and guidance from the very early stage of this research as well as giving me extraordinary experiences through out the work. Above all and the most needed, she provided me unflinching encouragement and support in various ways. Her truly scientist intuition has made her as a constant oasis of ideas and passions in technology, which exceptionally inspire and enrich my growth as a student, a researcher and as a professional. I am indebted to her more than she knows. I feel it is a privilege to have the opportunity to work under her supervision.

Many thanks go in particular to Dr. Rajesh K. Bhatia, Head, Computer Science and Engineering Department and Dr. Maninder Singh, Associate Professor, Computer Science and Engineering Department, who always shown their willingness

to help and mentor me, always encouraged me to keep going with work and always advised me with their invaluable suggestions.

I gratefully pay my sincere thanks to the members of Doctoral Committee, Dr. R. K. Sharma, Professor & Dean of Academic Affairs, Dr. R.S. Kaler, Professor, Electronics and Communications Engineering Department and Dr. Sushil Mittal, Professor & Dean of Research and Sponsored Projects for their observations, constructive criticism and valuable suggestions which helped me a lot in presenting the results and shaping this thesis.

I would like to thank all faculty and staff members of Computer Science and Engineering Department of university for providing help and resources whenever required. Many thanks to administrative and academic staff members of Thapar University, who have been kind enough to advise and help in their respective roles.

I am also grateful to my manager (at my current workplace), Vikas Batra, Manager, P & T, Ciena India Pvt. Ltd., for providing me facilities to keep my research work going on along with the job. I would like to convey my thanks to my seniors at my previous workplaces, Amit Sadana, Team lead, Quark Media House India Pvt. Ltd. and Basudev Mohanty, Senior Manager, Siemens Information Systems Ltd. for their support and encouragement while working with them.

This research has profited from the friendship, advice, encouragement and support of several remarkable people. I am indebted to my dear friend Umesh Gupta, Senior Software Engineer, Cadence Design Systems Pvt. Ltd., for continual counsel, willingness to listen and his valuable suggestions during this tenure. My special thanks to Dr. Sumanta Gupta, Senior Engineer, Ciena India Pvt. Ltd., Dr. Nitin Goel Lead Engineer, Ciena India Pvt. Ltd., Dr. Anju Sharma, Lecturer, Thapar University

and Dr. Sarbjeet Singh, Lecturer, Panjab University, who always shown their willingness to support and helped me, with a smiling face, as and when required.

Last but not the least; I would like to thank my family. My parents deserve a special mention for their inseparable support and prayers. Thanks to my loving sister, Jaswinder Kaur, for being supportive and caring. Words fail me to express my appreciation to my Mama ji, Late S. Aya Singh, who always believed in my capabilities more than anyone else. Also, my sincere thanks to my cousin, Bhupinder Singh, Lead Architect, GE Healthcare, Washington, for his continual encouragement and appreciation.

Finally, I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

Bhupinder Singh

September 2009.

List of Figures

Figure 1.1	Virtual Organization (Simpler View) [5] ……..……	4
Figure 1.2	View of Data [5] ……	6
Figure 1.3	View of Inter-grid spanning over the globe [5] …….	7
Figure 1.4	High level view of Grid Resource Management System ……..	9
Figure 1.5	Common Grid Scheduler Architecture ……..	11
Figure 1.6	Local Optimum Problem ……...	16
Figure 2.1	The system overview of Globus ToolKit ……..	32
Figure 2.2	Condor Architecture ……..	33
Figure 2.3	Layered Architecture of Nimrod/G ……...	36
Figure 2.4	Communication in Legion objects through protocol stack ……...	37
Figure 2.5	Alchemi architecture and communication between its components..	39
Figure 2.6	GridWay framework model for self adapting applications [71]……	40
Figure 2.7	Gridbus Broker Architecture [72] ……	41
Figure 3.1	Chromosome Representation ……	53
Figure 3.2	Schedule *S* before applying Mutation operator (Makespan = 7) …..	60
Figure 3.3	Schedule S’ after applying Mutation operator (Makespan = 5) ……	60
Figure 4.1	High level view with Peer Contact Nodes ……	69
Figure 4.2	Different states of a Query Agent …….	74

Figure 5.1	Flow diagram in GridSim based simulations ……	79
Figure 5.2	Class diagram of a Grid Simulator ……	83
Figure 5.3	Screen shot of development environment for Grid Simulator ……..	84
Figure 5.4	Communication among Grid Simulator components ……	85
Figure 5.5	High level view of Grid Test Bed used for experiments ……...	87
Figure 5.6	Grid User Login Screen …….	88
Figure 5.7	Register Resource Screen …….…….	89
Figure 5.8	Algorithm Parameters configuration screen ……..	89
Figure 5.9	Sequence diagram showing information flow in Grid Test Bed …..	91
Figure 5.10	Screen shot of development environment in Grid Test Bed …….	92
Figure 6.1	Comparison results for Inconsistent, High Machine heterogeneity *CT* matrix ……..	100
Figure 6.2	Comparison results for Inconsistent, Low Machine heterogeneity *CT* matrix ……..	101
Figure 6.3	Comparison results for Partially Consistent, High Machine heterogeneity *CT* matrix ……	101
Figure 6.4	Comparison results for Partially Consistent, Low Machine heterogeneity *CT* matrix ……	102
Figure 6.5	Comparison of cumulative time delay (T_{delay}) for proposed algorithms for inconsistent, high machine heterogeneity ……..	103
Figure 6.6	Comparison of cumulative time delay (T_{delay}) for proposed algorithms for partially consistent high machine heterogeneity …..	104
Figure 6.7	Comparison of SGASchedule Vs. Nimrod/G ……...	106
Figure 6.8	User Requests Profile ……	107
Figure 6.9	Number of Ants Vs. Query Hits ……	108
Figure 6.10	Network Size Vs. Query Hits …….....	109

List of Tables

Table 2.1	Comparison of different Grid Systems …….	43
Table 3.1	$t_{complete}$ of j_1, j_2, j_3, j_4 on p_1, p_2 ……...	60
Table 6.1	*4×4* excerpt from inconsistent, high task heterogeneity and high machine heterogeneity matrix ……..	98
Table 6.2	*4×4* excerpt from partially consistent, high task heterogeneity and high machine heterogeneity matrix …….	98
Table 6.3	Parameters and their values used in experiments …….	98
Table 6.4	Makespan results obtained from experimental results for proposed algorithms …….	100
Table 6.5	Comparison results for minimizing T_{delay} ……..……	102

Abstract

Grid computing, inspired by electrical power grid, is an emerging trend for making easy access to computing resources. It has evolved into an important discipline through an increased focus on resource sharing, co-ordination, manageability and high performance. Computational Grid is a distributed and heterogeneous computing system that combines open, shared, geographically distributed and heterogeneous resources to achieve high computational performance. Its main objective is to solve computationally hard problems which otherwise can not be solved by single CPU. This extremely high computing power is achieved by optimal utilization of geographically distributed heterogeneous resources which are lying idle. A computational grid may span from a few systems in a Local Area Network to a World Wide Grid (WWG) with million of computers connected via internet. Resource management in such a large system is complex and becomes more challenging as the size of the grid grows. Finding a set of matching resources and mapping a set of tasks on to set of resources in this environment is compute intensive problem. Dynamic nature of resources, different administrative policies and Quality of Service (QoS) requirements make the problem even tougher. Use of conventional methods becomes infeasible as the search space becomes very large. A new area of research, which makes use of meta-heuristic techniques to find the optimal or near optimal solution, tackles this problem. This thesis focuses on the use of meta-heuristic techniques like Genetic Algorithms (GA), Ants' Colony Optimization (ACO) and

Simulated Annealing (SA) and some hybrid method based on these, for optimizing resource discovery and resource scheduling / resource allocation in a grid resource management system. Three new algorithms for resource scheduling and one new algorithm for agent based resource discovery have been proposed, implemented and validated in this thesis.

Three grid resource scheduling algorithms named SGASchedule, ACOSchedule and HybridGSA has been proposed, implemented and validated in this thesis. SGASchedule is a GA based scheduling algorithm which used SexualGA based selection mechanism. In ACOSchedule, a set of artificial ants perform the job of grid resource scheduling and solutions obtained by Ants are further optimized by the use of Genetic Operators: Crossover and Mutation. HybridGSA is a hybrid algorithms based on SGASchedule and SA. It uses the SexualGA based selection mechanism to find the two parents, apply Genetic Operators: Crossover and Mutation and then selects or rejects new children based on the probability. This probability is computed using the temperature gradient properties of SA.

A new ACO based grid resource discovery algorithm, named ACORD, has also been proposed, implemented and validated in this thesis. This is an ACO guided Mobile Agents based resource discovery approach, where the mobile agents are used to simulate the ants and move across the network to performs the job of resource discovery. Pheromone trails deposited by ants provide a learning mechanism to guide mobile agents in resource discovery. Hence this algorithm has the benefits of both: Mobile Agents and ACO based learning mechanism.

1

Introduction

1.1 Grid Computing

Grid computing is an emerging computing paradigm which is inspired by the electrical power grid. Looking at the ease of use, pervasiveness and reliability of the electrical power grid, computer scientists too started exploring the design / development of an analogous infrastructure for wide-area parallel and distributed computing and data sharing. This resulted into the notion of computational power grid [1]. The motivation for computational Grids was initially driven by large-scale, resource (computational and data) intensive scientific applications that require more

resource than a single computer (PC, workstation, supercomputer, or cluster) could provide in a single administrative domain. Grid computing strives to aggregate diverse, heterogeneous, and geographically distributed and multiple domain spanning resources to provide a platform for transparent, secure, coordinated, and high-performance resource-sharing and problem solving [2]-[4].

Grids are becoming platforms for high-performance and distributed computing [3]. Grid computing can be thought of as an enhanced form of distributed computing. A grid benefits users by permitting them to access heterogeneous resources, such as machines, data, people and devices that are distributed geographically and organizationally. It benefits organizations by permitting them to offer unused resources on existing hardware and software. They allow users to execute compute intensive problems whose computational requirements cannot be satisfied by a single machine [4]. A computational grid environment behaves like a virtual organization consisting of distributed resources. A virtual organization is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place [4]. Grid computing has emerged as the new paradigm in distributed computing and has undergone significant developments over recent years.

Grid is type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost, user quality-of – self-service requirement. [2]

Foster, et.al. defined the grid problem as coordinated resource sharing and problem solving in dynamic multi-institutional, virtual organization [4]. The definition of grid conveys the following distinct dimensions of the grid:

"*Resource*" in a broad sense means data, computers, memory, scientific instruments, software, peripherals, network bandwidth etc.

"*Coordinated Sharing*" Resources together with their providers / consumers are clearly defined, and in that multiple resources may need to be organized in an integrated fashion to achieve various qualities of service. Achieving this coordination involves the establishment and enforcement of sharing agreements.

"*Dynamic Membership*" as participants may leave or join at any time.

1.1.1 Benefits for Grid Computing

The main objective of the grid computing is to create the illusion of a simple yet powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources [5]. Grid computing provides some of the potential benefits as listed below.

1.1.1.1 Full utilization of idle resources

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines are busy less than 5% of the time. In some organizations, even the server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage.

1.1.1.2 Parallel CPU capacity

The other attractive feature of the grid computing is massive parallel CPU capacity. This computing power is driving factor for evolution in industries like bio-medical field, financial modeling etc. along with scientific applications.

1.1.1.3 Virtual organizations / Resources

Grid computing enables the heterogeneous systems to collaborate by offering the important standards and hence creates the illusion of a large virtual computing system with a variety of virtual resources. The users of the grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger grid.

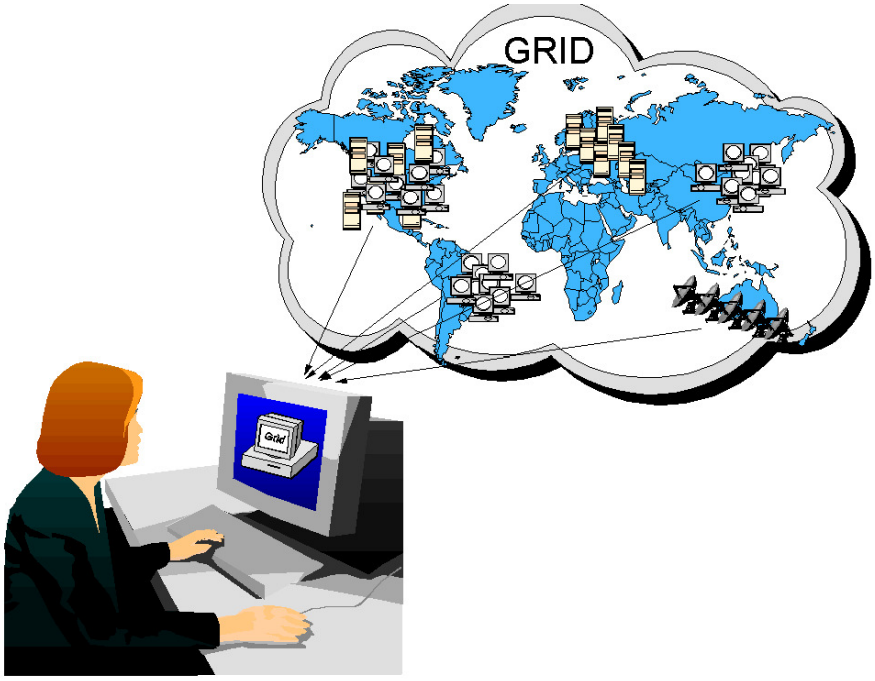


Figure 1.1 Virtual Organization (Simpler view) [5]

1.1.1.4 Load balancing

A grid unites a large number of resources contributed by individual machines into a greater total virtual resource. For grid enabled applications, the grid can offer a load balancing effect by scheduling grid jobs on machines with low utilization.

1.1.1.5 Reliability

Grid computing provides a software technology based approach to assure the reliable operation than using expensive hardware. Since the systems in the grid are

geographically dispersed so break down at one location leaves other parts of the grid unaffected. As soon as a failure is detected, Grid management software automatically resubmits the jobs to some other machines on the grid.

1.1.2 Types of Grid

A grid is a collection of machines, sometimes referred to as nodes, resources, members etc. They all contribute any combination of resources to the grid as a whole. Depending on the type of resource being shared a Grid can be broadly divided into two categories: Data Grid and Computational Grid.

1.1.2.1 Data Grid

The most common use of the grid is data storage. Data grid is an integrated view of data storage [6]. Every machine connected to the grid provides some quantity of storage for grid use. Storage can be primary or volatile memory or secondary memory using hard-disk or some other type of permanent storage media like magnetic disc or tape etc.

Secondary storage shared on the grid can be used in interesting ways to increase the capacity, performance and reliability of the data. Most of the grid systems use mountable networked file systems. Capacity can be increased by using the storage on multiple machines with a unifying file system. Any individual file or data base can span several storage devices and machines, eliminating maximum size restrictions often imposed by file systems shipped with operating systems. A unifying file system can also provide a single uniform name space for grid storage.

More advanced file systems on a grid can automatically duplicate sets of data, to provide redundancy for increased reliability and increased performance. Data striping can also be implemented by grid file systems. Similarly by implementing

journaling in a data grid, data can be recovered more reliably after certain kind of failures.

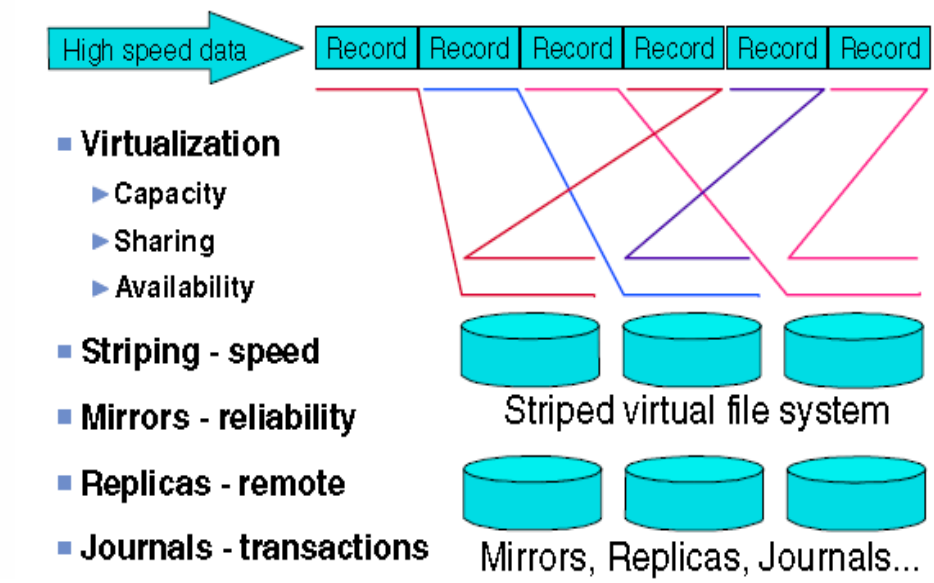


Figure 1.2: View of Data [5]

1.1.2.2 Computational Grid

Exploiting the computing cycles provided by processors is another use of Grid computing. Many scientific applications are compute intensive that require a lot of computing power to find out some optimal or sub-optimal solution. Computational grids aim to exploit the computing cycles of idle processors connected in the grid and provide a view of virtual supercomputing with number of processing elements working in parallel to provide the solution to a problem which is otherwise impossible to tackle with normal desktop computers. There are different ways to exploit the computing power of a grid resource.

- ✓ Run an existing application on an available machine on the grid rather than locally.
- ✓ To design an application to split its work so that different parts can be executed on different machines in parallel.

- ✓ To run an application that needs to be executed many times on different machines in the grid.

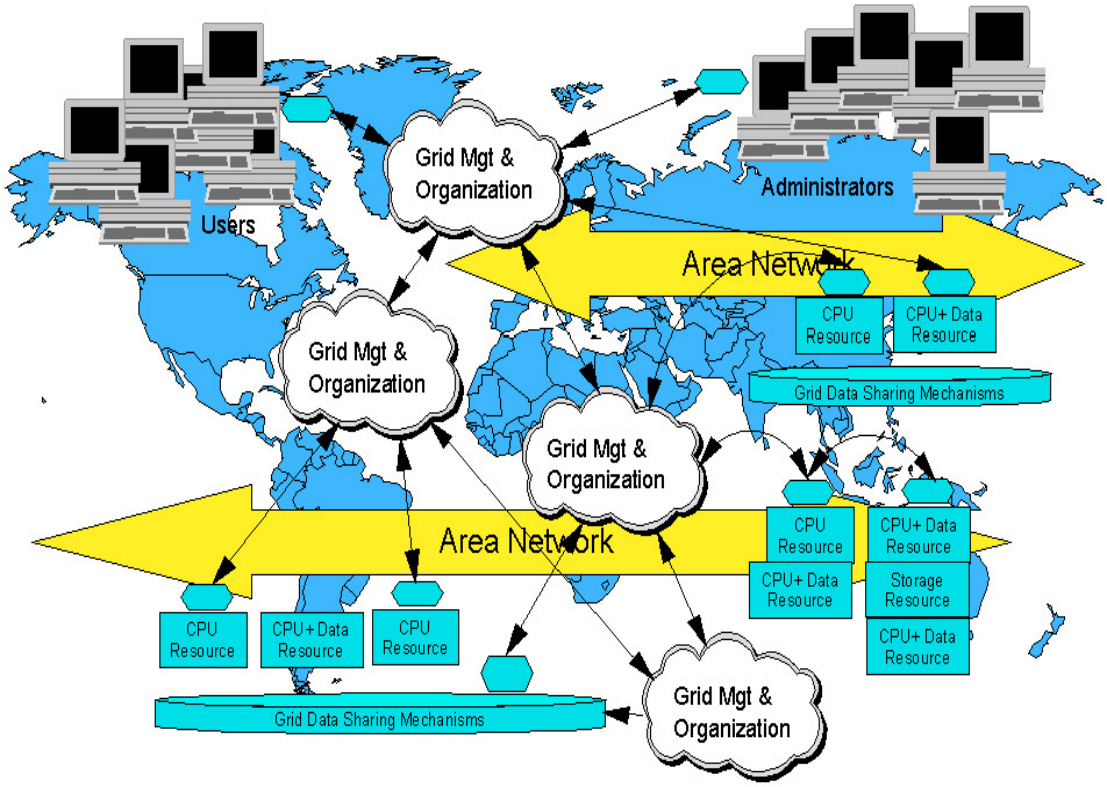


Figure 1.3: View of Inter-grid spanning over the globe. [5]

Detailed discussion about Inter-grid can be found in [7], computational grid is very similar to electric grid. When we require additional electricity we have to just plug into a power grid to access additional electricity on demand, similarly we plug into a computer grid to access additional computing power on demand using the cheapest possible resource. Basically, user submits a job to the grid through an interface on his computer that serves as a portal to the grid. Special grid management software accepts the job and breaks it down into hundreds or even thousands of independent tasks, then locates idle processors and distributes the tasks among them. Finally, it aggregates the works and spits out the results.

1.2 Resource Management Systems for Computational Grids

The theory behind the grid computing is not to buy more computational resources but to borrow the computing power from where it is not being used. The more are the resources the more complex is their management. Resource management in the computational grids deal with identifying requirements, matching resources to the applications, allocation of resources and scheduling and monitoring of the grid in a timely manner. Security and fault tolerance along with scheduling makes the process of resource management challenging for grid systems. The other factors like site autonomy and resource heterogeneity add more complexity to this system [8]-[11].

In the traditional resource management systems, the assumption is that they have complete control on the resource. Thus effective mechanisms and policies can be implemented for efficient use of that resource. But in Grid systems resources are distributed across separate administrative domains. This results in resource heterogeneity, differences in usage, scheduling policies, security mechanisms. Co-allocation of the resources also becomes complex due to site autonomy. Co-allocation is the problem of allocating resources in different sites to an application simultaneously. The Figure 1.4 shows the high level view of a Grid Resource management system. In a grid system, an end user submits to the management system the job to be executed along with some constraints like job execution deadline, the maximum cost of execution. The function of the resource management is to take the job specification and from it estimate the resource requirements like the number of processors required, the execution time, and memory required. After estimating the resource requirements RMS is responsible for discovering available resources [11] and selecting appropriate resources for job execution. Finally schedule the jobs on these resources by interacting with the local resource management system.

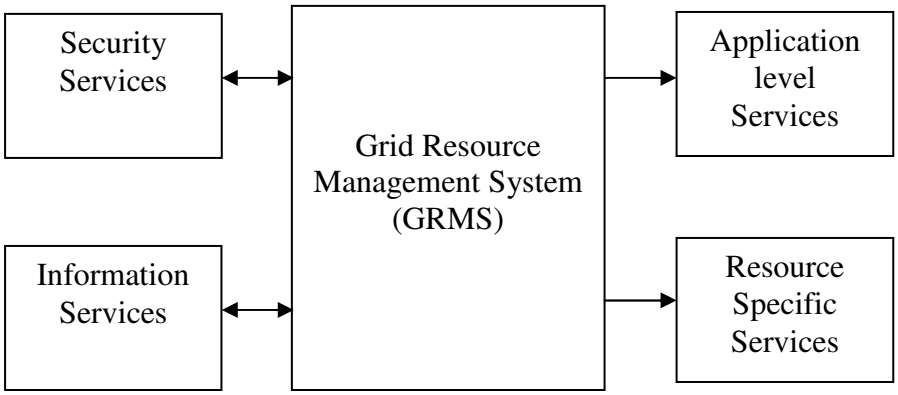


Figure 1.4: High level view of Grid Resource Management System.

Hence the major functions of the resource management systems include accepting the user jobs along with the specified constraints discover a set of matching resources that can execute the job respecting all the constraints and finally schedule the jobs on to the resource.

1.2.1 Grid Resource Discovery

As described earlier, grid environment consists of large number of shared resources distributed among the many locations. Discovery of suitable resources become one of the major components of the Resource Management i.e. performed on the distributed resources, applications etc. Resource discovery is an information-gathering process that digs deep into the details of application requirements from the perspective of client's business, target audience, and industry. Resource Discovery is a very important process because it allows you to see when and where things happened and to collect the information necessary to assess against your desire. Resource Discovery can be defined as a directory service directed to the spontaneous networking environments [12]. Discovery in the grid environment becomes complex as the resources are geographically distributed, heterogeneous in nature and owned by

different individuals and organizations each having their own resource management policies and different access and cost models. The problem of discovering grid resources can be defined as the problem of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements [13]. Resource discovery is performed by resource management system to obtain information about the available resources. In these networks, resources can enter or leave at any time and this mechanism aims to provide information about the resources available at a specific moment [14]. Resource Discovery deals with finding, locating, searching and organizing the matching data (information, facts, quires, numbers etc.), in response to queries of users or an automated mechanism with following trades-off:

- ✓ In minimal possible time
- ✓ Respecting resources' access policy constraints and standards.
- ✓ At minimal cost to the user

Resource Discovery systems become more and more important for highly distributed systems like grid, as resources may leave or join at any time. According to Koen et al. Resource Discovery systems can be categorized by different design aspects mentioned in the [15]. Each of these aspects represents a design choice for which several solutions are possible. The design aspects are: service provider, construction, foreknowledge, architecture, registration, discovery, supported resources, naming and queries.

Resource Discovery has generally two key steps [16]. The first step is to locate a resource, and the second step is to connect or match to it. Locating resource means getting a query to metadata or data. Resource locating would not be quite difficult in a local network that is managed by a central server. Traditional lookup service, i.e.

LDAP service, maintains all information, such as name, location, attribute in a single server that is normally based on central DBMS, which will result constant search performance. But it becomes variable when arbitrary joins and leaves are allowed. The second step after locating the resources is matching that means determining a match between user-specified query and resource characteristics.

1.2.2 Grid Resource Discovery Models

Grid resource discovery mechanisms need to update the resource information regularly. Based on how frequently the information is refreshed the resource discovery models can be categorized into Periodic and On-demand updating models.

1.2.2.1 Periodic Dissemination Model

Periodic models update the information about the recourses' availability after a regular interval of time. They can be further categorized as pull or push models based on who initiates the information update process. In the pull model, resource information system collects data about the resources regularly and updates its database. On the other hand in the push model, resources itself update its information in the database after a regular interval of time.

1.2.2.2 On – Demand Dissemination Model

On-Demand dissemination models are different from periodic models in a way that they don't update the information regularly. Instead the information is updated when an event occurs or some status change occurs.

1.2.3 Grid Resource Scheduling

Another challenging task in the grid resource management is the process of scheduling applications over Grid resources [17]. A Grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and

usually owns the resource. A Grid scheduler is in charge of resource discovery, Grid scheduling (resource allocation and task scheduling), and job execution management over multiple administrative domains.

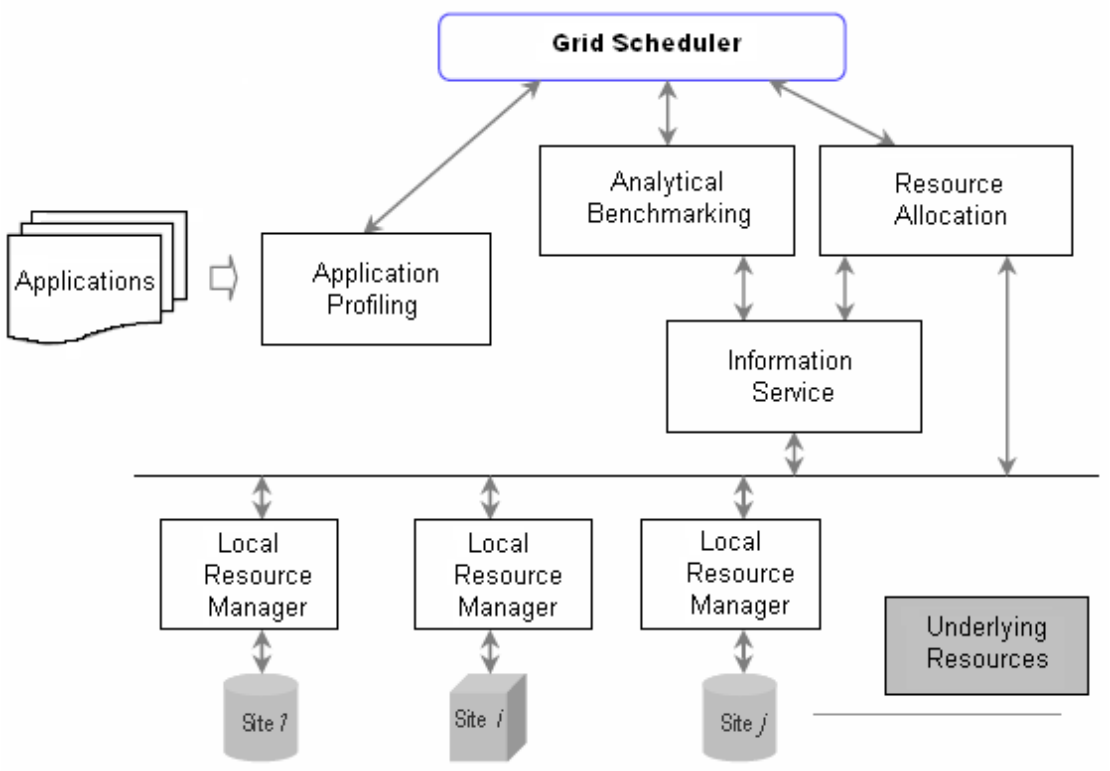


Figure 1.5: Common Grid Scheduler Architecture

Common grid scheduler architecture is shown in Figure 1.5. Local resources lie at the bottom of the architecture each being managed by a local resource manager (LRM). LRM enables low level processing of requests and executing a job that satisfy a request, remote monitoring and management of the jobs as well as updating the information service with the current availability and capabilities of the resources that it manages. Information service component maintains the various resource characteristics and answers to the queries for acquiring the information about the resources. An adequate scheduler must incorporate the current information of resources when making scheduling decisions. Application profiling interface provides a way for user to submit jobs to the grid where as the analytical benchmarking

component provides a measure of how well a computational resource will perform on a given job. Grid Scheduler (sometimes also known as broker) is most important of this architecture. It has two prime responsibilities: one is resource selection, which is the process of selecting feasible and available resources for a given application. Other prime task is mapping which is the process of placing the jobs on to resources for execution. Resource Allocation (RA) component implements a finally-determined schedule through allocating the resources to the corresponding jobs.

With a common Grid infrastructure, Grid scheduling is to answer the question: Given a set of Grid applications (usually computing intensive), how to schedule them over multiple decentralized resources? Each application has a number of tasks. In mapping tasks to resources, we have several basic questions to deal with:

- ✓ How do the relations between tasks affect scheduling decisions?
- ✓ How does the heterogeneity of resources affect the performance of a schedule?
- ✓ What performance models should a scheduler use to determine the quality of a schedule?

1.2.4 Grid Resource Scheduling Models

Grid scheduling is intrinsically more complicated than local resource scheduling because it must manipulate large-scale resources across management boundaries. In such a dynamic distributed computing environment, resource availability varies dynamically [17]. So scheduling becomes quite challenging. Research in the resource scheduling focuses mainly on these four areas.

1.2.4.1 Static Task scheduling

Given a set of tasks and a set of resources, a static scheduler computes the execution schedule before run time. In static task scheduling, resource information and performance parameters are assumed to be known. Depending on how a job can

be divided, relevant research can be categorized into two different areas: divisible workload scheduling [18]-[20], where they can divide workload into arbitrary-sized pieces ("chunks"), and fixed-sized independent tasks scheduling [21].

1.2.4.2 Application-level Scheduling

Application-level scheduling, which explores the effect of different application requirements on scheduling and suggests adaptive scheduling mechanisms based on runtime resource availability. The idea of application-level scheduling is to embed scheduling behavior into the application itself to make application aware of resource variations so that it could decide work size for each job piece based on dynamic resource availability. Early research in this area is quite application-coupled [22] [23].

1.2.4.3 Resource Availability Prediction

The most commonly requested resources by a Grid application requests are computing, data, and network resources. For a Grid resource, it is easy to get a machine's static resource information, such as CPU frequency, memory size, network bandwidth, file systems, etc. But application run-time resources, such as CPU load, available memory, and available network capacity, are variable because Grid is a dynamic resource-sharing computing environment in which there are many applications running. At time of scheduling decision making, accurate resource predication on run-time resource measurement parameters is critical to maximize application performance. In predication research, the analysis is based on historical data on previous resource availability information and job performance records [24]. Various statistical techniques can apply. If resource behavior follows or is assumed to follow certain distribution, stochastic process analysis can be used to predict future resource measurements on a fixed time point or during a certain interval of time.

Regression technique can be used for performance prediction in presence of a performance model.

1.2.4.4 Economic methods in Decentralized Task Scheduling System

The nature of Grid as a large-scale distributed system complicates the scheduling strategies and algorithms. But the cost of scheduling itself must be reasonable to make it efficient. As we look at Grid as a decentralized system, we may think of making the scheduling process decentralized as well. In research on decentralized scheduling, economic theories in market are taken into consideration, because, in nature, market is also a decentralized dynamic system dealing with competitive resources. In market, price is the only measurement while operating. If a pricing policy can be decided for producer (resource) and consumer (applications), using price reduces much communication cost during the process of scheduling. Current market-oriented programming models are based on general equilibrium theory. Given some assumptions, the general equilibrium theory can be transformed to optimization problems, which a scheduling problem is to solve [25]-[28].

Scheduling multiprocessor tasks with unit processing times is a strongly NP-hard problem, which makes it a candidate for optimization.

1.3 Meta-Heuristic Techniques

A meta-heuristic is a heuristic method for solving a very general class of computational problems by combining user-given black-box procedures (usually heuristics themselves) in the hope of obtaining a more efficient or more robust procedure [29]. The name combines the Greek prefix "meta" ("higher level") and "heuristic" ("to find").

Local Search is an iterative transition process that starts with an initial solution and produces a sequence of solutions by applying small local changes. If a new solution rates better when compared with the best current solution, it replaces the current best solution. The whole searching process halts upon reaching the maximum number of iterations, or when no improvement seems possible. The solution to the problem is the best solution found so far. Local search works well when the problem is linear, because in those cases, the local optimum is also the global optimum. However, in non-linear cases where there are several local optimums, a bad starting point can cause local search to terminate upon encountering the first local optimum. This problem is illustrated in diagram. In Figure 1.6 the global optimum *y* is the solution. If the local search algorithm starts at point *a*, then the solution *x* is returned because it is the local optimum. However, if the algorithm were to start from *b* then we are guaranteed to get *y* returned as the solution. Thus, a good starting point plays an important role in local search, but to overcome local optimality one has to switch to meta-heuristics.

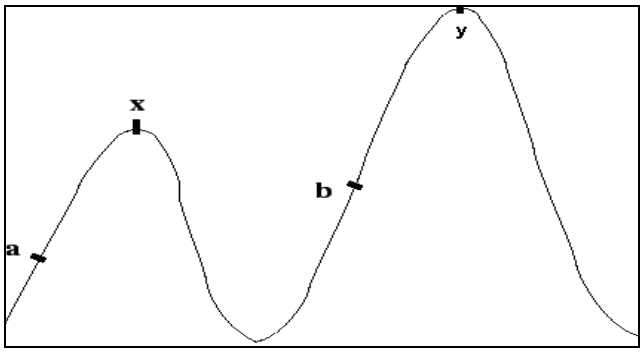


Figure 1.6: Local Optimum Problem

Meta-heuristic is a collection of heuristic algorithms applicable to a wide variety of problems. The heuristic approach is often associated with ``rules of thumb'' or clever insights. Based on the heuristic provided, the algorithm performs the search process iteratively to look for a solution. The iterative search process is terminated

when no improvements are possible. The choice of meta-heuristic algorithm depends on a few factors, such as the solution quality required and the availability of problem knowledge. Because meta-heuristic is a rather ad hoc technique, there is no guarantee a solution can be obtained. In most cases, meta-heuristic techniques served as the last resort when the other two techniques could not produce the desired solutions. Some of the meta-heuristics like Genetic Algorithms (GA), Simulated Annealing (SA) and Ants' Colony Optimization (ACO), used to tackle the problem of resource management in Computational grids are discussed below.

1.3.1 Simulated Annealing (SA)

Simulated annealing is a probabilistic search algorithm. The term simulated annealing derives from the process of heating and then cooling a substance slowly to finally arrive at the solid state [30]. The search algorithm simply mimics the physical process as follows.

In the early stages of the execution, the temperature is high resulting in higher probability, and cause jumping to occur more frequently. Jumping occurs as a way of avoiding local minima, accepting a poorly performed solution with a higher probability. Otherwise, any solution performing better than the current solution is accepted and replaced as the best solution found so far. As the execution time elapses, the temperature decreases thus reduces the frequency of jumping. This probabilistic nature of the system guarantees the exploration of other solution space instead of terminating whenever encountering the first local optimum.

The simulation process terminates after a number of successive executions with no improvements, and returns the best solution so far. The only drawback of simulated annealing is the long execution time to obtain quality solutions. Although it is possible to achieve global optimum solution, it comes at a cost of slower cooling

procedure and longer iteration at each temperature level. Conversely, if in favour of shorter execution time, the algorithm compromises the solution quality.

1.3.2 Genetic Algorithms (GA)

GAs are adaptive methods that can be used to solve optimization problems [31], based on the genetic process of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and *Survival of the Fittest*, first clearly stated by *Charles Darwin* in *The Origin of Species*. By mimicking this process, GAs are able to "*evolve*" solutions to real world problems, if they have been suitably encoded. GA search is constrained neither by the continuity of the function under investigation, nor the existence of a derivative function. It is assumed that a potential solution to a problem may be represented as a set of parameters. These parameters (known as genes) are joined together to form a string of values (known as a chromosome). A fitness function must be devised for each problem to be solved. Given a particular chromosome, the fitness function returns a single numerical fitness or figure of merit, which will determine the ability of the individual, which that chromosome represents. Reproduction is another critical attribute of GAs where two individuals selected from the population are allowed to mate to produce offspring, which will comprise the next generation. Having selected two parents, their chromosomes are recombined, typically using the mechanisms of crossover and mutation. Traditional view is that crossover is the more important of the two techniques for rapidly exploring a search space. Mutation provides a small amount of random search, and helps ensure that no point in the search space has a zero probability of being examined. If the GA has been correctly implemented, the population will evolve over successive generations so that the fitness of the best and the average individual in each generation increases towards the global optimum.

Parallel and distributed algorithms based on simulated annealing and genetic algorithms are applied for the optimization of test vector generation in VLSI circuits [32][33], which is also an NP- complete problem.

1.3.3 Ant Colony Optimization (ACO)

Compared to humans, an individual ant has very little brainpower. While humans have approximately 10 billion neurons, ants have only 250,000. The real power of ants resides in their colony brain. The self-organization of those individuals is very similar to the organization found in brain-like structures. Like neurons, ants use mainly chemical agents to communicate; one ant releases a molecule of pheromone that will influence the behavior of other ants [34]. Ant algorithms are often compared with other evolutionary approaches such as Genetic Algorithms, Evolutionary Programming and Simulated Annealing. It is important to remember that Ant algorithms are non-deterministic and rely on heuristics to approximate to a sub-optimal solution in cases where the number of combinations is extremely huge and impossible to calculate using a deterministic algorithm.

Simulation of ant's behaviour in computer system is a kind of parallel search over several constructive computational threads. Concurrent and asynchronous set of computational agents, known as colony of ants, move through the states of the problem corresponding to its partial solution. The decision to move to next state is influenced by pheromone trail on the route and heuristic information. Each ant constructs the solution during its tour. Pheromone trails are updated at the completion of tour, which will direct the search of future ants.

1.4 Thesis Objectives and Contributions

As observed in the previous sections, grid resource management is a compute intensive problem. Two major areas that need focus are Grid Resource discovery and Grid Scheduling. Given a set of independent tasks and a set of available resources, an efficient grid resource discovery mechanism needs to find the matching resources in a minimal span of time and a grid scheduler attempts to minimize the total execution time of the task set by finding an optimal mapping of tasks to machines. The metric used to find such a mapping is the estimate of "turnaround time" or completion time (machine available time + expected time to compute).

Finding the best mapping is actually a combinatorial optimization problem. Grid resource management is further made challenging by dynamic availability of the resources. The common solution is to use some heuristic search procedure to find a near-optimal solution quickly. Most of the existing research in this area is quite theoretical, because even heuristic search is often too expensive for large search spaces. In a Grid, however, due to the existence of large number of tasks and resources, a practical and efficient meta-heuristic based search algorithm can be a desirable solution for finding the set of matching resources to a user query and then mapping the resource to applications/tasks. Also, more practical experiments should be done to verify the results under a broad range of machine conditions.

In this thesis, therefore, various Grid Resource Management Systems and Meta-heuristic techniques (like GA, SA and ACO) are explored and different ways are described to optimize the resource management in computational grids by using meta-heuristic search algorithms.

1.4.1 Objectives of the Thesis

The objectives of the thesis are:

- ✓ To study and explore Grid Resource Management Systems.
- ✓ To study and analyze Meta-Heuristic techniques (Simulated Annealing & Genetic Algorithms or some hybrid techniques based on these) that can be used for resource optimization.
- ✓ To explore the possibility of using software Agents in Grid Resource Management system.
- ✓ To demonstrate the usability of proposed techniques in Grid Resource Management, so as to provide better Quality of Service. (QoS)

1.4.2 Contributions

To support the thesis "Meta-heuristic techniques for Grid Resource Management" and to demonstrate that meta-heuristics based search algorithms can deliver significant value in order to optimize the grid resource management; several novel research contributions have been made in this work. These are as follows:

- ✓ A detailed study of some of existing resource management systems has been done and Quality of Service (QoS) parameters are identified.
- ✓ An objective function has been formulated, taking the QoS parameters (as identified in first objective) into considerations, which tries to take care of user specified deadlines, as well as tries to ensure the maximum utilization of resources.
- ✓ Grid scheduling has been formulated as a search problem and meta-heuristics like GA, SA, ACO or some hybrid of these techniques has been tried to find the optimal solution. Three new meta-heuristic based grid-scheduling algorithms, as given below, have been proposed, implemented and validated:

- SGASchedule: Proposed SexualGA based Resource Scheduling Algorithm
- ACOSchedule: Proposed ACO based Resource Scheduling Algorithm
- HybridGSA: Proposed Genetic Simulated Annealing based Hybrid Resource Scheduling Algorithm

✓ A software mobile agent based decentralized approach for grid resource discovery has been proposed. Also, a new ACO based algorithm has been proposed that tries to find a set of resources matching to the user query in a quick and efficient manner. Software mobile agents are the components that actually travel to search for resources and ACO based algorithm guides the mobile agent's travel to find the matching resources.

✓ To demonstrate the usability of proposed techniques, a Grid Simulator has been designed and implemented based on well-known grid simulation toolkit known as GridSim. This Grid Simulator implements a centralized scheduler and provides an easy way to integrate and test new scheduling algorithm.

✓ To demonstrate how the above-proposed techniques behave in actual networked environment, a Grid Test bed has been designed and implemented. This test bed provides a grid like environment with web-based user interface. This environment can be used for testing and validating algorithms for grid resource management.

✓ The experimental results prove that proposed algorithms perform better as compared to the existing algorithms. The existing algorithms for resource management mainly focus on the optimizing the resource utilization where as the proposed algorithms try to optimize the resource utilization as well as the

cumulative time delay. Hence the proposed algorithms take care of the interests of both: the resource providers and the grid users.

1.5 Thesis Organization

The thesis is organized as follows.

Chapter 1: Introduction: This chapter provides the basic introduction to the Grid Computing, types of grid and benefits of grid. It provides an overview of Grid Resource Management System. Basic introduction to the meta-heuristics used for optimization and Mobile Agents is also given in this chapter.

Chapter 2: Literature Review: This chapter provides the literature review of earlier work done. A comparative study and analysis of the existing grid resource management systems has also been provided in this chapter.

Chapter 3: Proposed Algorithms for Grid Resource Scheduling: This chapter discusses in detail, three new algorithms for grid resource scheduling proposed in this thesis. This chapter provides the objective function formulation that takes care of the interests of both: resource providers and resource users. Three algorithms are proposed. These are

1. Proposed SexualGA based Resource Scheduling Algorithm (SGASchedule),
2. Proposed ACO based Resource Scheduling Algorithm (ACOSchedule),
3. Proposed Hybrid algorithm for Resource Scheduling based on Genetic Simulated Annealing (HybridGSA).

Details about these proposed algorithms are discussed in this chapter.

Chapter 4: Proposed algorithm for Grid Resource: This chapter provides the details of proposed resource discovery algorithm for grids. This algorithm uses mobile agents for resource discovery. To guide the mobile agents in resource discovery, An ACO based algorithm (ACORD) has been proposed, implemented and tested. This

ACO based algorithm provides learning to the mobile agents, while they travel from one node to another for resource discovery.

Chapter 5: Design and Implementation of Proposed Algorithms: This chapter provides the design and implementation details of the all four proposed algorithms (three for grid resource scheduling and one for grid resource discovery). Grid scheduling algorithms are implemented on the grid simulator based on the GridSim. GridSim allows modeling and simulation of entities in parallel and distributed computing systems for design and evaluation of scheduling algorithms.

Also, the implementation of scheduling and resource discovery algorithms is done on Grid Test bed. Grid Test bed provides the actual grid like environment with a simple web based user interface for submission of jobs, registering and de-registering of resources and administrative tasks. This testbed provides inconsistent and partially consistent completion time (CT) matrix based prediction system which is based on the simulation model used in [21] for the comparison of scheduling heuristics.

Chapter 6: Experimental Details and Results: This chapter provides the details of different experimental scenarios and results of algorithms and techniques proposed in this thesis. Experiments are done on GridSim based scheduler to evaluate proposed algorithms on large-scale grids in a controlled manner. Comparisons have been made with Nimrod/G implementation on GridSim. Experiments are performed on Grid Test Bed to compare the proposed algorithms existing scheduling algorithms based on GA, ACO and SA. This chapter also provides the experimental details and results of the ACO guided mobile agents based resource discovery algorithm.

Chapter 7: Conclusion and Future Scope: Finally, this chapter provides the conclusions and future scope of the work. It is clear from the experimental results that proposed meta-heuristic based algorithms are better to tackle the problem of resource

management in grid environment as compared to the existing ones. The future research can be focused on the parallel implementation of the proposed algorithms.

1.6 Summary

This chapter starts with an introduction and motivation of grid computing followed by different types of grid and its benefits. A brief discussion on the Grid resource management system is provided. The major problems of grid resource management like grid scheduling and grid resource discovery are introduced. An overview of different meta-heuristics is also provided. Finally this chapter provides the problem formulation and objectives of the thesis followed by the thesis contributions and thesis organization. Next chapter focuses on the study of existing Grid Resource Management systems and existing algorithms for grid scheduling and grid resource discovery.

2

Literature Survey

2.1 Introduction

Grid Computing is the next generation of IT infrastructure. It consists of collection of heterogeneous resources spread across multiple administrative domains. Grid Computing provides scalable, secure and extremely high performance mechanism for discovery and access to remote computing resources, in a seamless manner [35]. The reference [36] presents a middleware infrastructure, called as Internet Operating System (IOS), which aims at freeing application developers from dealing with non-functional concerns while seeking to optimize application performance and global resource utilization and [37] discusses about the adaptive

middleware and programming technology for dynamic grid environments. Computational Grid has the potential to become the main execution platform for high performance and distributed applications. However, such systems are extremely complex. The possibility to discover a matching set of resources from a pool of an enormous amount of resources and mapping the jobs to the resources for execution is one of the challenging problems that a grid resource management system has to tackle. This chapter provides the survey of existing grid resource management systems and also an overview of existing resource discovery algorithms and scheduling heuristics.

2.2 Grid Resource Management System (GRMS)

Grid computing basically aims to harness the computing power of idle resources by borrowing the resources from the resource providers, rather than to buy more computational resources. Resource management in the computational grids deal with identifying requirements, matching resources to the applications, allocation of resources and scheduling and monitoring of the grid in a timely manner. Since a World Wide Grid (WWG) can consist of all the available resources connected over the internet, so finding a set of matching resources as per the requirements of the jobs/ tasks to be executed, and then generating and efficient job-resource mapping becomes very complex and tedious. Security and fault tolerance further add the complexity to the management of resources, and make this system more complex and challenging.

Traditional resource management systems can implement effective mechanisms and policies for the efficient use of the resources, as they have complete control over the resource. On the other hand in the grid resource management systems have resources distributed over different administrative domains. Hence the grid resource management system has to take care of resource heterogeneity, difference in usage, scheduling policies and security mechanisms. Co-allocation of resources

(allocating resources in different sites to an application simultaneously) is also made complex due to site autonomy, which further adds the complexity to the resource management systems for computational grids [8]-[11].

A high level view of grid system is shown in Figure 1.4 (Chapter 1). In a grid system, an end user submits to the management system the job to be executed along with some constraints like job execution deadline, the maximum cost of execution. The function of the resource management is to take the job specification and from it estimate the resource requirements like the number of processors required, the execution time, and memory required. After estimating the resource requirements RMS is responsible for discovering available resources [11] and selecting appropriate resources for job execution. Finally schedule the jobs on these resources by interacting with the local resource management system.

2.2.1 Grid Resource Management System Taxonomy

Taxonomy for distributed system scheduling for static and dynamic task scheduling is presented in [38] and [39] respectively. Grid Resource Management System taxonomy based [40] on the resource management system architecture is given below. It is divided into following types: machine organization, resource namespace organization, resource discovery and dissemination and scheduler organization.

2.2.1.1 Machine Organization

The organization of the machines in a grid determines the scalability of the architecture. Machine organization also affects the communication patterns of the resource management system. Machine organization can be categorized as *centralized* or *de-centralized*. In the *centralized* organization there is a single controller or a designated set of controllers that perform the scheduling for all the machines. On the

other hand in the *de-centralized* organization, the control is distributed among the machines in the grid. Centralized organization is generally suffered from scalability issues and hence not recommended for a grid. Other type of categorization can be done as a *flat*, *hierarchical* or *cell structure* organization. A *flat* organization is a kind of peer to peer organization where all the machines communicate with each other without any intermediate machine. In a *hierarchical* organization all the machine in the same level can communicate with the machines directly above or below or peer to them in the hierarchy. This organization is generally recommended for a grid as this is scalable. Last in this categorization is the *cell structure*, where are the machines in the cell communicate like flat organization but the communication with the machines outside the cell or in other cells is via some designated contact nodes only. Internal structure of the cell is generally hidden from other cells. Cells can be further organized as flat or hierarchical structure.

2.2.1.2 Resource Namespace Organization

The grid resource management system needs to maintain a globally named pool of resources. Resource namespace organization affects the other resource management functions like resource discovery and resource dissemination. Resource namespace can be organized as *relational*, *hierarchical* or *graph based*. A *relational* namespace divides the resources in to relations and used the concepts from relational databases to indicate the relationships between tuples in different relations. A *hierarchical* namespace follows a system of systems approach and a name is constructed by traversing down the hierarchy. A *hybrid* approach using *relational and hierarchical* namespace organization is also popular. Globus MDS is an example of hybrid namespace organization. Third approach is the *graph based* organization,

where the resource are linked together and the resource names are constructed by following the links from one object to another.

2.2.1.3 Resource Discovery and Dissemination

Resource discovery services are an important function of resource management in a grid system. These are used by the scheduling system to obtain the information about the resources. Discovery process is initiated by network applications to find the suitable resources in a grid. Resource discovery approaches can be classified as *query based* or *agent based*. *Query based* systems enquire a centralized or distributed information store for the resource availability. In *agent based* systems information is gathered by discovery agents. The approaches for updating the resource information in the database constitute the resource dissemination process. This process is complimentary to resource discovery and involves advertising the resources in the grid. Dissemination protocol effect the amount of data transferred between systems. Resource dissemination can be either *periodic* or *on-demand*. In the *periodic* dissemination models the information is updated periodically. These can further categorized as *pull* or *push* model based on whether the information is updated / pushed by resource or the information store collects the information, respectively. *On-demand* resource dissemination models update information on the triggering of and event or some status change.

2.2.1.4 Scheduler Organization

Scheduling models can be categorized as *centralized, hierarchical* or *de-centralized*. Scheduler organization determines the structure of resource management system and its scalability. In *centralized* scheduler organization there is one scheduling controller and all the jobs are submitted to this controller. This single scheduler is responsible for scheduling all the jobs on to the resources. Due to the

availability of information at one single location, optimal scheduling decisions are generally achieved, but this approach lacks in the scalability and fault tolerance. In a *hierarchical* organization scheduling controllers are organized in a hierarchy. This approach provides the advantages of scalability and fault tolerance over the centralized scheduler. Other approach is *de-centralized* model in which there is no centralized scheduler and scheduling is done by resource requesters and resource owners independently. This approach is scalable and maintains the site autonomy. Hence this is suitable for grid. Individual schedulers cooperate with each other in making scheduling decisions. This can be further categorized as *cooperative* and *non-cooperative* schedulers based on whether the individual schedulers cooperate with each other or not.

2.3 Grid Resource Management Systems Survey

Grid systems can be broadly categorized as compute, data or service based grids. This section discusses the details of some existing computational grid systems. Globus, Condor, Condor-G, Legion, Nimrod/G, Alchemi, GridWay and Gridbus Broker are studied.

2.3.1 Globus

Globus software [43]-[46] is designed to enable applications that federate distributed resources, whether computers, storage, data, services, networks, or sensors. Initially, work on Globus was motivated by the demands of “virtual organizations” in science. The collection of tools in the Globus toolkit provide basic services and capabilities like Resource Management, Resource Discovery, Security, Information Services etc. that are required for Grid Computing. Globus resource management system is a collaboration of different components which consists of *resource brokers*,

resource co-allocators [45] and *resource manager* (GRAM) [46]. Globus is developed as a layered architecture in which the higher level services can be developed using the lower-level core services [47]. Globus emphasizes on hierarchical integration of grid services and components.

Globus uses a de-centralized scheduling model which supports QoS via resource reservation. Scheduling policies can be extended in the Globus is by the application level scheduler / brokers like Nimrod/G. Extensible Resource Specification Language (RSL) is used to specify the resource requests. Brokers are responsible for taking high level RSL specification and transforming them into more concrete specifications. Globus provides a component to implement resource management, data management, and information services, as shown in Figure 2.1.

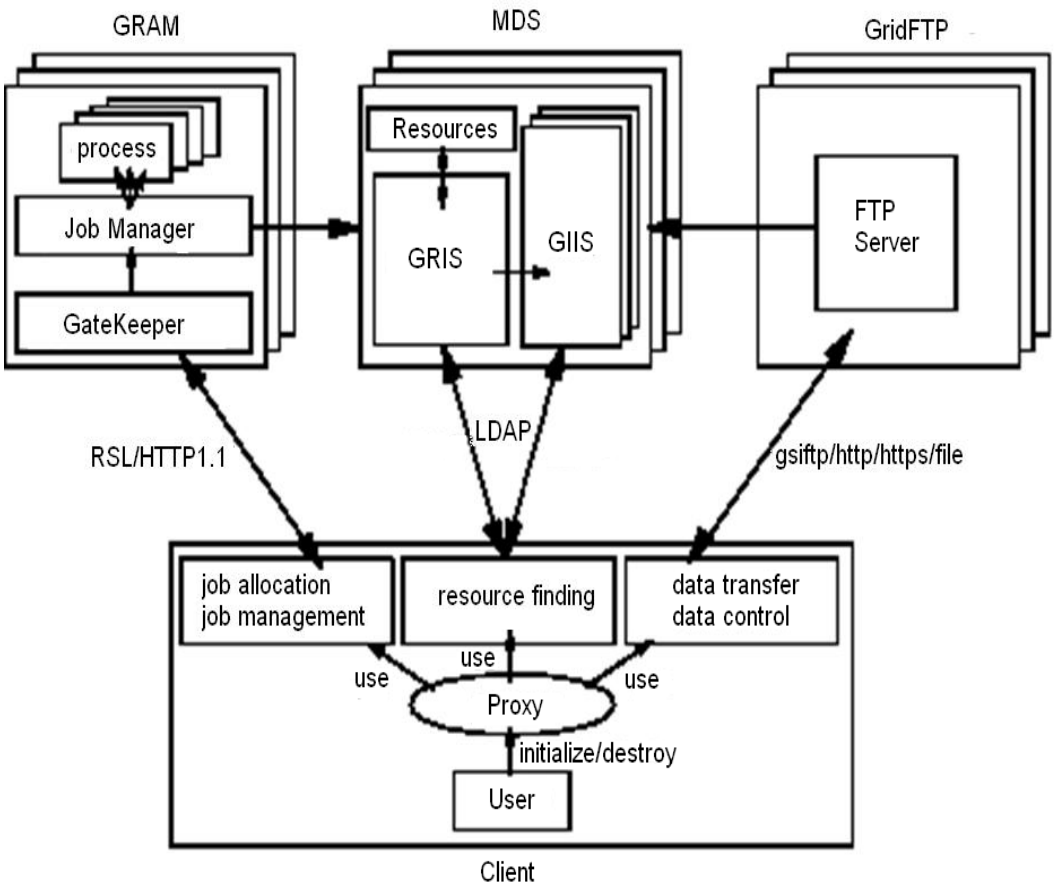


Figure 2.1: The system overview of Globus ToolKit

Resource discovery in the Globus is done by querying the LDAP (Lightweight Directory Access Protocol) based information store called as Metacomputing Directory Services (MDS). MDS consists of two components: Grid Index Information Service (GIIS) and Grid Resource Information Service (GRIS). GRIS is responsible for providing all resource discovery services. Resource information is updated by push dissemination and GIIS provides the global view of resources. Globus has a hierarchical namespace organization. Globus toolkit also provides an open source set of services and software libraries that supports the development of Grid systems and applications [48].

2.3.2 Condor

Condor [49] [50] is a high throughput resource management system that manages a heterogeneous pool of resources. It harnesses the computing power of idle resources by steeling the idle CPU cycles. Condor system follows a layered architecture. A set of resources managed by condor is known as condor pool. Condor keeps all the jobs submitted by the user in a queue. These jobs are then scheduled by the condor on to the machines in the pool transparent to the user [51][52]. Condor can also migrate a running job from one machine to another until it is completed [53][54].

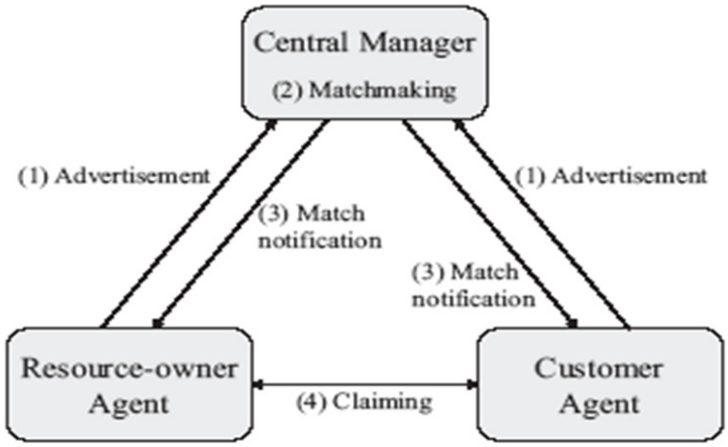


Figure 2.2: Condor Architecture

Condor uses a resource specification language, known as *Classified Ads* to specify the resource requests. ClassAds uses a semi structured data model and a query language as a part of data model that enables the advertising agents to include constraints to resource requests and offers and hence specify their compatibility.

Condor has a centralized scheduling model and uses a dedicated machine, known as Central Manager, which is responsible for scheduling the jobs onto the resources in the condor pool. Condor also supports pre-emption of jobs. In case a resource is withdrawn then already running jobs are check-pointed and pre-empted to other resources, thus ensuring the resource owner autonomy as well as proper completion of the jobs.

Condor can have multiple condor pools and each pool follow a flat RMS organization. Resource dissemination is through periodic push mechanism where each machine updates the resource status with the central information store owned managed by *Condor Collector*. A Condor resource agent runs on each machine periodically advertising its services to the collector. Collector is queried by condor matchmaker for resource discovery that is used to determine the matching resources and job requests [55], [56].

2.3.3 Condor-G

Condor-G [57] is the marriage of technologies from the Condor project and the Globus project. Condor-G combines the inter-domain resource management protocols of the Globus Toolkit and the intra-domain resource and job management methods of Condor to allow the user to harness multi-domain resources as if they all belong to one personal domain. So, it leverages recent advances in two distinct areas:

- ✓ security and resource access in multi-domain environments, as supported within the Globus Toolkit, and

- ✓ management of computation and harnessing of resources within a single administrative domain, embodied within the Condor system.

Condor-G provides the grid computing community with a powerful, full-featured task broker. Condor-G is mainly focused on job management part of Condor that lets the user to submit jobs into a queue, have a log detailing the life cycle of submitted jobs, manage input and output files, along with everything else that is expected from a job queuing system. Condor-G uses the Globus Toolkit to start the job on the remote machine instead of using condor-developed protocol. Hence Condor-G provides a "window to the Grid" for users to both access resources and manage jobs running on remote resources.

2.3.4 Nimrod/G

Nimrod/G is a grid resource broker [58] which is used as scheduling component in a new framework called Grid Architecture for Computational Economy (GRACE) [26] which is based on using economic theories for grid resource management systems. This is designed to run parametric application on computational grids [59]-[61]. It uses the middleware services offered by Globus toolkit. The main components of the GRACE infrastructure are a Trade Manager (TS), trading protocols and Trade Server (TS). TM is the GRACE client in the Nimrod-G resource broker that uses the trading protocols to interact with trade servers and negotiate for access to resources at low cost. Trade Server is the resource owner agent that negotiates with resource users and sells access to resources. TS uses pricing algorithms as defined by the resource owner that may be driven by the demand and supply. It also interacts with the accounting system for recording resource usage.

Resource discovery is accomplished by using the MDS services. Nimrod/G uses GRAM APIs to dispatch jobs over grid resources. Users can specify the deadline

and budget for their jobs. Also resource providers can specify the price for offering the resource and other services. Nimrod/G tries to find the cheapest resources which can meet the deadline conditions for the jobs [60]. Though, Nimrod/G uses the middleware services provided by Globus Toolkit, but it can be extended to other middleware services also.

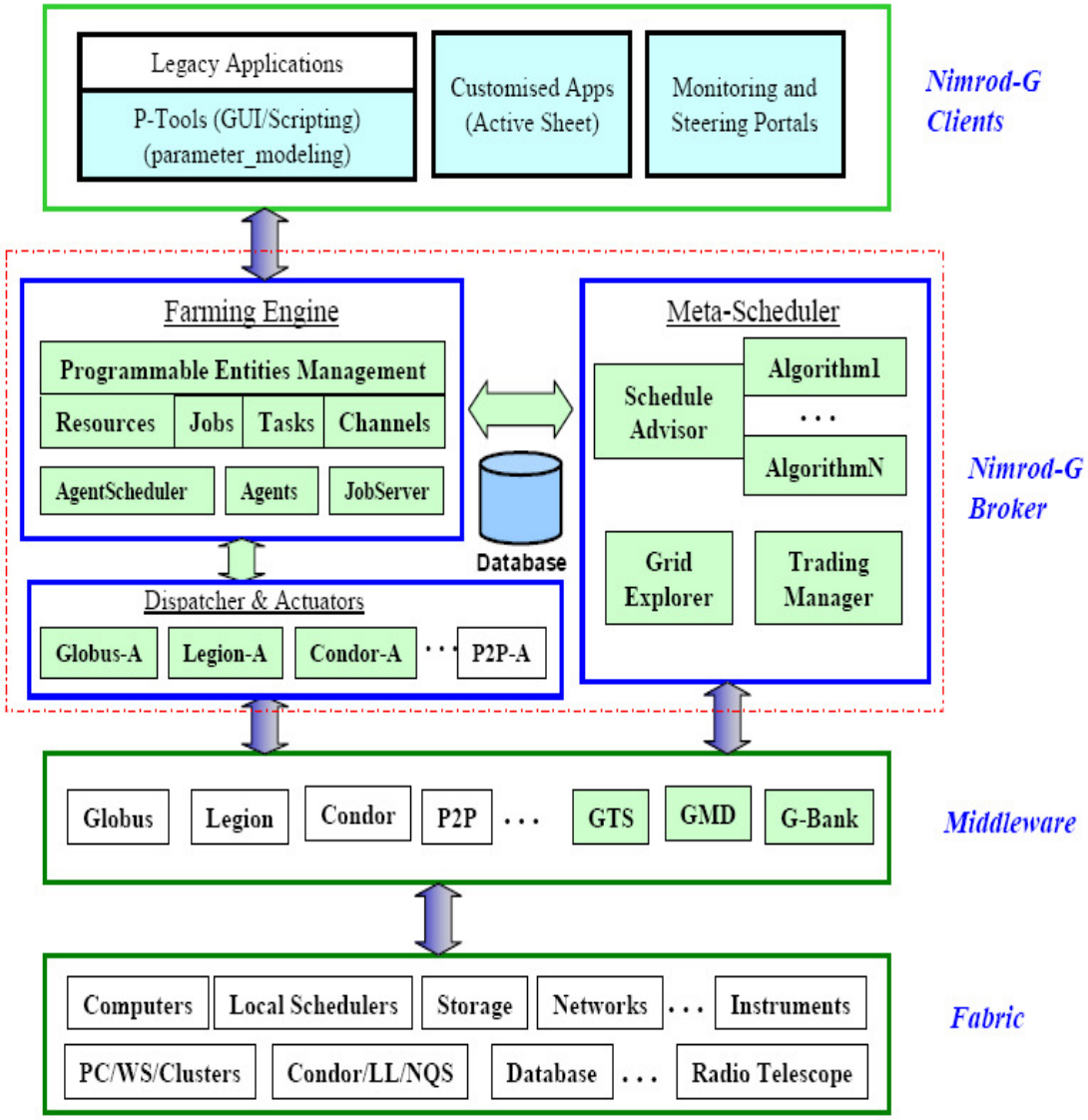


Figure 2.3: Layered Architecture of Nimrod/G [25]

Nimrod/G used a hierarchical machine organization, hierarchical namespace organization and market driven computational model for resource management.

2.3.5 Legion

Legion is an object based operating system for Grids [62]. This is a metasystem that offer software infrastructure for the Grids. This software infrastructure consists of a framework for scheduling different classes of applications with different type of scheduling strategies. The Legion objects are defined and managed by corresponding class. New instances for the classes are created and scheduled for execution and activated and deactivated [64][65].

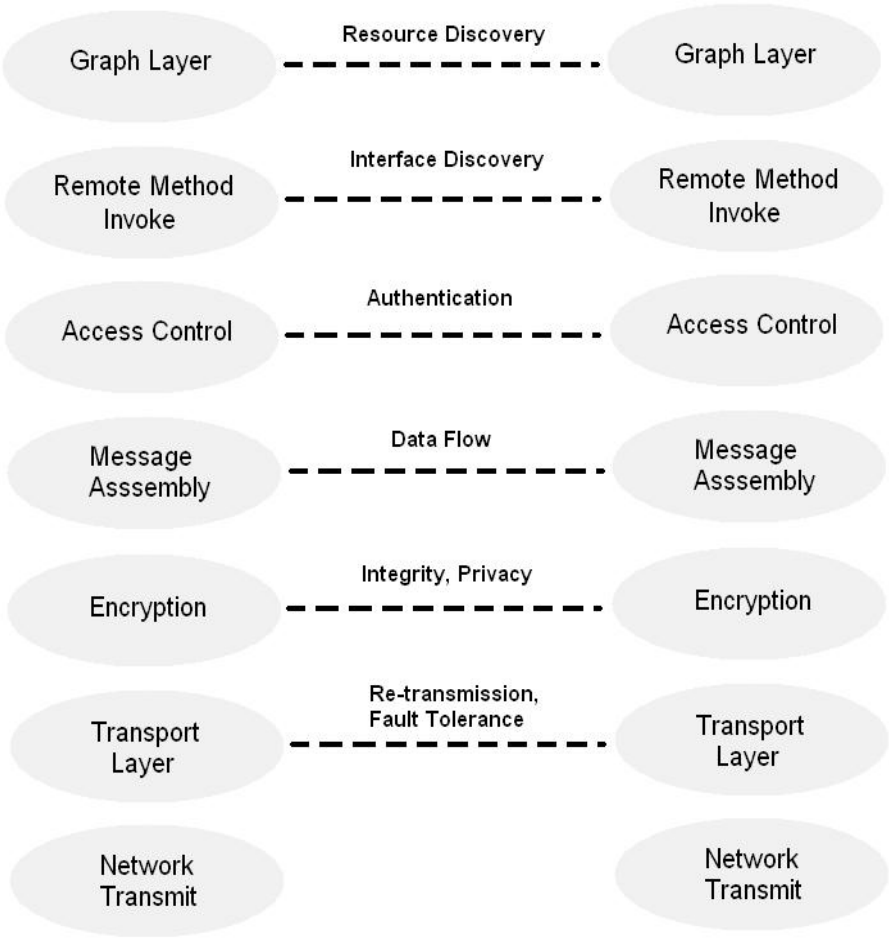


Figure 2.4: Communication in Legion objects through protocol stack

Legion has a hierarchical scheduling structure. Higher level scheduler does the job of global scheduling where it allocates the jobs to a cluster or a resource group. Actual scheduling of jobs on the resources with in the cluster is done by the local

resource manager of that domain [66]. Legion objects are active processes and scheduling in the Legion is placing the objects on the processors. Co-allocation of resources in not supported.

Resources are organized in a graph based organization. Resource information is maintained in an object based information store and is managed through Collections [66]. Collection is a kind of information about multiple resources that is aggregated into single object. Periodic pull mechanism is used for resource dissemination where the Collections pull state information from host objects. Use of Collections makes the system more scalable where more collections can be added as required and these collections can communicate with each other to send and receive data from each other. Communication between any two objects goes through Legion protocol stack as shown in Figure 2.4.

2.3.6 Alchemi

Alchemi is .NET-based enterprise Grid computing and runtime machinery for creating a high-throughput resource-sharing environment [67]. The Alchemi Grid Computing framework was conceived with the aim of making Grid construction and development of Grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility [68]. Alchemi architecture follows the master-worker parallel programming paradigm [69] in which a central component dispatches independent units of parallel execution to workers and manages them. Alchemi includes the runtime machinery (Windows executables) to construct computational Grid and a .NET API and tools to develop .NET Grid applications and Grid-enabled legacy applications. Reference [70] gives the performance comparison with Globus.

Alchemi consists of four major components: Manager, Executor, Owner and Cross-Platform Manager as shown in Figure 2.5

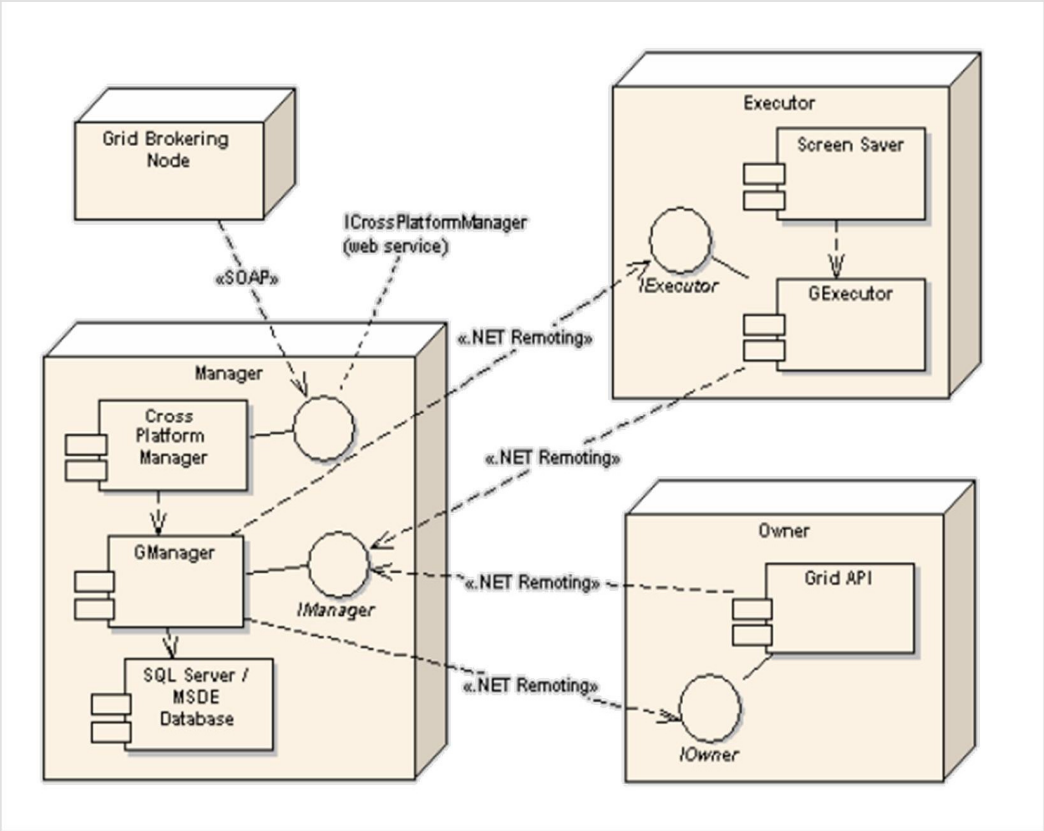


Figure 2.5: Alchemi architecture and communication between its components

Manager logically couples the Windows desktop machines running the instance of Alchemi Executor service. Executors register themselves with the Manager which in turn keeps track of their availability. Threads received from the owner are placed in a pool and scheduled to be executed on the various available executors. Owner provides an interface with respect to Grid applications between the application developer and the Grid. Cross-Platform Manager is an optional component of the Alchemi which provides a web services based interface to manage the execution of platform independent grid jobs.

Alchemi is an open source software framework that allows us to painlessly aggregate the computing power of networked machines into a virtual supercomputer (Computational Grid) and to develop applications to run on the Grid.

2.3.7 GridWay

GridWay is a new experimental framework based on Globus that allows an easier and more efficient exuection of jobs on a dynmic Grid environment in a "submit and forget" fashion [71]. The core of the Gridway framework is a personal submission agent that performs all the scheduling stages and watches over the correct and efficient execution of jobs. Adaptation to changing conditions is achieved by dynamic rescheduling: once the job is initially allocated, it is rescheduled when a migration circumstance is detected.

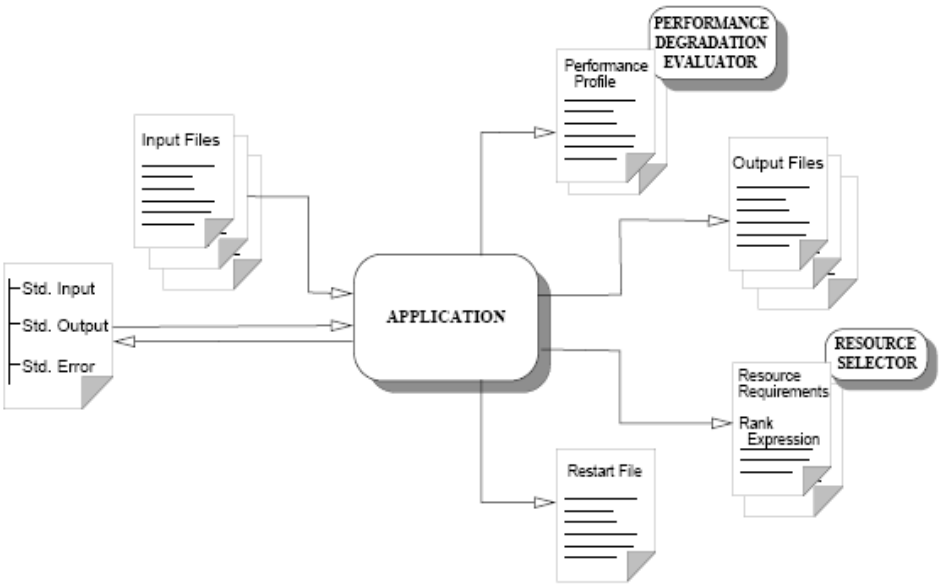


Figure 2.6: GridWay framework model for self adapting applications [71]

Job execution is performed in three stages by the prolog (prepares the remote system and stages the input files), wrapper (executes the actual job) and epilog (stages the output files and performs cleans up) modules which can be defined on a per job basis Migration is performed by combining the above stages. First, the wrapper is canceled and then the prolog is submitted to the new candidate resource and finally epilog is submitted to the old resource for clean up.

The submission system uses the resource selector (which evaluates the requirements for job scheduling) and performance evaluator module (which periodically evaluates the application performance) to detect performance slowdown and request re-scheduling. It also provides the application with fault tolerance capabilities. GridWay doesn't necessarily require source code changes in the application, but with minimal changes, applications could benefit from the self-adapting features provided by the framework.

2.3.8 Gridbus Resource Broker

The Gridbus broker extends the Nimrod-G computational Grid resource broker model to distributed data-oriented grids and extends Nimrod-G's parametric modeling language by supporting dynamic parameters [72]. Nimrod-G specializes in parameter-sweep computation with no functions for accessing remote data repositories and for optimizing on data transfer.

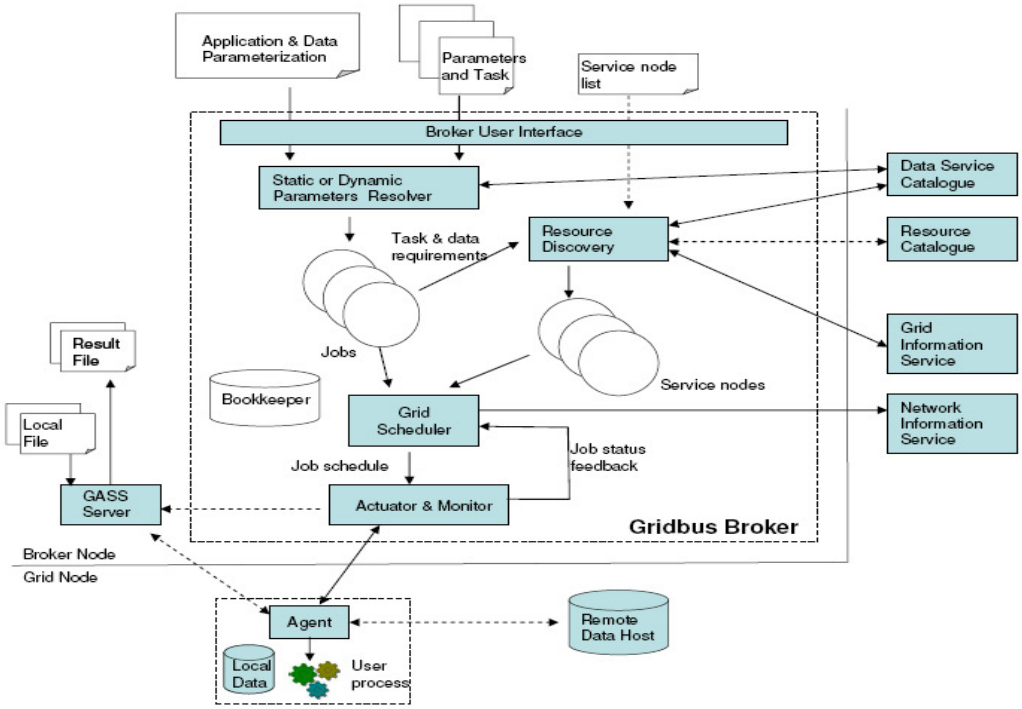


Figure 2.7: Gridbus Broker Architecture [72]

The architecture of the Gridbus broker is shown in Figure 2.7. The inputs to the broker are the tasks and the associated parameters with their values. A task is a sequence of commands that describe the user's requirements and the associated parameters can be static or dynamic. The task requirements drive the discovery of resources such as computational nodes and data resources. Task description is decomposed into jobs, which is sent to Grid node. The set of jobs along with the set of service nodes are an input to the scheduler. The scheduler matches the job requirements with the services and dispatches jobs to the remote node. For the jobs requiring remote data, it interacts with network monitoring service to optimize the data transfer. Actuator component dispatches the jobs to the remote nodes. The Monitoring component keeps track of job status – whether the jobs are queued, executing, finished successfully or failed. The Bookkeeper keeps a persistent record of job and resource states throughout the entire execution.

2.4 Comparison of Grid Resource Management Systems

Table 2.1 below summarizes different grid systems discussed above, based on the taxonomy discussed above in section 2.2.1.

GRID SYSTEM	MACHINE ORGANIZA-TION	NAMESPACE ORGANIZA-TION	RESOURCE DISCOVERY AND DISSEMINA-TION	SCHEDULER ORGANIZA-TION
Globus	Hierarchical Cell machine organization	Hierarchical namespace	Distributed queries discovery, Periodic push dissemination	De-centralized scheduler. Use external schedulers for scheduling.
Condor	Flat machine organization	Hybrid namespace	Centralized queries discovery, Periodic push dissemination	Centralized scheduler
Condor-G	Hierarchical machine	Hybrid namespace	Distributed Queries	De-centralized scheduler.

	organization		discovery, Periodic Push dissemination	
Nimrod/G	Hierarchical Cell machine organization	Hierarchical namespace	Distributed queries discovery, Periodic dissemination	Hierarchical decentralized scheduler
Legion	Hierarchical machine organization	Graph namespace	Distributed Queries discovery, Periodic pull dissemination	Hierarchical scheduler
Alchemi	Centralized	Hierarchical namespace	Centralized queries discovery	Centralized scheduler uses First come first served Policy.
GridWay	Hierarchical Cell machine organization	Hierarchical namespace	Distributed queries discovery, Periodic push dissemination	De-centralized scheduler. Use external schedulers for scheduling.
Gridbus Resource Broker	Hierarchical Cell machine organization	Hierarchical namespace	Distributed queries discovery, Periodic dissemination	Hierarchical decentralized scheduler

Table 2.1: Comparison of different Grid Systems.

2.5 Grid Scheduling Heuristics

Scheduling in a Grid Resource Management System is mainly done by a scheduler or a broker. The scheduler at a lower level implements some scheduling heuristic to generate an optimal schedule. The reference [73] presents a cost-effective scheduling algorithm for message passing multiprocessor system where as duplication based scheduling algorithms are discussed in [74], [75]. Existing scheduling heuristics as given below are discussed and compared in [21].

2.5.1 Opportunistic Load Balancing (OLB)

OLB is very simple heuristic and assigns the jobs to resources, in an arbitrary manner as soon as the resource is available. Assignment is done without taking job's

expected execution times into consideration. Main aim of OLB is to keep all machines as busy as possible. OLB generally results in very poor makespans.

2.5.2 Minimum Execution Time (MET)

MET works in contrast to OLB and assign each job, in an arbitrary manner to the resource with the best expected execution time for that job, regardless of the availability of the resource. MET focuses on assigning each job on a best resource for it. MET tries to find good job-resource pairings, but because it does not consider the current load on a resource, it will often cause load imbalance between the processors.

2.5.3 Minimum Completion Time (MCT)

MCT combines the benefits of both OLB and MET, by assigning each job to the resource with minimum expected completion time for that job. MCT tries to avoid the circumstances in which OLB and MET perform poorly.

2.5.4 Min-min

Min-min establishes the minimum completion time for every unscheduled job (in the same way as MCT), and then assigns the job with the least minimum completion time (hence Min-min) to the resource which offers it this time. Min-min is based on the minimum completion time, as is MCT. However, Min-min considers all unmapped jobs during each mapping decision and MCT only considers one job at a time.

2.5.5 Max-min

Max-min is very similar to min-min. Again the minimum completion time for each job is established, but the job with the maximum minimum completion time is assigned to the corresponding resource. Max-min is based on the intuition that it is

good to schedule larger jobs earlier on so they won't cumulate at the end causing a load imbalance.

2.5.6 Duplex

Duplex heuristic is a combination of both Min-min and Max-min heuristics. Duplex performs both Min-min and Max-min heuristics and the uses the better solution.

2.5.7 Genetic Algorithms (GAs)

GAs are an evolutionary techniques that is used to search for optimal solution in a very large search space. GAs are inspired from human genetics and generally work by encoding the problem in the form of chromosomes. GA operators like crossover and mutations are applied and new generations are evolved. Fitness is computed after every generation and further exploration is stopped as soon as an acceptable fitness value is achieved. GAs are till now best known heuristics for scheduling.

2.5.8 Simulated Annealing (SA)

SA is an iterative technique that considers only one possible solution (mapping) for each job at a time. This solution uses the same representation as the chromosome for the GA. SA uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search of the solution space. This probability is based on a system temperature that decreases for every iteration. As the system temperature "*cools*" it is more difficult for poorer solutions to be accepted.

2.6 Grid Resource Discovery Algorithms

Number of algorithms has been developed for Resource Discovery process in Grids. Few of popular algorithms are discussed below.

2.6.1 Flooding Algorithm

This algorithm is based on agent-based approach and is widely used by Internet routers [76]. It is also used in peer to peer file sharing system, advertising certain routing tables or link states to routers and some part of routing protocols that are used in ad-hoc wireless networks. Every node acts as a transmitter and receiver and every node tries to send every message to every node of its neighbor. Newly added edge is not used for any communication. Direct communication exists only in between initially existing set of neighboring edges of the network. The required number of rounds of this algorithm is equivalent to the diameter of the graph. This algorithm can be very slow if not started with a graph, which has small diameter. This algorithm is not quite good for Grid Computing, due to the size of the Grid environment which, consists of many nodes.

2.6.2 Swamping Algorithm

Agent based approach acts as the base approach for implementing this algorithm. Swamping algorithm is identical to flooding algorithm except that this algorithm allows a node to connect not only with the set of initial neighbors but also with all of its current neighbors [76]. The main advantage of this algorithm is that it has better speed than the flooding algorithm and it needs $O(\log n)$ rounds to converge to a complete graph. The disadvantage of this algorithm is, its communication complexity grows very quickly.

2.6.3 Random Pointer Jump Algorithm

In this algorithm, in each round, each node contacts with a random neighbor, and then this random neighbor sends all of its neighbors to the sender node. Finally sender neighbor and random neighbor's neighbors get merged [76]. This algorithm

claimed that a strongly connected graph with *n* nodes needs *O(n)* complexity time to converge to a complete graph it is good choice for strongly connected graphs. This can further be categorized as *Before Random Pointer Jump* or *After Random Pointer Jump* based on whether the connection to the neighbor is made before passing the information or after passing the information to the other node.

2.6.4 Name Dropper Algorithm

This is an algorithm for querying resources in a weakly connected network where it is assumed that all machines already know each other [76]. The algorithm, which takes $O(log_2 n)$ rounds to learn each other, $O(n^2 log_2 n)$ number of connections and $O(n^2 log_2 n)$ number of pointers in a network. It is claimed [76] that Name Dropper algorithm is more efficient in terms of required time and total number of network communications compared to four other algorithms, mainly *flooding algorithm, swamping algorithm, random pointer jump algorithm*, and *random pointer jump with back edge* algorithm.

2.6.5 Time-To-Live based Reservation Algorithm

The time-to-live based reservation algorithm is fully decentralized Resource Discovery algorithm and is most suitable for Grid Computing" [77]. This algorithm is based on a simply unicast request transmission that can be easily implemented. The addition of a reservation algorithm enables resource discovery mechanism to find more available matching resources. The deadline for resource discovery time is decided with time-to-live (TTL) value.

2.6.6 Discovering Intermittently Available Resources (DIAR) Algorithm

This algorithm is used to discover the occasionally available resources in multimedia environment [78]. Different policies for a QoS based Resource Discovery

service for a given graph theoretic approach is defined. The performance evaluation of QoS policies based on different time-map strategies in a centralized system proves that randomized placement strategies and increased server storage can facilitate better performance to discover a particular resource.

2.6.7 Absorption (Randomized Resource Discovery) Algorithm

The Randomized Resource Discovery algorithm, also called "absorption algorithm", is based on a strongly connected graph and is developed in two stages [79]. First stage is to determine the ultimate leader in the network where a network of *n* nodes is partitioned into clusters; each cluster has its leader node and all nodes in a cluster know their leader. In second stage, the ultimate leader of the network broadcasts the pointers to the each member of the network in one time step. The authors claimed that this absorption algorithm takes $O(log\ n)$ steps time complexity, can send $O(n\ log\ n)$ number of messages, and can pass $O(n^2 log\ n)$ number of pointers with high probability.

2.7 Summary

This chapter provides the review of grid resource management systems and core algorithms. Taxonomy for assessing the Grid Resource Management System is given and a survey of existing grid middleware is presented to show how different components had been implemented in grid systems. Following the comparison of different Grid systems, a detailed review of existing grid scheduling and grid resource discovery algorithms is presented. Next chapters (Chapter 3 and Chapter 4) focus on the details of proposed meta-heuristics based algorithms for grid scheduling and proposed ACO guided mobile agent based algorithm for grid resource discovery.

3

Proposed Algorithms for Grid Resource Scheduling

3.1 Introduction

Computational Grid is a distributed and heterogeneous computing system with a number of resources of different types. Since a worldwide grid could span over million of computers connected via Internet so grid resource scheduling becomes complex as the size of the grid grows. Mapping a set of tasks on to set of resources in this environment is compute intensive problem. Dynamic nature of resources, different administrative policies and Quality of Service (QoS) requirements make the

problem even tougher. Use of conventional methods becomes infeasible as the search space becomes very large. A new area of research has been setup to tackle this problem, which makes use of meta-heuristic techniques to find the optimal or near optimal solution to the problem. This chapter proposes and discusses in detail, three new algorithms for resource scheduling on computational grids. These are hybrid algorithms based on well-known meta-heuristics Genetic Algorithms, Ants' Colony Optimization and Simulated Annealing.

Many algorithms exist [80], [81], [84]-[89], [93]-[95], [100]-[103] in literature based on these techniques for resource scheduling. The proposed techniques are different from existing ones in many ways. First, most of the existing algorithms are not particularly designed and tested for grid systems. Secondly, the algorithms that are tested in grid environment consider only makespan as Quality of Service (QoS) parameter. They fail to take the user interests into consideration, which is one of the important QoS parameter of grids. So, proposed algorithms differ in the objective / fitness function from the existing ones. Proposed algorithms try to find the optimal schedule based on the objective that ensures the proper utilization of the resources as well as try to respects the interests of grid users by minimizing the delay in meeting deadlines for the grid jobs.

3.2 Problem Formulation

Grid scheduler is the important part of Grid Resource Management System (GRMS), which gathers the information about the resources and chooses the best resource as per the job requirements. This is followed by the actual execution of the jobs. The problem of finding the best "*job-resource*" pair is a compute-intensive problem and need to be formulated mathematically to find the optimal solution.

To formulate the problem, we consider mapping a set of independent user jobs $\{J_1, J_2, J_3, \ldots, J_N\}$ to a set of heterogeneous processors / resources $\{P_1,P_2,P_3,\ldots,P_M\}$ (Though we can have any type of resource that can be shared on a computational grids, but for our simulations we have considered only processors as a resource.) This mapping is done with an objective of minimizing the completion time and utilizing the resources effectively, and also minimizing the delay in meeting user specified deadlines. The speed of each resource is expressed in number of cycles per unit time, and the length of each job in number of cycles. Each job J has processing requirement C_j cycles and processor P_i has speed of V_i cycles/second. Any job J has to be processed by P_i, until completion.

3.2.1 Objective / Fitness Function

The Objective / fitness function is used to differentiate between high and low quality solutions. It is defined in terms of objectives of the problem in hand. For grid scheduling problem, the users have goal of satisfying the deadlines of jobs submitted by them whereas the resource providers like to minimize the makespan. The schedule S is evaluated on the basis of fitness function, which is defined as

$$\textbf{Fitness, } F(S) = 1/(\omega * MS + \theta * T_{delay}) \quad (3.1)$$

Where MS denotes the makespan of the schedule and T_{delay} denotes the cumulative delay in meeting deadlines. ω $(0 \leq \omega \leq 1)$ and θ $(0 \leq \theta \leq 1)$ are the weights to prioritize the components of the fitness function as per the needs of the resource provider and job users. The components MS and T_{delay} of the fitness function are defined as follows.

Let $t_{end}(i)$ and $t_{dline}(i)$ be turnaround time and deadline time for i_{th} job, then

$$t_{delay}(i) = t_{end}(i) - t_{dline}(i) \quad (3.2)$$

{t_{delay} is delay in meeting deadline for i_{th} job, and

$t_{delay} = 0$ *if* $t_{end} < t_{dline}$ } and

$$T_{delay} = \Sigma t_{delay}(i) \qquad for\ 1<i<N \tag{3.3}$$

Makespan of schedule *S* is defined as

$$MS = Max\ (t_{processing}(k)) \qquad for\ 1<k<M \tag{3.4}$$

{where $t_{processing}(k)$ is the time in which processor P_k will complete processing jobs assigned to it.}

All the proposed algorithms, discussed in next sections, will try to maximize the fitness function by minimizing the *MS* and T_{delay}.

3.3 SexualGA based Resource Scheduling Algorithm (SGASchedule)

Genetic algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Inspired by Darwin's theory about evolution, Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection i.e. survival of the fittest. Use of Genetic Algorithms (GAs) [82] in mapping jobs to resources, allows good solutions to be found quickly. So it allows the resource scheduler to be applied to more general problems. Many researchers have investigated the use of GAs to schedule tasks in homogeneous [82], [83] and heterogeneous [84]-[87] multi-processor systems with notable success. This section discusses the proposed SexualGA [89] based scheduling algorithm.

3.3.1 SexualGA

SexualGA [89] is an enhanced selection scheme that tries to mimic the sexual selection in human beings. In that context sexual selection is described as the concept of male vigor and female choice, meaning that male individuals try to spread their

gene material as wide as possible and female individuals are more selective by choosing rather above average fit males to guarantee a high survival probability of their off-springs. So it seems to be preferable not to use identical selection mechanisms for male and female population members also in GAs. Inspired by this view of sexual selection, a new selection mechanism for GAs (SexualGA) [89] is presented by introducing two different selection operators, one for the selection of male and one for the selection of female individuals.

3.3.2 Chromosome Representation

Schedule, *S,* of independent jobs is represented by a chromosome. Chromosome is basically a $N \times 1$ vector, where the position i ($0 < i < N$) represents the job / task and the entry at position i ($s[i]$) is the processor / machine to which the job has been assigned. For example, in Figure 3.1, $S\ [1] = P_2$, means job J_1 is assigned to Processor P_2.

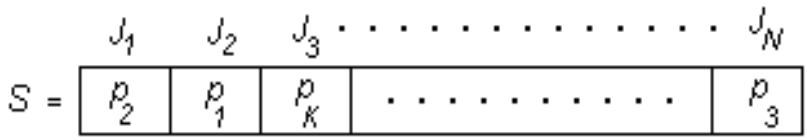


Figure 3.1: Chromosome representation

3.3.3 Initial Population

Initial population can be generated either randomly or by seeding. In the first method all the chromosomes are generated randomly from a uniform distribution where as in later some of the chromosomes in the initial population are generated using best-known techniques and others are generated using random generation. In the proposed algorithm seeding approach is used, where one chromosome is Min-min solution and rest is generated at random.

3.3.4 Selection

Genetic Algorithms use a selection mechanism to select individuals from population to insert into a mating pool. SexualGA [89] is new selection mechanism, which is used in proposed SexualGA based scheduling algorithm. The main idea of this scheme is to use two different selection schemes for selecting two parents required for each cross-over. The concept of male vigor and female choice are simulated using *Roulette Wheel Selection (RWS)* as the first selection scheme and *n-Tournament Selection* as the second selection scheme. So this selection mechanism has the advantage of not losing the genetic diversity and hence better control over the selection pressure.

3.3.5 Crossover and Mutation

Crossover operation is performed to generate new off-springs for the next generation. Two parents for the crossover are selected using the above-mentioned selection scheme and two-point crossover is performed. Using the vector representation for the chromosomes as shown in Figure 3.1, crossover operation is similar to exchanging the contents of two vectors for all positions before first crossover point and after the second crossover point. *Mutation* in the vector representation is considered as moving a job from one processor to another or swapping of two Jobs. For moving a job from one processor to another, two random numbers l and k are generated such that $S[l] = k$, where $0<l<N$ and $0<k<M$.

3.3.6 Exit criteria

An exit criterion specifies when to stop further exploration, and accept the solution. Proposed algorithm uses two exit criteria. 1) Standard deviation of the fitness value of the individuals in the population and 2) No change in the elite

chromosomes after 50 iterations. After the fitness function for all the chromosomes in the population is calculated, standard deviation is computed. If the standard deviation reaches a sufficiently low threshold value or there is no change in the elite chromosomes after 50 iterations, solution is said to converge.

3.3.7 Building the new population

New population is build from previous population through the crossover, mutation and elitism. This process is executed as follows. X_c % of the chromosomes, for the new population, is constructed using the crossover process as described earlier. X_m % of the chromosomes is constructed by applying the mutation operator. After the application of crossover and mutation, the rest of the chromosomes are taken out of the existing population by making them identical to existing population's best chromosomes, so as to make the size of new population equal to the existing one.

3.3.8 Algorithm (Pseudo Code)

```
// Initialize the variables
Initialize numberOfIterations
Initialize populationSize
Initialize Xc
Initialize Xm
Initialize threshHoldSD

// Array to hold elite chromosomes for population
EliteList eliteChromosomes[numberOfIterations]
// Generate initial population
pop0 = GenerateInitialPopulation()
// Calculate the fitness of individuals in population
pop0.CalculateFitness()
// Calculate Standard Deviation
pop0.CalculateSD()
done = false

// Evolve new generations
While (!done)
{
  // Generate next Population
  // perform crossover
  for ( i = 0 ; i < populationSize * Xc  ; i++)
```

```
  {
    // select first parent using Roulette wheel
    //selection
    parent1 = pop0.SelectParent(Type.RWS)
    // select 2nd parent using Tournament Selection
    parent2 = pop0.selectParent(Type.TOURNAMENT)
    // perform crossover and add to new population
    pop1.crossAndAdd(parent1, parent2)
  }

// perform mutation
  for ( i = 0 ; i < populationSize * Xm  ; i++)
  {
    pop1.mutateAndAdd(pop0)
  }
  // Add elite chromosomes
  pop1.addEliteChromosomes(pop0, eliteChromosomes)
  // Calculate fitness and standard deviation
  pop1.CalculateFitness()
  pop1.CalculateSD()

 // Check the criteria
  done = criteriaReached(pop1,
                   eliteChromosomes,
                   numberOfIterations,
                   threshHoldSD)
  pop0 = pop1
  numberOfIterations = numberOfIterations + 1
}
```

3.4 ACO based Resource Scheduling Algorithm (ACOSchedule)

Ant Colony Optimization (ACO) is a paradigm for designing meta-heuristic algorithms for combinatorial optimization problems. It is a probabilistic technique that was initially proposed by Marco Dorigo in 1992 in his Ph.D. thesis [90]-[93] to search for optimal path in the graph; based on the behavior of ants seeking a path between their colony and a source of food. Since then this idea was used to solve many compute intensive problems. In this section proposed grid scheduling algorithm based on ACO is discussed in detail. Many attempts have been made to apply ACO to scheduling problems [94]-[96]. Proposed algorithm differs from the other algorithms in two ways: 1) the objective function takes care of the interests of both, grid users

and resource providers as discussed in section 3.2.1 and 2) It uses the genetic operators to further optimize the results at the end, before the start of next iteration.

3.4.1 Pheromone Trail Definition

In ACO based algorithms ant's search is highly influenced by the pheromone trails, $\tau(i, j)$, left by other ants. So the definition of pheromone trail is one of the important decisions in ACO algorithms. Since grid is a heterogeneous environment, and some jobs may run more efficiently on some processors than others. So the pheromone will be stored in $N \times M$ matrix, which will have single entry for job processor pair in the problem. This value will represent the favourability of scheduling the job i on processor j.

3.4.2 Heuristic Information

One characteristic of a job that is very important is the critical time ($t_{critical}$) of a job. This is the minimum time in which a job can be processed if all the required resources are made available without any delay. Since the lower value is preferred so the inverse of this will be used.

$$\eta(i) = 1/(t_{critical}) \quad \{\text{where } t_{critical} \text{ is the critical time for job } i.\ \} \tag{3.5}$$

3.4.3 Pheromone Trail Updating

As mentioned above the pheromone matrix will have an entry for each job-processor pair in the problem, and each such pair in S_{best} 's solution (S_{best} is iteration's best solution) is reinforced in proportion to the relative fitness value of S_{best} compared to the global best solution, $S_{gb.}$ (S_{gb} is initially set to the Min-min solution). ρ is the rate of pheromone evaporation, where $0 \leq \rho \leq 1$. Hence

$$\tau(i,j) = \rho \cdot \tau(i,j) + F(S_{best}) / F(S_{gb}) \tag{3.6}$$

3.4.4 Building the solution

A specified number of ants are initialized and start building their tours in search of an optimal solution. The Job *j* is allocated to the processor *p*, in an arbitrary order, based on the pheromone value between *j* and *p* and critical time of *j*. The probability of selecting job *j* to be scheduled next is given by Equation 3.7.

$$Prob(j) = \frac{[\tau(j, p^{j}_{min})]^{\alpha} \cdot [\eta(j)]^{\beta}}{\sum^{n}_{i=1} [\tau(i, p^{i}_{min})]^{\alpha} \cdot [\eta(i)]^{\beta}} \quad (3.7)$$

where α and β are used to scale the pheromone trail and heuristic value respectively. If $\alpha = 0$, the decision is purely based on heuristic information and if $\beta = 0$ then the selection is purely on the basis of pheromone information for the job processor pair.

A job is then selected based on this value, and the chosen job *j* is then allocated to the processor, $p^{j}_{min,}$ on which it has minimum $t_{critical.}$ The schedule, found by an ant, at the end of iteration, is encoded in a vector *S*, of size *N*, where *N* is the number of jobs that are to be executed. *S[j]* indicates the machine where job *j* is assigned. This process is repeated until all jobs have been scheduled and a complete solution has been built. The pheromone trail update procedure is then used on the iteration best ant.

3.4.5 Optimizing ACO by Genetic Operators

At the end of iteration, when all the ants have built their tours, pheromone is updated by iteration's best ant. As the pheromone is deposited, the job processor pairs with more pheromone are more likely to be selected every time as compared to the other job processor pairs, with less pheromone value. So it may stop the further exploration of search space and solution may be limited to local minima. Hence, in

order to balance the load properly and to stop the solution to stuck to local minima, genetic operators are used.

The solution is represented in the form of chromosome by the same vector *S*, prepared by the ants while building their tours. The population size is equal to the number of ants. The solutions found by all the ants in the current iteration are the members of the population. The chromosomes are ordered in the non ascending order of the fitness of the solution. Crossover is performed selecting one parent from a specified percentage (*x* %) of chromosomes from lower end (solutions with lower fitness value) of the list and other parent from rest of the population. After performing few cross over operations a new population is prepared with *x* % new chromosomes and rest taken from the earlier population as it is. Again the fitness of all the solutions is taken and *SD* is computed. If there is any decrease in the *SD* for the population then the best ant from the new populations is used to update the pheromone trail other wise the iteration best ant, before applying the genetic operators is used to update the pheromone.

Mutation operator can also be applied. Mutation in the vector representation is considered as moving a Job from one machine to another or swapping of two Jobs. For moving a job from one machine to another, two random numbers l and k are generated such that $S[l] = k$, where $0<l<N$ and $0<k<M$. A limited number of crossover *(10 – 15)* or mutation operations (with least mutation probability) are performed so as not to loose the objective of finding an optimal solution in time.

Consider a simple example of mutation operator which optimizes the solution by decreasing the make span of the solution. Assume there are four jobs j_1, j_2, j_3, j_4 to be scheduled on processors p_1 and p_2. The completion time $t_{complete}$ of these jobs is as given in Table 3.1.

	j_1	j_2	j_3	j_4
p_1	3	4	2	3
p_2	4	3	4	1

Table 3.1: $t_{complete}$ of j_1, j_2, j_3, j_4 on $p_1 p_2$

Assuming that after the completion of the tour, the schedule S generated by an ant is represented by a vector as shown in Figure 3.2. Jobs j_1 and j_2 are to be processed on processor p_1 and jobs j_3 and j_4 are to be processed on p_2 as per this schedule, the makespan of the schedule is 7.

j_1	j_2	j_3	j_4
p_1	p_1	p_2	p_2

Figure 3.2: Schedule S before applying mutation operator
(Makespan = 7)

Now if the mutation operator is applied to this solution and a swap is performed in j_2 and j_3, the new schedule will be represented by vector S' as shown in Figure 3.3.

j_1	j_2	j_3	j_4
p_1	p_2	p_1	p_2

Figure 3.3: Schedule S' after applying mutation operator.
(Makespan = 5)

So the makespan for this schedule S', after the application of mutation operator, is 5 which is reduced as compared to previous solution formed by the application of pure ACO. Hence the use of genetic operators helps in exploring the complete search space and finding the optimal solution.

3.4.6 Algorithm (Pseudo Code)

```
// Initialize Variables

initialize NumberOfIterations
initialize NumberOfJobs
initialize NumberOfResources
initialize ResourceList[NumberOfResources]
initialize JobList[NumberOfJobs]
initialize MaxAnts
```

```
// Declare pheromone as 2d array
Initialize Pheromone[NumberOfJobs][NumberOfResources]

// Declare Solution pool to hold solution of all ants.
initialize SolutionPool[MaxAnts]

// input jobs and resource list
JobList = getJobsToSchedule()
RessoureList = getAvailableResources()

// Initialize global best solution as Min min solution
Initialize sGlobalBest[NumberOfJobs]
sGlobalBest = GenerateMinMinSchedule(JobList,
                                     ResourceList)

Repeat for every Ant
{
  while ( there are unscheduled Jobs in JobList )
  {
     For (every Resource in the resource list)
     {
     -  Get Next unscheduled Job.
     -  Compute probability to schedule on current
        resource.
     -  schedule the job on to the resource based on
        computed probability
     }

  }
  Add the Solution S to SolutionPool
}

// Find the iteration best solution in the solution pool
sIterationBest = findBestSolution (SolutionPool)

// Apply genetic operators find new iteration best
// solution
newSolutionPool = ApplyGeneticOpeartors (SolutionPool)
newIterationBest = findBestSolution(newSolutionPool)

if (computeFitness(sIterationBest) <
        computeFitness(newIterationBest))
{
   sIterationBest = newIterationBest;
}

// Update phreomone based on global best and iteration
// best solution
UpdatePheromone(Pheromone,
          sGlobalBest,
          sIterationBest);
```

```
// Update Global Best Solution if Required.
if(computeFitnesss(sIterationBest) >
            computeFitness(sGlobalBest))
{
   sGlobalBest = sIterationBest;
}

Repeat above steps for every iteration.

Output sGlobalBest as the final solution.
```

3.5 Hybrid Genetic Simulated Annealing based Grid Scheduling Algorithm (HybridGSA)

Genetic Simulated annealing heuristic is a hybrid of Genetic Algorithms and Simulated Annealing (SA) techniques. SA [97]-[102] is an iterative technique that considers only one possible solution at a time. SA uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search of the solution space. This probability is based on a system temperature that decreases at every iteration. As the system temperature cools, it is more difficult for poorer solutions to be accepted. GA and SA both independently are valid approaches towards problem solving. However they have their own strengths and weaknesses. While GA can begin with a population of solutions in parallel, it has poor convergence properties. On the other hand SA has better convergence properties but cannot exploit parallelism.

Proposed HybridGSA algorithm provides a hybrid approach for the scheduling jobs on the grid. Hence HybridGSA algorithm maintains the diversity of GA while reducing the processing effort by following the temperature schedule as in SA. Since whole flow is same as SGASchedule algorithm described in section 3.3, so here we only describe parts that are different from SGASchedule.

3.5.1 Initial Temperature and Annealing Schedule

Initial temperature, *T*, is set to a value such that the acceptance probability for the chromosomes of the population is very high i.e. close to *1.0*. The annealing schedule is generated by multiplying the current temperature by a constant less than *0*, which is known as the cooling rate, Υ. The probability of accepting new solution in reproduction (crossover and mutation) is implemented as a function of temperature, *T*. So initially the new off-springs are accepted with a high probability, where as, in the later stage their probability of being accepted is lowered as the temperature decreases. Temperature, *T,* is decreased at cooling rate, Υ, after every iteration and is given by Equation 3.8.

$$T_i = \Upsilon * T_{i-1} \quad \{\text{where } i \text{ is the iteration number}\} \tag{3.8}$$

3.5.2 Crossover

Crossover operation is performed to generate new off-springs for the next generation. Two parents for the crossover are selected using the above-mentioned selection scheme and two-point crossover is performed. Using the vector representation for the chromosomes as shown in Figure 3.1, crossover operation is similar to exchanging the contents of two vectors for all positions before first crossover point and after the second crossover point. After the generation of new off-springs, change in the energy, δE, is computed using the Equation 3.9 and their probability, *P*, of acceptance is given by Equation 3.10.

$$\delta E = f_{offspring} - f_{Parent} \tag{3.9}$$

where $f_{parent} = Max\ (f_{Parent1}, f_{Parent2})$, Maximum fitness of two parents

$f_{offspring}$ = Fitness of the new offspring

$$P = exp\ (\ -\delta E / T)\quad \{P = 1\ if\ \delta E > 0\} \tag{3.10}$$

After the finding the probability of acceptance, a uniform random number, μ, is generated between *0* and *1*. New offspring is accepted if, $P >= \mu$, otherwise the new offspring is rejected.

3.5.3 Mutation

Mutation in the vector representation is considered as moving a job from one resource to another or swapping of two Jobs. For moving a job from one resource to another, two random numbers *l* and *k* are generated such that *S[l] = k,* where $0<l<N$ and $0<k<M$. After applying the mutation, the change in energy, δE, is computed using Equation 3.9 and probability of acceptance, *P*, is computed using Equation 3. 10. If, $P > \mu$, (a uniform random number between *0* and *1*), the new offspring is added to the next population, otherwise it is rejected.

3.5.4 Algorithm (Pseudo Code)

```
// Initialize the variables
Initialize numberOfIterations
Initialize populationSize
Initialize Xc
Initialize Xm
Initialize threshHoldSD
Initialize SystemTemprature
Initialize CoolingRate
// Array to hold elite chromosomes for population
EliteList eliteChromosomes[numberOfIterations]
// Generate initial population
pop0 = GenerateInitialPopulation()
// Calculate the fitness of individuals in population
pop0.CalculateFitness()
// Calculate Standard Deviation
pop0.CalculateSD()
done = false
// Evolve new generations
While (!done)
{
```

```
  // Generate next Population
  // perform crossover
  for ( i = 0 ; i < populationSize * Xc  ; i++)
  {
    // select first parent using Roulette wheel
    //selection
    parent1 = pop0.SelectParent(Type.RWS)
    // select 2nd parent using Tournament Selection
    parent2 = pop0.selectParent(Type.TOURNAMENT)
    // perform crossover and add to new population
     offsprings[2] = parent1.cross(parent2);
     // Find out the max fitness of two parents
     fParent = Max(parent1.fitness(), parent2.fitness());
     // Check if first offspring is to be added to new
     // population
     DeltaE = offspring[0].fitness() - fParent;

     if (DeltaE > 0 ) P = 1;
     else  P = exp ( -DeltaE/T);

     if ( P > Random.Getdouble() )
     {
          pop1.Add(offspring[0]);
     }
     // Check if second offspring is to be added to new
// population
     DeltaE = offspring[1].fitness - fParent;
     if (DeltaE > 0 ) P = 1;
     else P = exp (-DeltaE/T);

     if ( P > Random.Getdouble() )
     {
          pop1.Add(offspring[1]);
     }

}

// perform mutation
  for ( i = 0 ; i < populationSize * Xm  ; i++)
  {
     offspring = parent1.mutate();
     DeltaE= offspring.fitness() - parent1.fitness();
     if (DeltaE > 0 ) P = 1;
     else P = exp (-DeltaE/T);

     if ( P > Random.Getdouble() )
     {
          pop1.Add(offspring);
     }
  }
```

```
  // Add elite chromosomes
  pop1.addEliteChromosomes(pop0, eliteChromosomes)
  // Calculate fitness and standard deviation
  pop1.CalculateFitness()
  pop1.CalculateSD()
  // Check the criteria
  done = criteriaReached(pop1,
                   eliteChromosomes,
                   numberOfIterations,
                   threshHoldSD)
  pop0 = pop1
  numberOfIterations = numberOfIterations + 1

  // Lower the temperature
  SystemTemprature = CoolingRate * SystemTemprature;
}
```

3.6 Summary

This chapter starts with the identification of Quality of Service (Qos) parameters and then formulates an objective function based on these QoS parameters. This objective function tries to take care of interests of users by minimizing the cumulative time delay and also tries the maximum utilization of resources by minimizing the make span of the schedule. Further three new scheduling algorithms, SGASchedule, ACOSchedule and HybridGSA are proposed and discussed in detail. Next chapter provides a detailed discussion on the proposed ACO guided mobile agents based resource discovery algorithm.

4

Proposed Algorithm for Grid Resource Discovery

4.1 Introduction

Resource discovery in the computational grids is a challenging problem because of the dynamic nature of user requests as well as dynamic availability of resources and their heterogeneous nature [104]. Growing size of grid makes it essential for the grid resource discovery mechanisms to be highly scalable and de-centralized [105]. P2P resource discovery systems [106][107] provide a good example of de-centralized resource discovery. Software mobile agents [115][116], with their autonomous behavior, intelligent decision making capability and mobility are

particularly suitable for tasks such a de-centralized resource discovery. At the same time ants (natural mobile agents) based algorithm provide a mechanism for building self-adaptive systems. This paper presents ants colony optimization based algorithm, which is used to guide software mobile agents in efficient resource discovery.

Many resource discovery mechanisms have been proposed in the literature of Grid environments. Some of them, such as MDS2 (Meta-computing Directory Service) [108] and Data Grid Resource Discovery [109], are specific resource discovery services. However, the great majority of them are contained in more general grid proposals such as MDS3 (Monitoring and Discovery Services) [110], NWS (Network Weather Service) [111], VIRD (Vega Infrastructure for Resource Discovery) [112], and Nimrod/G [59].

The resource discovery mechanisms proposed for grids are either hierarchical or centralized. Most of these mechanisms retrieve data related to the machines that compose the grid (operating system used, CPU load, and memory occupation, among others). But for large scale distributed environment like grid the centralized approach is not very much suitable. Hence a de-centralized approach is desirable [112].

As a solution to this problem an ACO based algorithm has been proposed. Mobile agents are used to simulate this ant like natural agents. This further adds the capabilities of mobile agents like distributed and asynchronous processing etc. to this approach. The terms ants, mobile agents, query agents and natural agents are used interchangeably in this thesis.

4.2 Problem formulation

To formulate the problem we consider the whole grid as a network of virtual organizations (VOs) [1] with each VO having a designated contact node. Connection between the VOs is via contact nodes only as shown in Figure 4.1).

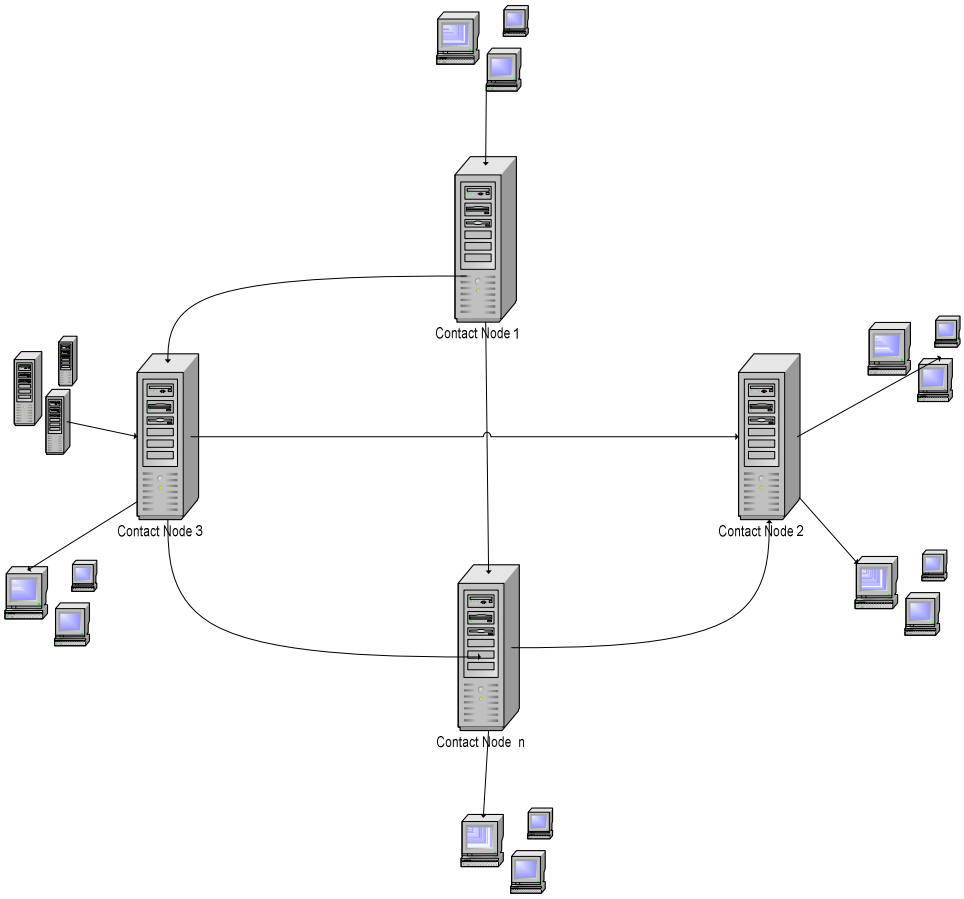


Figure 4.1: High level view with Peer Contact Nodes

The subnet of contact nodes can be represented by a directed graph. Machines/ contact nodes are represented by nodes in the graph and an edge between two nodes represent a relation between a pair of machines / contact nodes (an edge between nodes *A* and *B*, means machine *A* has knowledge of node *B*). Communication can only happen between the machines that know each other, rather than some broadcast message from central machine to which every machine responds. The problem is to find the set of matching resources that can full fill the request. The resource request is specified in the form of resource attributes / characteristics that need to be matched. The set of resource attributes that is used to find matching resources is also known as resource query.

4.3 Mobile Agents

Software agent is a computing entity that performs some task or tasks on behalf of somebody [113]-[115]. A mobile agent is a software agent that has an additional property that it is not bound to operate only in the system in which it started. One special thing about mobile agent is that during its lifetime it can be stopped at one point, its state can be saved and code can be moved to another machine and then can resume its execution. Mobile agents have many benefits over other technologies [116].

Mobile agents can reduce the need for bandwidth. They can package the interaction between two machines as a discrete packet of network traffic and helps to make all interactions locally rather than having all individual interactions over the network [117]. Hence MAs reduce the network traffic by moving the processing to raw data instead of moving raw data to processing [118]-[121]. Also they can grow dynamically to accommodate more data. Mobile agents can be dispatched asynchronously. Autonomy in the MAs enables them to take dynamic decisions based on the previous learning. MAs can be easily cloned and dispatched too and finally, mobile agent based solution are very fault tolerant. If some MA gets destroyed in between, the results from other surviving MAs can still be used. So they can be used to benefit or to simplify different types of application areas. Some examples of these application areas include e-commerce, distributed information retrieval, telecommunication networks services, and monitoring and notification [117].

4.4 Proposed Algorithm for Resource Discovery based on ACO guided Mobile Agents (ACORD)

This section discusses the ACO guided mobile agents based proposed resource discovery algorithm for computational grids. Ants are simulated using Mobile agents

and pheromone trails laid down by ants are used to guide the mobile agents in their search. This provides the basis for efficient and de-centralized resource discovery mechanism for computational grids. Different components of this algorithm are described below.

4.4.1 Fitness function

Fitness function is used to differentiate between high and low quality solutions. It is defined in terms of objectives. For resource discovery our objective is to find the set of resources matching to the query made by the grid resource management system. So we define our fitness function as the ratio of resources found to the resources requested.

Let Q denotes the query, i.e. the set of the resources requested, and M_N is the set of matching resources found till Node N. Then the fitness function, $F_N(Q)$, at any Node N is defined as

$$F_N(Q) = \xi(M_N) / \xi(Q) \tag{4.1}$$

where $\xi(M_N)$ denotes the resources in Set M_N

As the ants move from one node to another, the fitness function, $F_N(Q)$, is evaluated at every node N, pheromone is deposited based on the fitness and Ants move to the next node till the exit criteria is reached.

4.4.2 Pheromone Trail Definition

In ACO based algorithms ant's search is highly influenced by the pheromone trails, $\tau(N)$, left by other ants. So the definition of pheromone trail is one of the important decisions in ACO algorithms. Since, grid is highly distributed and heterogeneous environment, so different nodes may have different availability of

resources. So the pheromone trail, $\tau(N)$ will be representing the Query Hits along the path of node *N*.

4.4.3 Heuristic Information

Heuristic information, $\eta(N)$, in the ACO algorithms is another factor that affects the move of an ant. This is the piece of information other than the pheromone trail that decides the probability of the node to be chosen next. In our problem we define this as the number of resources available at node *N*. Hence

$$\eta(N) = \xi(R_N) \tag{4.2}$$

where R_N is the set of available resources at node *N*.

$\eta(N)$ is normalized with respect to the availability of resources in the neighborhood of node *N*.

4.4.4 Pheromone Trail Updating

After every successful search pheromone trail needs to be updated along the search path so as help other ants to find resource quickly. The amount of pheromone to be deposited at any node *N* is defined as

$$\tau(N) = \rho \cdot \tau(N) + F_N(Q) / F_{end}(Q) \tag{4.3}$$

where $F_N(Q)$ is the fitness at Node *N* for query *Q* and $F_{end}(Q)$ is the fitness at the last node in the search Path and ρ is the rate of pheromone evaporation.

4.4.5 Building the Solution

A specified number of ants are initialized and start building their tours with respect to the query *Q*. At any Node *I*, the probability of moving to the next node *K* is given by Equation 4.4.

$$Prob(K) = \frac{[\tau(K)]^{\alpha} \cdot [\eta(K)]^{\beta}}{\sum^{n}_{i=1} [\tau(i)]^{\alpha} \cdot [\eta(i)]^{\beta}} \quad (4.4)$$

where *n* is the number of nodes in the forwarding table of Node *I*.

New node *K* is selected according to the probability calculated using the equation 4.4 and ant is moved to next node. This process is repeated till the fitness at newly selected node exceeds the threshold value ("query hit"). As soon as the ant reaches at the node where there is query hit, the ant performs the backwards journey from final / end node to the source node and on its way back, it keeps on updating the pheromone trail on all the nodes in its path. On the other hand if the ant has visited maximum nodes, it simply returns to the source node with a "query miss" without updating the pheromone trails.

4.4.6 Exit Criteria

An exit criterion specifies when to stop further exploration. Proposed algorithm uses two exit criteria. 1) If the fitness at any node *I* exceeds fitness threshold, which is a "query hit" or 2) If ant has visited maximum number of nodes as specified, this is a "query miss". In either of the cases further exploration is stopped.

4.5 Algorithm to be executed by Query Agent

During the process of resource discovery, a Query Agent goes through different states. States diagram shown in Figure 4.2 show the transition of Query agent to different states.

I. Query Agents are created by a contact node in VO and initialized with the query according to which they have to search for resources. Query is specified in terms of resource characteristics.

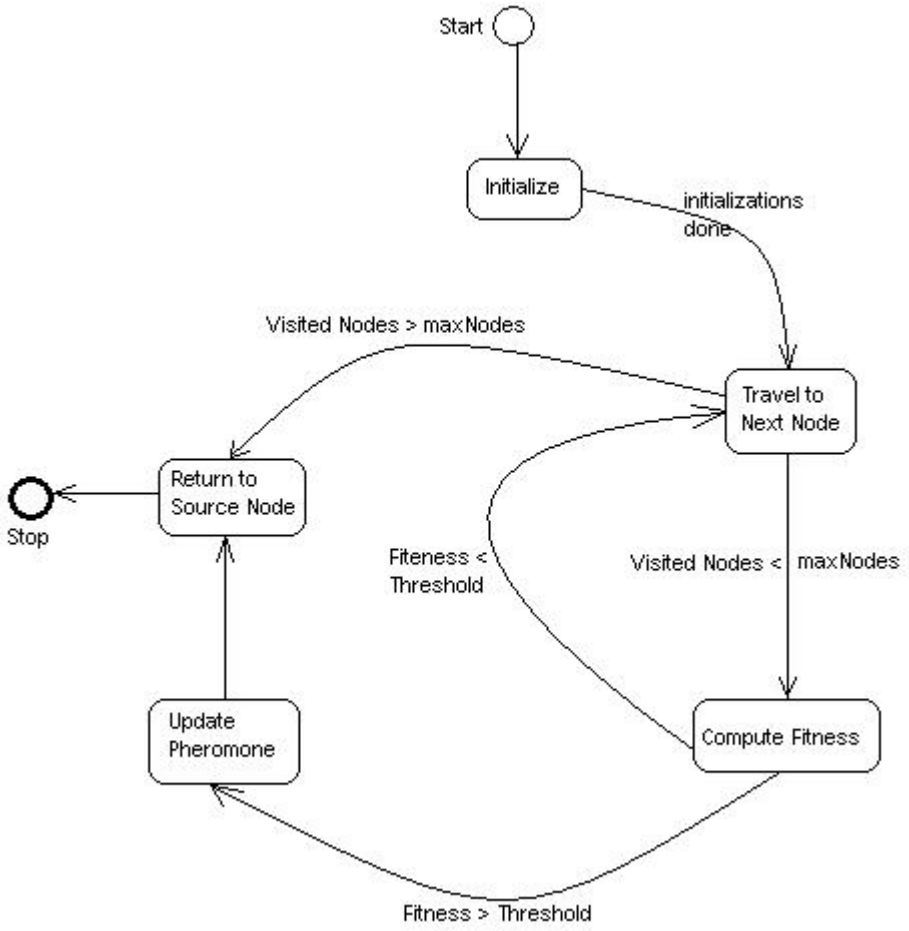


Figure 4.2: Different States of a Query Agent

II. *Branching factor* limits the number of query agents to be created for a specific query.

III. QA is also initialized with *MaxNodes* that it can travel at most. *MaxNodes* is the maximum number of nodes a QA is allowed to visit. It ensures that a QA should return in finite time to source contact node.

IV. After arriving at any node it increments the *Visited Nodes* count, check for the resources at the current node and computes the fitness and accordingly take the decision to move to next node or return to source node as per the algorithm mentioned in the previous section.

V. After submitting the results to the source contact node (the creator of QA) it dies.

VI. Source contact node collects results (resource characteristics) from all the QAs dispatched, and allocate resources to the different jobs as per the requirement.

VII. Pheromone trails are also evaporated after regular interval of time to avoid saturation.

4.5.1 Algorithms (Pseudo Code)

```
// Initialize Variables
initialize BranchingFactor
initialize MaxNodes
initialize Threshold

// Create Query Agent Array
QueryAgentList QueryAgent[BranchingFactor]

// Initialize Query Agents
for ( i = 0; i < BranchingFactor; i++ )
{
     QueryAgent[i].initialize();
}

Repeat following steps for Every QueryAgent[i]
{
     Repeat at every Node
     {
          //Increment VisitedNodes
          QueryAgent[i].visitedNodes += 1;

          if ( QueryAgent[i].visitedNodes <= MaxNodes )
          {
               // Compute Fitness
               fitness=QueryAgent[i].ComputeFitness(Node);

               if ( fitness < Threshold )
               {
                    // Find Next Node.
                    NextNode=QueryAgent[i].findNode(Node);

                    // Travel to next Node
                    NextNode.Move(QueryAgent[i]);
               }
               else
               {
                    // Update Pheromone Trails.
                    QueryAgent[i].UpdatePheromone();
```

```
                    // Return to source Node. (Query Hit)
                    QueryAgent[i].returnToSource(HIT);

               }
          }
          else
          {
               // Return to Source Node (Query Miss)
               QueryAgent[i].returnToSource(MISS);

          }

     } // End Repeat at every node
}
```

4.6 Advantages of proposed ACO based Resource Discovery Algorithm (ACORD)

Proposed ACO based resource discovery algorithm (ACORD) provides several advantages over traditional grid resource discovery system. Some of them are list below.

De-Centralized System: Proposed algorithm provides a resource discovery model which employs the benefits of a peer to peer system and, hence, is de-centralized system. A grid is considered as a peer to peer network of virtual organizations (VOs) where the contact nodes are allowed to communicate with other peer contact nodes.

Scalability: As mentioned above, proposed system is considers grid as a collaboration of different virtual organizations. This can be viewed as graph of contact nodes. With the de-centralized control it can easily cope with the increased number nodes / addition of new virtual organizations.

Parallel resource discovery: More than one QA are generally dispatched corresponding to a query and all search in parallel through different paths in the network for matching resources.

Fault Tolerant: Since more than one QA are usually dispatched, so if some QA is crashed in between, the results from other QA can provide the solution without affecting the quality.

ACO guided search: Proposed model provides ACO guided search. QAs from previous searches deposit pheromone trails that guide next QAs to find the fruitful nodes.

4.7 Summary

Details of the proposed ACO guided mobile agent based resource discovery algorithm are provided in this chapter. This algorithm assumes a model where each virtual organization has a designated contact node and one or more virtual organization interact with each other to find a set of matching resources via contact nodes. Mobile agent contains the query and where as pheromone trails laid down by ants provides the learning and aid in quick and efficient resource discovery.

5

Design and Implementation of Proposed Algorithms

5.1 Introduction

Chapter 3 and chapter 4 proposed the algorithms for grid resource scheduling and resource discovery. Details of design and implementation of proposed algorithms are discussed in this chapter. Two types of implementations are used to evaluate the proposed algorithms: 1) Grid Simulator based on GridSim toolkit, and 2) Grid Test bed based on the simulation model used Braun et. al.**[**21**].**

5.2 GridSim Toolkit

The GridSim toolkit [122]-[126] supports modeling and simulation of a wide range of heterogeneous resources, such as single or multiprocessors, shared and

distributed memory machines such as PCs, workstations, SMPs, and clusters managed by time or space-shared schedulers. That means, GridSim can be used for modeling and simulation of application scheduling on various classes of parallel and distributed computing systems such as clusters, Grids, and P2P networks. Application schedulers in Grid environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. Reference [101] provides a list of the salient features of GridSim. GridSim is a Java-based discrete event simulation toolkit. Java classes represent the entities essential for application, resource modeling, scheduling of jobs to resources, and their execution along with management. Figure 5.1 (taken from [125]) shows the flow diagram of GridSim based simulations. Visual Modeler [127] for GridSim provides a graphical user interface based tool that enables the users to specify inputs required for scheduling like jobs and resource characteristics etc., in an easy manner. A java program is automatically created by the Visual modeler for the simulation that can be executed using GridSim.

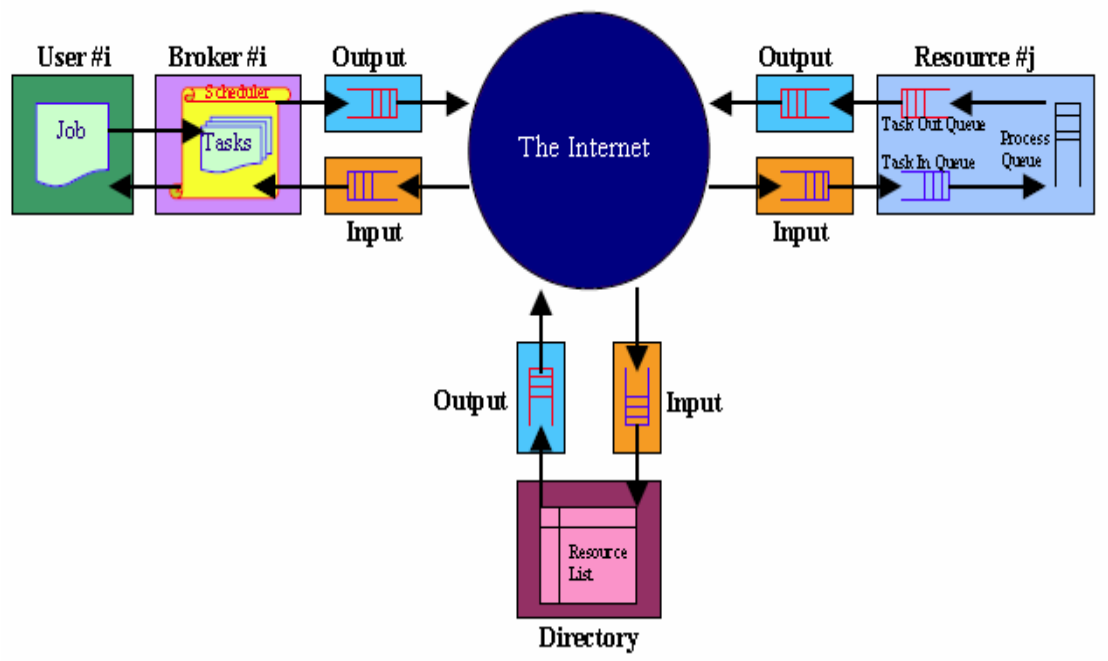


Figure 5.1: Flow diagram in GridSim based simulations [125]

5.2.1 Resource modeling

In GridSim toolkit, CPUs can be created (called as Processing Elements (PEs)) with different MIPS (Million Instructions per Second) or SPEC like ratings. One or more CPUs can be put together to create a machine. Grid resource is composed of such one or more machines. Grid resource can be a single processor, shared memory multiprocessors (SMPs) or a distributed memory cluster of computers and can be managed by time shared or space shared scheduling systems depending on the type of the resource. When a resource entity is created, it registers resource information and contact details to with the grid information service (GIS) entity. The GIS then can be queried for list of resources in a given Grid domain to get resource handles that can be used to query resources directly for their capabilities, costs and other configurations.

5.2.2 Application Modeling

Application model in the GridSim is generally defined by the developers (of schedulers / brokers). Each task in GridSim can be heterogeneous in terms of processing time and input files size, which is defined through a Gridlet object. A Gridlet is a tiny GridApp that contain all the information related to a job/ task and job execution management details such job's processing requirements, disk I/O operations, the size of input files etc. that help in computing the execution time of remote resource and the size of output files. Each grid user can be modeled with different characteristics like type of job created, scheduling policy, activity rate, time zone etc.

5.2.3 Steps for Simulating Application Scheduling

Simulating application scheduling with GridSim involves three high – level steps.

- ✓ First step is the creation of grid resources and gird user. Grid resources that need to be created can have different capability / speed, different time zones and different policies (space - shared or time - shared) as in the real environments. Similarly Grid users can be created with different requirements.
- ✓ Second step is the modeling of applications by creating a number of Gridlets (Gridlets can be considered as grid jobs) and define all the parameters in associated with the jobs. Grouping of Gridlets is also possible depending upon the application model for processing.
- ✓ Finally, last step is the implementation of resource brokers. Resource brokers first inquire the Grid Information System (GIS), then in quire for resource capability including cost and the depending on scheduling heuristics, strategy, or algorithms assign Gridlets to resources for execution. Scheduling policies can be system centric or user centric.

5.3 Implementation details of Grid Simulator based on GridSim Tool kit

Highly dynamic and heterogeneous nature of grid makes it impossible to create a repeatable and controlled environment for experimentation and evaluation of scheduling strategies. Hence a Grid Simulator is developed, which is able to deal with grid scheduling problems like job and resource heterogeneity and runtime changes like arrival of new job, joining / leaving of resources, due to dynamic nature of grid. is This Grid simulator is based on well-known GridSim tool kit This simulator implements a centralized Grid Scheduler, which uses the proposed meta-heuristics based scheduling algorithms (proposed in Chapter 3) for schedule generation.

5.3.1 Components of Grid Simulator

Grid simulator is a modular simulation environment. It consists of independent entities like centralized scheduler, submission system and grid resource, that corresponds to real world entities. Each submission system is associated with it a task generator that is actually used to simulate the job arrivals.

Submission System stores the incoming jobs as they arrive. Also it stores the jobs as they are returned after execution. It communicates with the scheduler to get scheduler information and then selects the resource on to which the job is to be submitted for execution.

Task Generator is associated with the submission system and is used to simulate task / job arrivals. The task arrival times correspond to the selected statistical distribution.

Scheduler is primarily responsible for schedule generation and optimization. Scheduler provides an interface which helps to plug-in the proposed scheduling algorithms for schedule generation. Scheduler has been designed taking the advantages of object oriented paradigm to make it reusable, easy to maintain and easy to integrate with other scheduling algorithms. One part of the scheduler maintains the communication with other components like submission system. Scheduler also implements the functions to estimate the required parameters like makespan, cumulative deadline delay etc. based on the generated schedule and information about the jobs currently under execution.

5.3.2 Class Diagram

Class diagram in Figure 5.2 shows the relation among different classes. *CGridSimulato*r is the main controller class that mimics the grid resource management system. This class contains the information of resources registered with

the grid system. Also, it has the information about the jobs that needs to be executed. Scheduler uses this information about the resources and jobs and estimates the schedule based on the proposed scheduling algorithms.

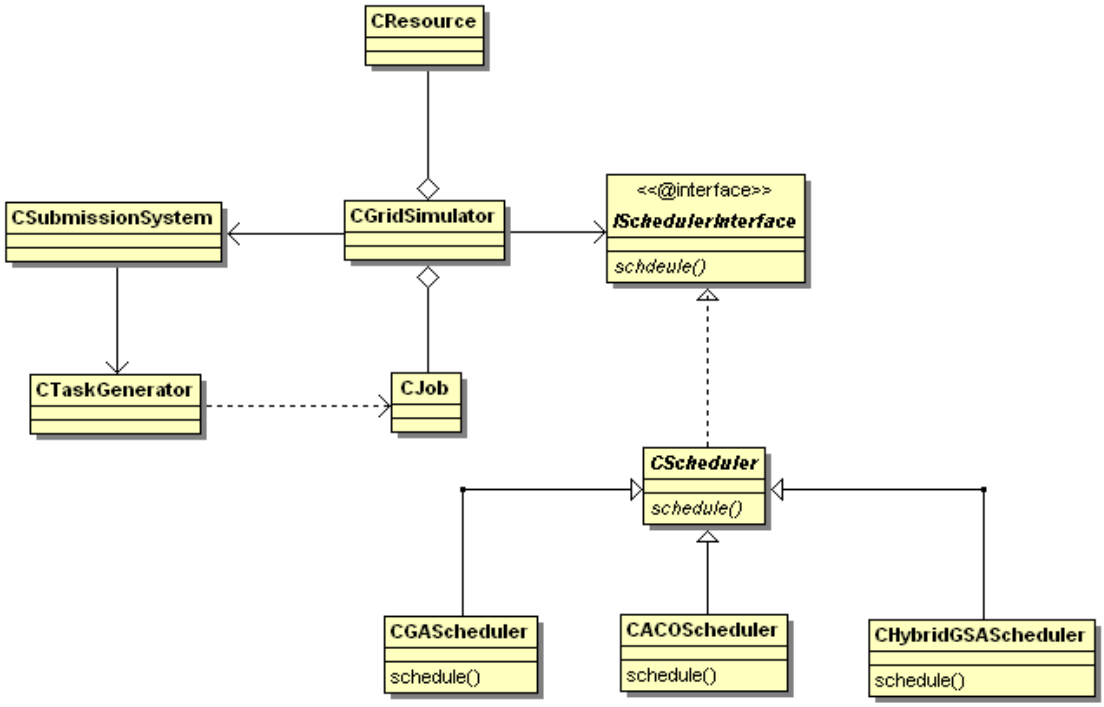


Figure 5.2: Class Diagram of Grid Simulator

CJob and *CResource* classes are used to represent the jobs/ task properties and resource characteristics respectively. Since the main aim of this grid simulator is to evaluate different scheduling algorithms, so this implementation is kept very modular. *CGridSimulator* class provides an interface, *IScheduleInterface*. To add a new scheduling algorithm, this interface needs to be implemented by the any child of the *CSchedule* class. *CSchedule* is an abstract scheduler class to implement the IScheduleInterface. *CGAScheduler*, *CACOSchedudler* and *CHybridGSAScheduler* are the concrete scheduler classes that actually implement the proposed algorithms SGASchedule, ACOSchedule, and HybridGSA respectively. Figure 5.3 shows the screen shot of the development environment for this grid simulator.

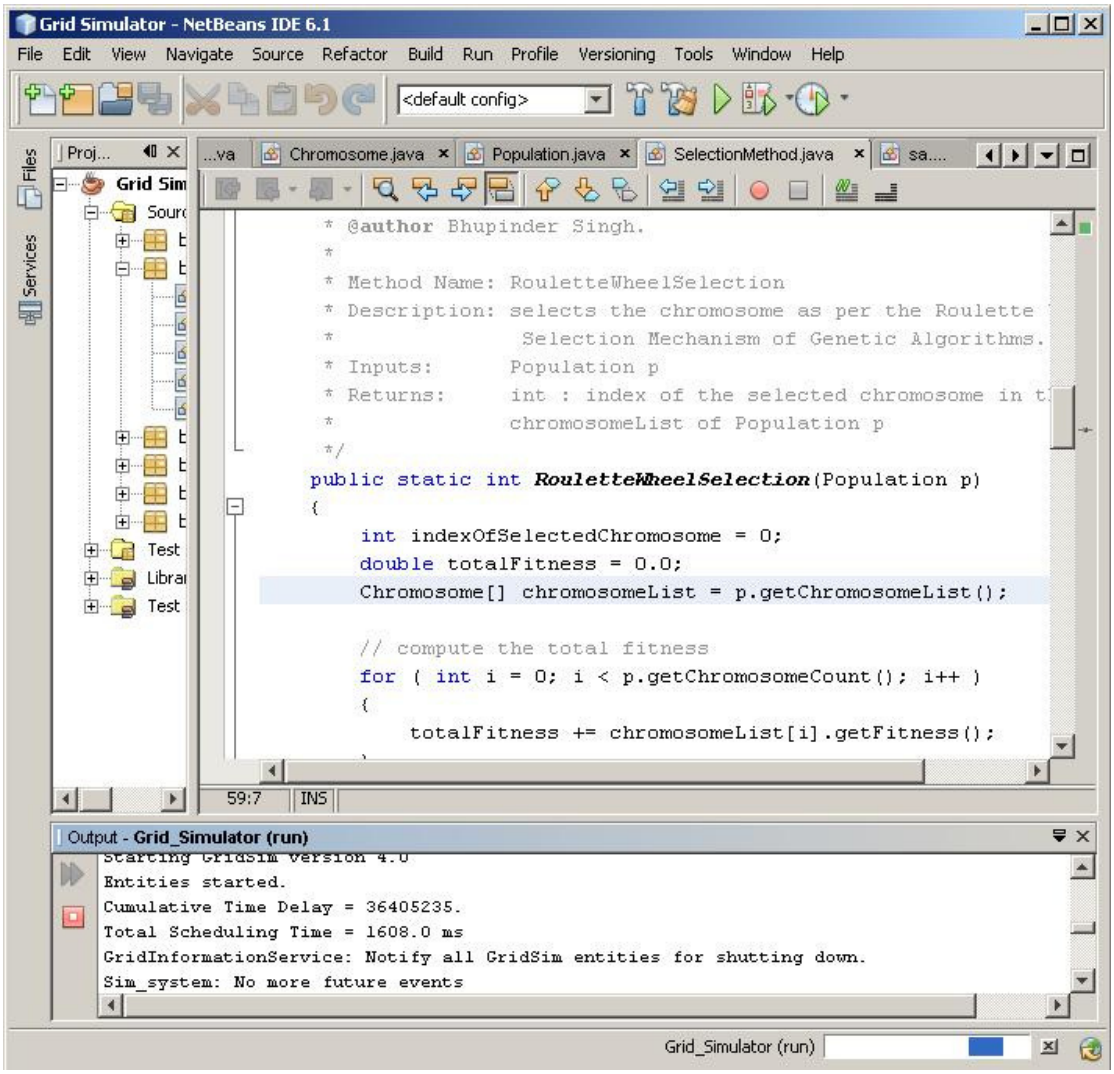


Figure 5.3: Screen shot of development environment for Grid Simulator

5.3.3 Communication Mechanism

The Figure 5.4 shows the communication among submission system, scheduler and grid resource. Submission system submits the job descriptions to the scheduler. The scheduler uses the list of all available grid resources and their characteristics like number of CPUs and their rating. Scheduler has the information about and access to all the available resources registered with the system. On the basis of this information the scheduler generates schedule for each grid resource. Using these schedules and also information of jobs currently in execution the scheduler is

able to approximate various parameters of the schedule such as makespan cumulative time delay for the jobs before their execution and completion.

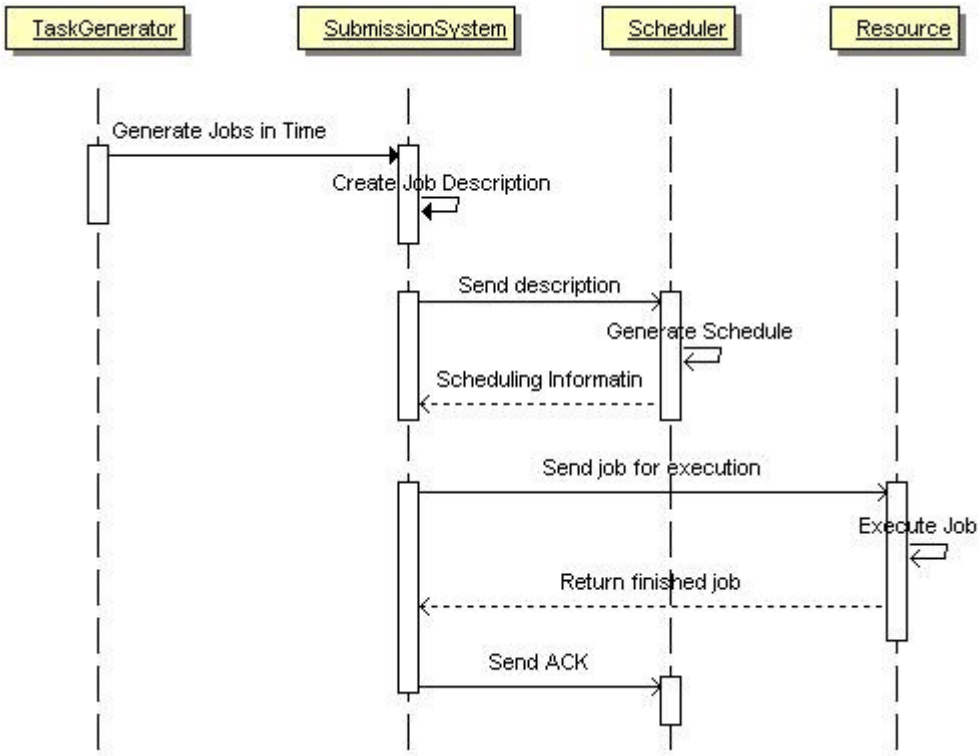


Figure 5.4: Communication among Grid Simulator components

According to the constructed schedule the scheduler responds to the submission system with scheduling information so that submission system is able to know which resource to be selected to execute a particular job. Whole communication among different components is asynchronous and is done via events. As soon as the new jobs are made available by task generator, their description is send by submission system to the scheduler. Submission system does not wait for the scheduler response. As soon as a job is finished an acknowledgement (*ACK*) is send to the scheduler. Scheduler then updates its internal information and current resource load. Pseudo code fro Submission system flow and scheduler flow are given in next sub-section.

5.3.4 Pseudo Code for Submission System Flow

```
1. Create new Gridlets / Jobs
2. Send the job description to scheduler for schedule
generation.
3. Wait for next event
     3.1 ACK (Acknowledgement):
             o Do nothing
     3.2 Gridlet_return:
             o Job execution complete
4. When All jobs finished
             o Send End_of_simulation to Scheduler
```

5.3.5 Pseudo Code for Scheduler Flow

```
1. Get information of registered resources
2. Wait for next event
     2.1 New_Gridlet:
             o Send ACK to Submission system
             o Add to batch of jobs
             o Find Schedule and execute
     2.2 Gridlet_Finished:
             o Lower Resource load
             o Return Job/Gridlet to submission system
     2.3 End_of_Simulation:
             o Get / Calculate Statistics
```

5.3.6 Extensibility

Grid simulator is modular and main functionality is divided into separate parts. So this is easy to extend, modify and maintain. With minor modifications, it is easier to simulate different type of jobs, scheduling algorithm and optimization criteria. To test any new scheduling algorithms, the only thing that needs to be done is to add a new class (e.g. *CNewScheduleAlgo*), which is derived from *CSchedule* class. This new class will override the method *schedule()* inherited from the base class. Similarly, to simulate the job/ task / Gridlet arrival times according to a different statistical distribution, task generator needs to be modified only. The rest of the classes remain intact so the experiments can be repeated with exactly the same setup. All the components interact with each asynchronously via events (refer to pseudo code in the previous sub-sections 5.3.4 and 5.3.5). So any change in the state machine of one

component does is independent and does not impact the state machine of other components. This makes the grid simulator closer to the real world.

5.4 Grid Test Bed

Grid Test bed provides the actual grid like environment with a simple web based user interface for submission of jobs, registering and de-registering of resources and administrative tasks. The Figure 5.5 shows the high level diagram of Grid Test Bed environment.

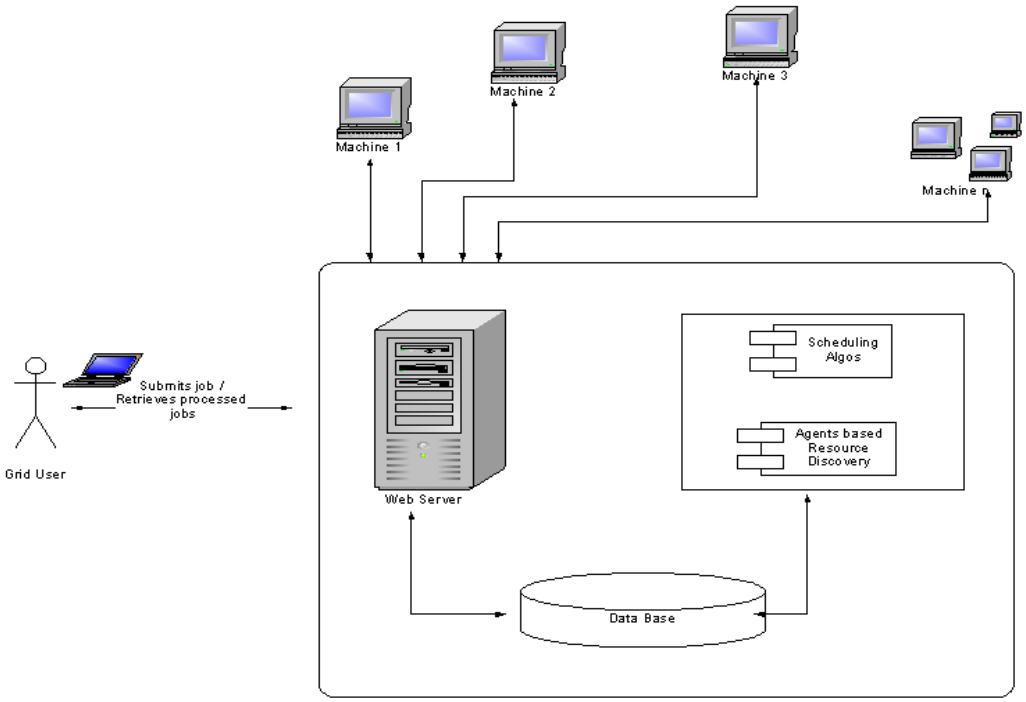


Figure 5.5: High level view of Grid Test Bed used for experiments.

Microsoft Internet Information Server is used as a web server that listens to the user requests provided to the system by web interface. The information provided by the user is stored in the database. User for the system can be grid users or resource owners. Grid users send the job requests for the jobs to be executed. Job characteristics are stored in the database. Grid users also specify the deadline time for the jobs. This is the time before which the job must be executed. Similarly resource

owners use web based interface to register and de-register the resources. Whole information about the resources is also maintained in the database. Scheduling algorithm component provides the implementation of proposed scheduling algorithms. Administrator can change the scheduling algorithms to be used for schedule generation via web interface. The other component is has the implementation of proposed ACO based resource discovery algorithm. All algorithms work on the information present in the database about various jobs and resources and update the information as soon as the job's processing is finished.

5.4.1 Web based User Interface

Web based user interface provides the way for the users to login into the system (Figure 5.6) and submit their jobs for the execution.

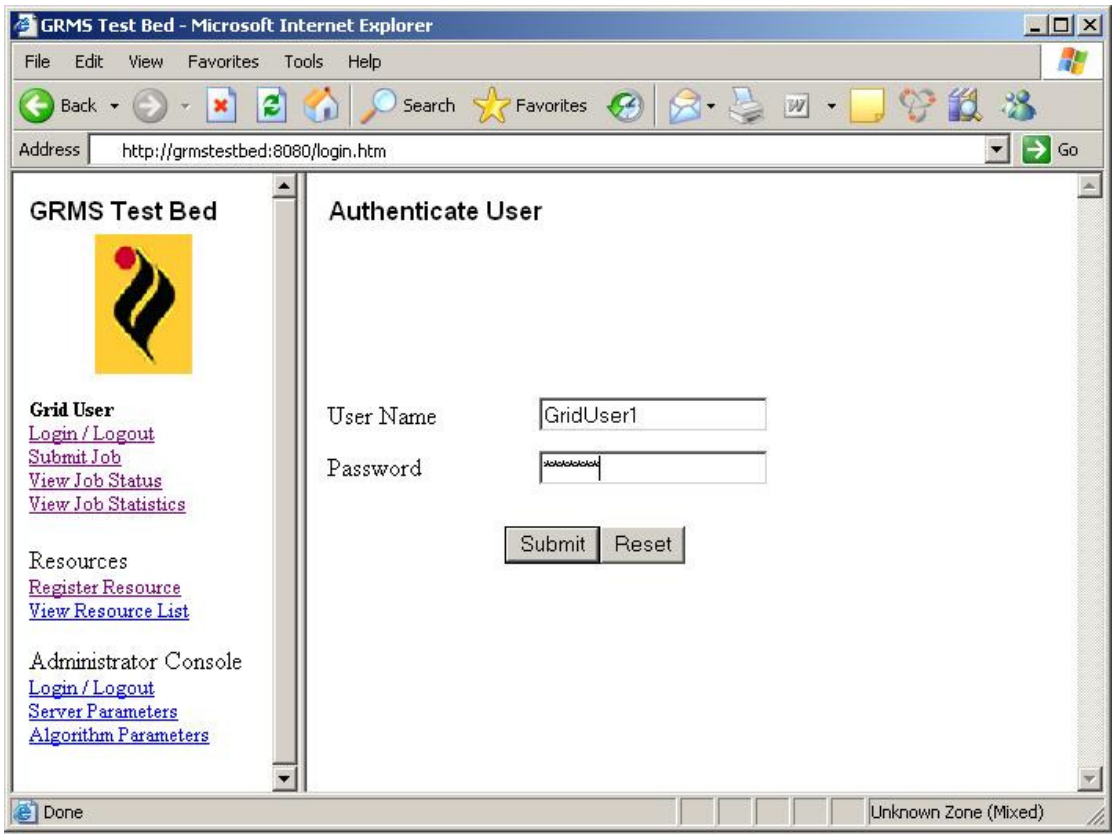


Figure 5.6: Grid User Login Screen

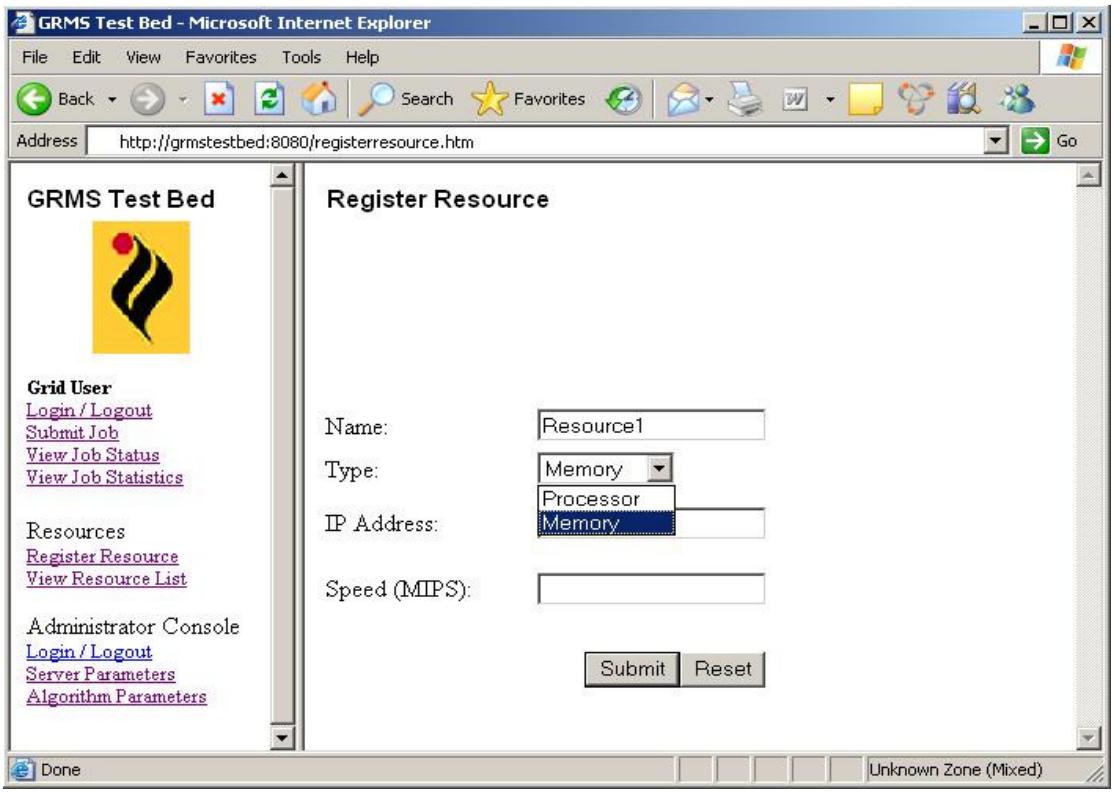


Figure 5.7: Register Resource Screen

.

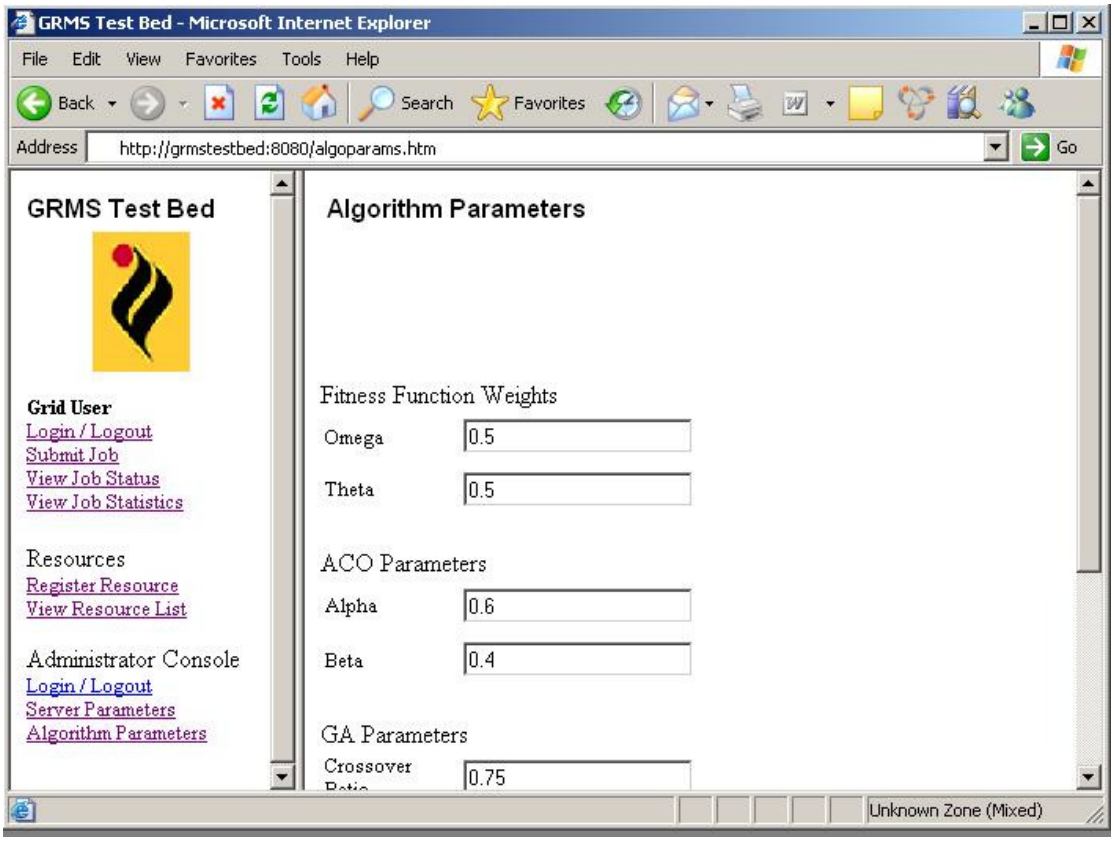


Figure 5.8: Algorithm Parameters configuration screen

The information about the jobs and resources is stored in a database. The Figure 5.7 shows the screen to register resource. Resource discovery algorithms and scheduling algorithms interact with this database to get the information about the jobs and the matching resources so as to generate the schedule. After the schedule generation the resources are contacted via their IP address and the jobs are transferred to the resources for execution. Web interface also provides the way for administrators to configure the system, like log files location, size, server port etc. Administrators can also change the algorithm parameters for proposed algorithms as shown in the Figure 5.8.

5.4.2 Information flow in Grid Test Bed

All components work independently of each other and information sharing is achieved via common database. Database maintains the information about resources, job descriptions and also the different states of jobs and resources. When ever a job execution is completed the information is updated in the database. Grid users can view the status of a job at any time by entering the job IDs. Similarly the statistics for the completed jobs can be viewed by the grid user via web interface by clicking on "View Job Statistics" in the menu bar.

Sequence diagram in the Figure 5.9 shows the high level flow of information and control among various components. A screen shot of the development environment is shown in the Figure 5.10.

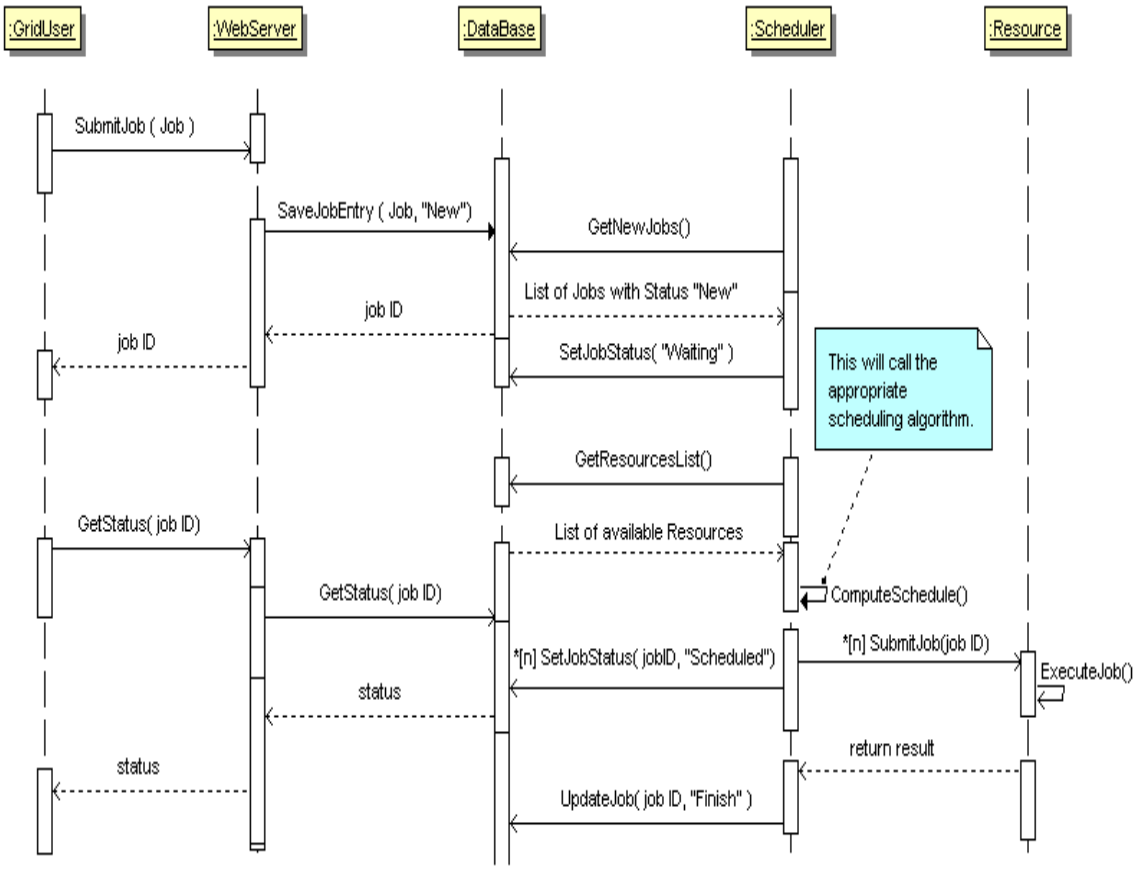


Figure 5.9: Sequence diagram showing information flow in Grid Test Bed

5.4.3 Simulation Model used in Grid Test Bed

This test bed uses the simulation model as used in [21]. Since the results for the most of existing scheduling heuristics are already available, hence it is possible to compare the performance of proposed algorithms with the existing scheduling heuristics. Details of the simulation model are provided in chapter 6 (section 6.2). A completion time matrix (*CT*) is computed based for the job completion times on different resources. The prediction system in the grid test bed uses this completion time matrix to predict the execution time of a job on a resource. *CT* is an $N \times M$ matrix, which contains an entry for every job- resource pair. This entry specifies the completion time for a job on that particular resource.

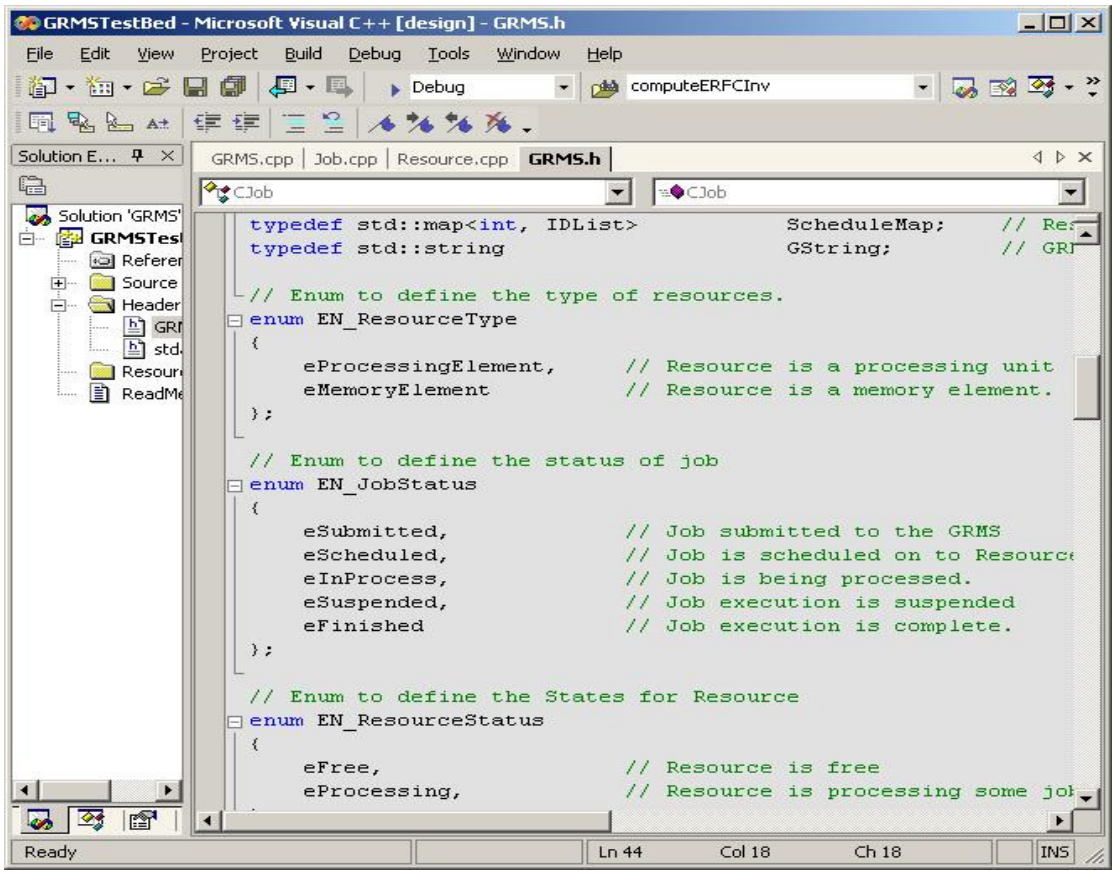


Figure 5.10: Screen shot of development environment in Grid Test Bed

In the actual grid systems state estimation is generally done based on the task profiling and analytical benchmarking. In this experimental test bed, a pre - computed completion time matrix is used, instead of actual task profiling. Completion time matrix can be consistent, partially consistent or inconsistent that corresponds to the homogeneous, semi-heterogeneous, heterogeneous environments respectively. Since grid is highly dynamic and heterogeneous environment, hence only partially consistent and inconsistent matrices are considered for the experiments. Furthermore the heterogeneity can be considered as machine or task heterogeneity. CT matrix can be varied based on low machine and task heterogeneity and high task or machine heterogeneity. Taking into consideration that grid is highly heterogeneous environment; only high machine heterogeneity and high task heterogeneity *CT* matrix are considered in experiments.

5.4.4 Mobile Agent based Framework

To test and validate proposed resource discovery algorithm, a basic framework for mobile agents is implemented in the Grid Test bed. This framework is implemented on the basis of soap services. A soap server is build using the gSoap [128] (a C++ based toolkit for implementing soap services) that listen on a designated port. This server is implemented as a Microsoft Windows service running as a background process on every machine registered with the grid. This soap server publishes the interfaces for Mobile agents transfer from one node to another. Mobile Agent at any node connects to service listening on the designated port on the next node. As soon as the connection is established, the mobile agent calls a method on the remote machine published as a soap method, and travels to the next node. The complete state of mobile agent is serialized and transferred to next node, where it is again de-serialized and mobile agent with the same state is constructed.

5.5 Summary

In this chapter, implementation detail about the grid simulator and grid test bed are provided that implement the proposed algorithms. Grid simulator is GridSim toolkit based environment that implements a centralized scheduler. Grid test bed is grid like environment that is build on the top of simulation model used by the Braun et.al. in their study of scheduling heuristics. The main aim of these implementations is to evaluate and validate the proposed algorithms against a benchmark to demonstrate their usability. Grid Test bed also implements a simple framework for mobile agents. This framework publishes the soap methods for mobile agents travel across various nodes. Mobile agents travel from node to node by serializing and de-serializing their state at source and destination nodes respectively. Next chapter provides the details about the experiments done and their results.

6

Experimental Details and Results

6.1 Introduction

An experiment is generally used to help solve practical problems and to support or negate theoretical assumptions. To support the algorithms proposed in Chapter 3 and Chapter 4 of the thesis, several experiments were performed taking different scenarios into consideration. Chapter 5 gives the implementation details of the grid test bed and GridSim based grid simulator used for experimentation. This chapter provides the details of all experimental scenarios, input parameters and

comparison with the existing algorithms. Experimental test bed uses the simulation model as used by Braun et. al.[21].

6.2 Simulation Model

To evaluate the proposed algorithms, a simulation model similar to one used in the [21] is used. To simulate the heterogeneous environment, completion time for every job on every resource is contained in a completion time matrix, *CT*, which is an $N \times M$ matrix. Since, in a heterogeneous environment any job executed on a machine/ resource can have different completion time, hence, this matrix contains the completion time of every job on every machine / resource. In the actual grid systems state estimation is generally done based on the task profiling and analytical benchmarking. In this experimental test bed, a pre - computed completion time matrix is used, instead of actual task profiling.

A row in the *CT* matrix contains the completion time for a given job on each resource, where as a column consists of the completion time of every job on a given resource. Hence, for a job J_i and a resource R_j, an entry of the *CT* matrix contains the completion time of job J_i on resource R_j.

6.2.1 Constructing CT matrix

Constructing the *CT* matrix is a three step process as given below in the pseudo code. In the *Step 1*, all necessary initialization are performed. Completion time matrix, *CT*, is a two dimensional array of size $N \times M$ where, N is the total number of jobs and M is the total number of machines / resources. To construct the *CT* matrix, first a baseline vector, *B*, of floating point number is generated. Baseline vector, *B*, will contain N elements.

Step 2 explains the creation of Baseline vector, *B*. To generate the baseline vector, *B*, a uniform random number is X_b is generated such that $X_b \in [1, U_b)$ (U_b is the upper bound of the range of possible values of *B*). X_b is generated repeatedly for *N* times and baseline vector, *B,* is constructed such that $B[i] = X_b^i$ where $1 \le i \le N$.

In the *Step 3*, rows of the *CT* matrix are generated from the baseline vector, *B*. A uniform random number X_r, known as row multiplier, is generated such that $X_r \in [1, U_r)$ (U_r is the upper bound of the range of possible values of the row multiplier, X_r). M different row multipliers are required for a row. $X_r^{i,j}$ is generated for every element of the *CT* matrix in a row and final value of *CT* matrix element is generated as $CT[i,j] = X_r^{i,j} * B[i]$, where $1 \le i \le N$ and $1 \le j \le M$.

6.2.2 Pseudo Code

```
// Step 1. Do Initializations
Initialize CT[N][M]
Initialize B[N]
Initialize Ub
Initialize Ur

// Where CT[N][M] is the Completion time matrix
// B[N] is the base line vector
// N = Total number of jobs
// M = Total Number of Machines / Resources
//  Ub = upper bound of range of possible values of
baseline vector B.
// Ur = upper bound of Range of possible values for Row
multiplier

// Step 2. Generate a base line vector
For i = 1 to N
{
     // Get a Random Number Xi in the range (1 - Ub)
     Xb = getRandom(1, Ub);
     B[i] = Xb
}

// Step 3. Generate Rows and Fill the CT[N][M]
For i = 1 to N
{
     For j = 1 to M
```

```
        {
                // Generate Row Multiplier Xr,
                // where Xr is a uniform Random number in the
                // range (1-Ur)
                Xr = getRandom(1, Ur);
                CT[i][j] = B[i] * Xr
        }
}
```

6.2.3 Achieving Heterogeneity

The characteristics of the *CT* matrix are varied to achieve the heterogeneity. The variation among the execution times of jobs for a given machine is defined as *task heterogeneity*. Task heterogeneity is achieved by changing the upper bound of the random numbers with in the base line column vector, *B*. High Task heterogeneity is achieved taking the U_b = 3000 and low task heterogeneity is achieved by taking U_b = 100. Similarly high machine heterogeneity and low machine heterogeneity is achieved by varying the upper bound, U_r, of the random number used to multiply the base line column vector. High machine heterogeneity is represented by taking U_r = 1000 and low machine heterogeneity is achieved by taking U_r = 10.

Further the *CT* matrix can be categorized, based on the consistencies, as *consistent, partially consistent and inconsistent*. Consistent means, if a resource R_j executes the job J_i faster than resource R_k then, it will execute all the jobs faster than R_k. On the other hand, inconsistent means a resource R_j can be faster than resource R_k for some tasks and slower for others. Partially consistent matrix is inconsistent matrix that contains the consistent sub matrix of predefined size. Since grid is a truly heterogeneous environment, hence, only inconsistent and partially consistent matrices are considered here for experimentation. Sample 4×4 excerpt for inconsistent and partially consistent matrices are shown in the Table 6.1 and Table 6.2 respectively. These are taken from the actual matrix of 512×16, used in the experiments.

933803.1	2156144.2	2202018.1	2286210
479091.9	150324.5	386338.1	401682.9
1400308.1	2378363	2458087.2	351387.4
576144.9	1475908.2	424448.8	576238.7

Table 6.1: 4×4 excerpt from inconsistent, high task heterogeneity and high machine heterogeneity matrix

101202.5	16453.7	64152.5	29172.8
195067.8	79175.8	787263.3	173239.2
434445.7	135989.1	496326.8	221097.9
138449	32730.9	93025.9	90044.4

Table 6.2: 4×4 excerpt from partially consistent, high task heterogeneity and high machine heterogeneity matrix

6.2.4 Algorithm parameters used in experiments

This section describes the parameters and their values for different algorithms that are common for all the experiments. Other values that are specific to the experiments are described along with the experimental details. Table 6.3 below provides the parameters names and their values taken in the experiments.

Algorithm Name	Parameter Name	Value	Description
SGASchedule	X_c	0.75	Cross Over Probability
	X_m	0.01	Mutation Probability
	ThresholdSD	1.00E-06	Threshold Standard Deviation
ACOSchedule	α	0.6	Priority of *Pheromone trail* to decide the probability of next move
	β	0.4	Priority of *Heuristic information* to decide the probability of next move
HybridGSA	T	1	Initial Temperature
	γ	0.9	Cooling Rate
	X_c	0.75	Cross Over Probability
	X_m	0.01	Mutation Probability
	ThresholdSD	1.00E-06	Threshold Standard Deviation

Table 6.3: Parameters and their values used in experiments.

Table 6.3 shows the parameters and their values with respect to different algorithms as taken in the experiments. X_c and X_m are the crossover and mutation

probabilities used in GA based algorithms SGASchedule and HybridGSA. Similarly *ThresholdSD* is the threshold standard deviation. If the standard deviation of the population comes below this *ThresholdSD* the solution is assumed to be converged. T and Υ are the initial temperature and cooling rate for simulated annealing, used in HybridGSA, which are used to decide the selection probability of newly formed offspring.

6.3 Experimental Results for proposed scheduling algorithms using Grid Test Bed

This section provides the experimental details and results for the proposed scheduling algorithms SGASchedule, ACOSchedule and HybridGSA. As mentioned in the simulation model, Completion Time matrix can vary as per the heterogeneity. Since grid is highly dynamic and heterogeneous environment, so we have not considered consistent CT matrix for our experimentations. Inconsistent and partially consistent matrices are considered only. Also for our experimental purposes we are considering high task heterogeneity only, where as both high and low machine heterogeneity is considered. Since high task heterogeneity is considered for all experiments, hence $U_b = 3000$ is taken for the construction of all *CT* matrices. To compare the results with already existing results, we have considered the 512×16 matrices as given in [21], i.e. $N = 512$ (number of jobs) and $M = 16$ (number of machines). Table 6.4 shows results obtained from different experiments for minimizing the makespan and graphs in Figure 6.1 – 6.4 below shows the comparative analysis among different proposed scheduling algorithms and also with the existing GA and SA based algorithms discussed in [21].

6.3.1 Minimizing makespan

The existing results were available for minimizing the makespan only, so minimizing makespan is taken as the objective function for the experiments and comparison purposes. This is achieved by setting the $\theta = 0$ in the Equation 3.1. All these values are averaged over *10* simulations runs for each type of matrix and each type of algorithm.

	SGASchedule	HybridGSA	ACOSchedule
Inconsistent, High Machine Heterogeniety	341512	467784	389161
Inconsistent, Low Machine Heterogeniety	71019	90127	82194
Partially Consistent, High Machine Heterogeniety	391156	521016	434032
Partially Consistent, Low Machine Heterogeniety	82114	112147	100213

Table 6.4: Makespan results obtained from experimental results for proposed algorithms

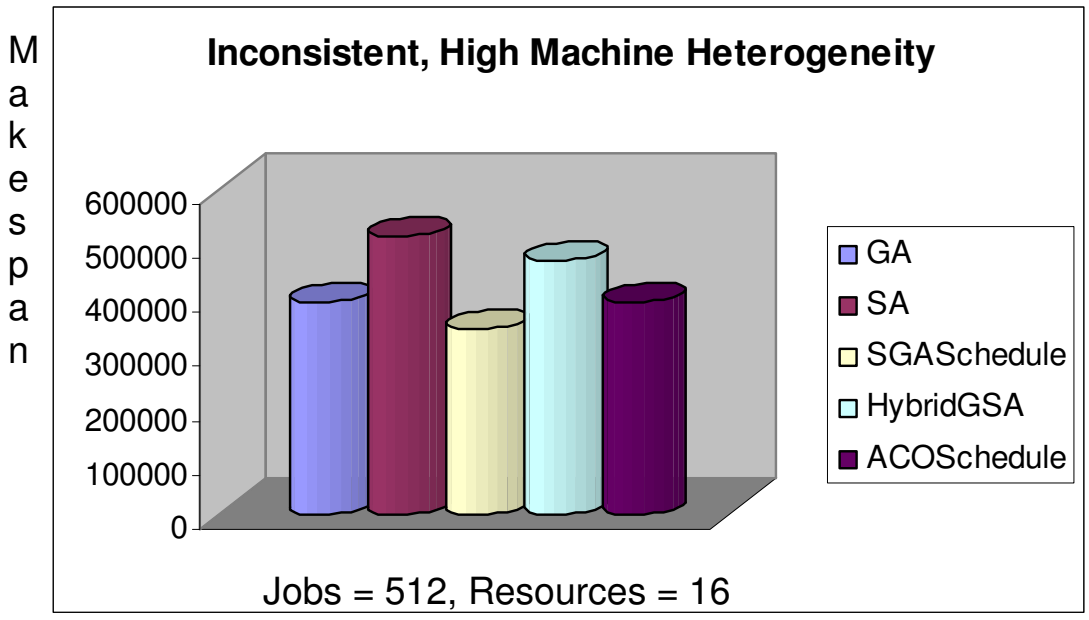


Figure 6.1: Comparison results for Inconsistent, High Machine heterogeneity CT matrix

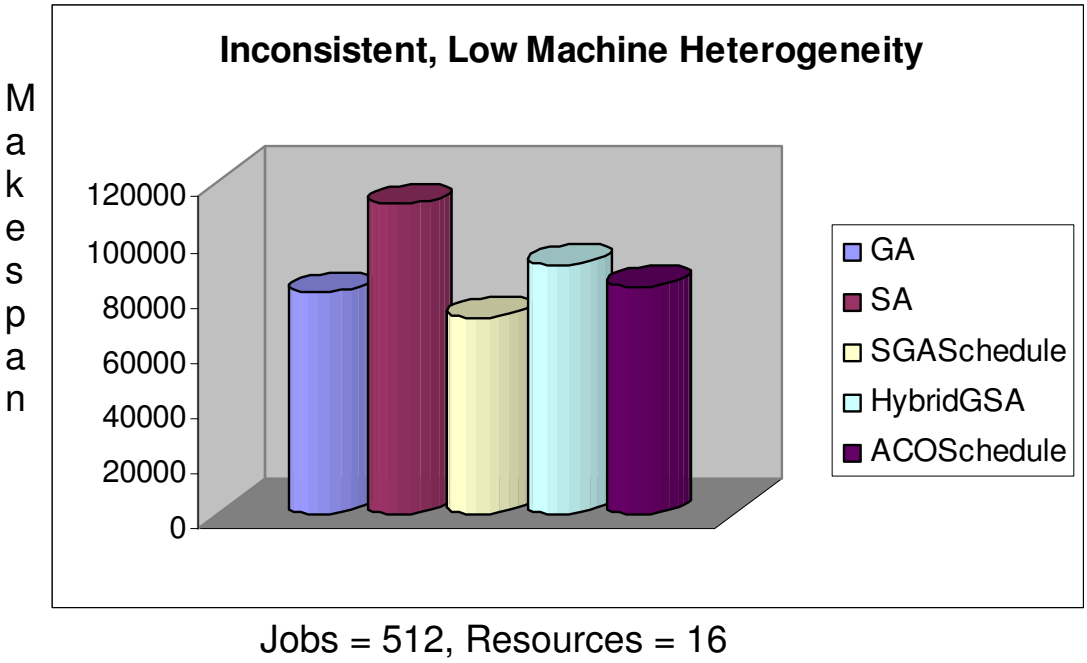


Figure 6.2: Comparison results for Inconsistent, Low Machine heterogeneity CT matrix

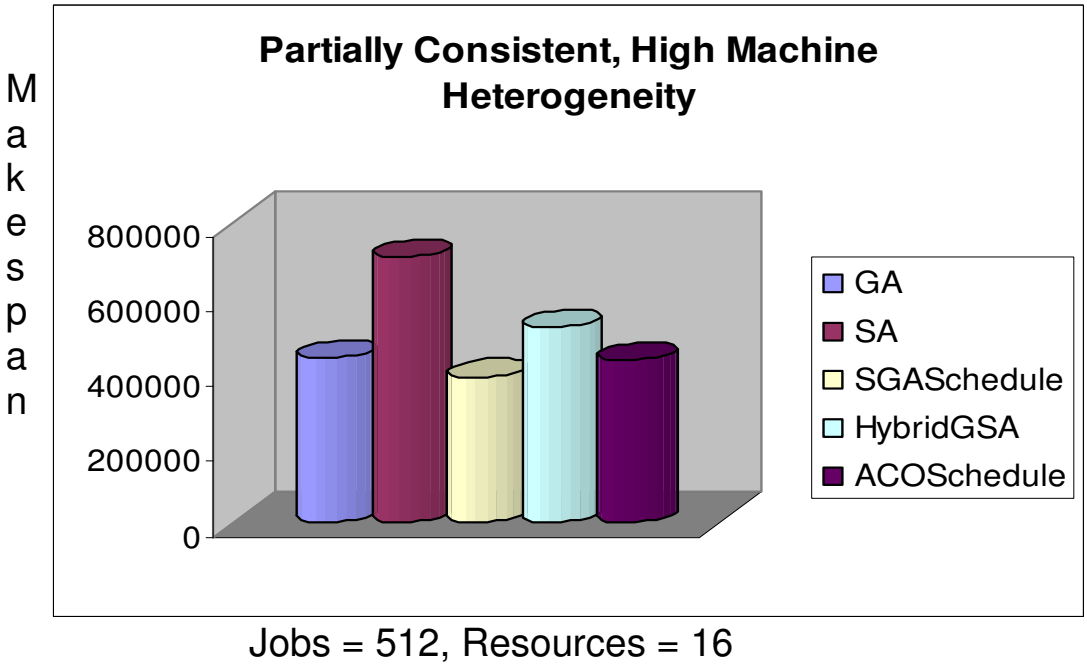


Figure 6.3: Comparison results for Partially Consistent, High Machine heterogeneity CT matrix

Graphs in Figure 6.1-6.4 show the comparison of proposed algorithms with the existing ones. GA and SA represent the results of existing genetic algorithms and simulated annealing based scheduling algorithms as given in [21], where as the

SGASchedule, ACOSchedule and HybridGSA represent the proposed grid scheduling algorithms.

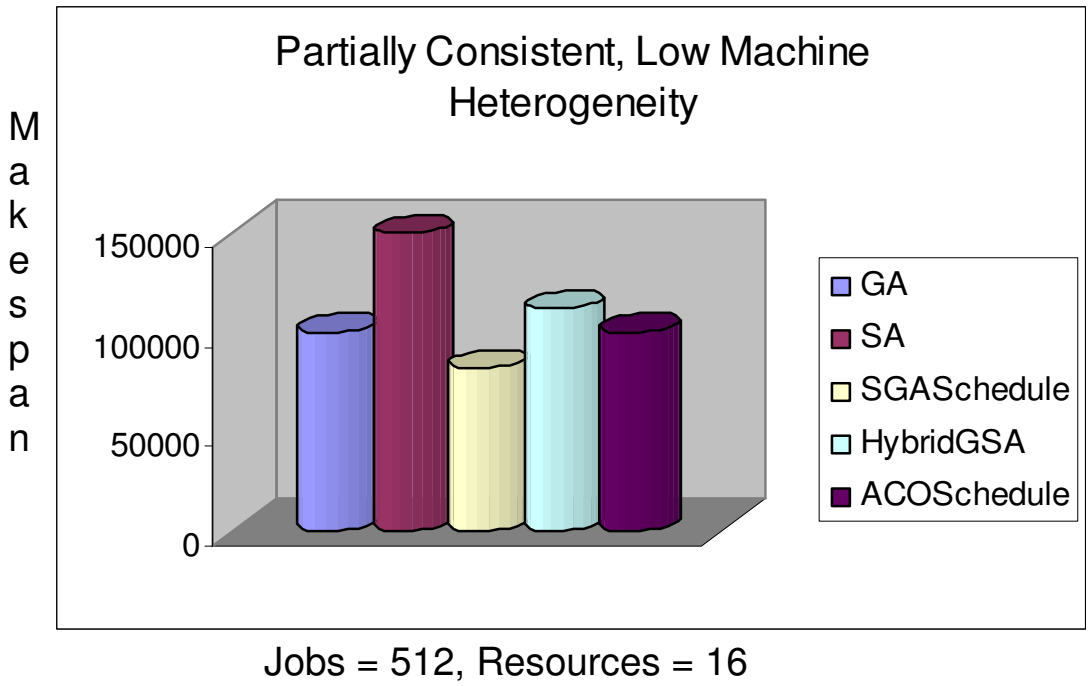


Figure 6.4: Comparison results for Partially Consistent, Low Machine heterogeneity CT matrix

6.3.2 Minimizing Cumulative Time delay (T_{delay})

Next experiments were carried on to compare proposed algorithms for cumulative time delay (T_{delay}), instead of makespan. This is achieved by setting $\omega = 0$ in the Equation 3.1.

	SGASchedule	HybridGSA	ACOSchedule
Inconsistent High Machine Heterogeneity	17213	23121	19213
Partially consistent High Machine Heterogeneity	19347	24921	20519

Table 6.5: Comparison results for minimizing T_{delay}

Since there were no results for minimizing the cumulative time delay in for existing algorithms. Hence the comparison is done only for the proposed algorithms. This is achieved with the experimental test bed by specifying deadline, while submitting the jobs. Experiments were done inconsistent *CT* matrix with high machine heterogeneity and partially consistent *CT* matrix with high machine heterogeneity. Also for both of these experiments task heterogeneity is considered as high, as in all other experiments. Results are given in Table 6.5. Comparison among proposed algorithms is depicted by the graphs in Figure 6.5 and Figure 6.6.

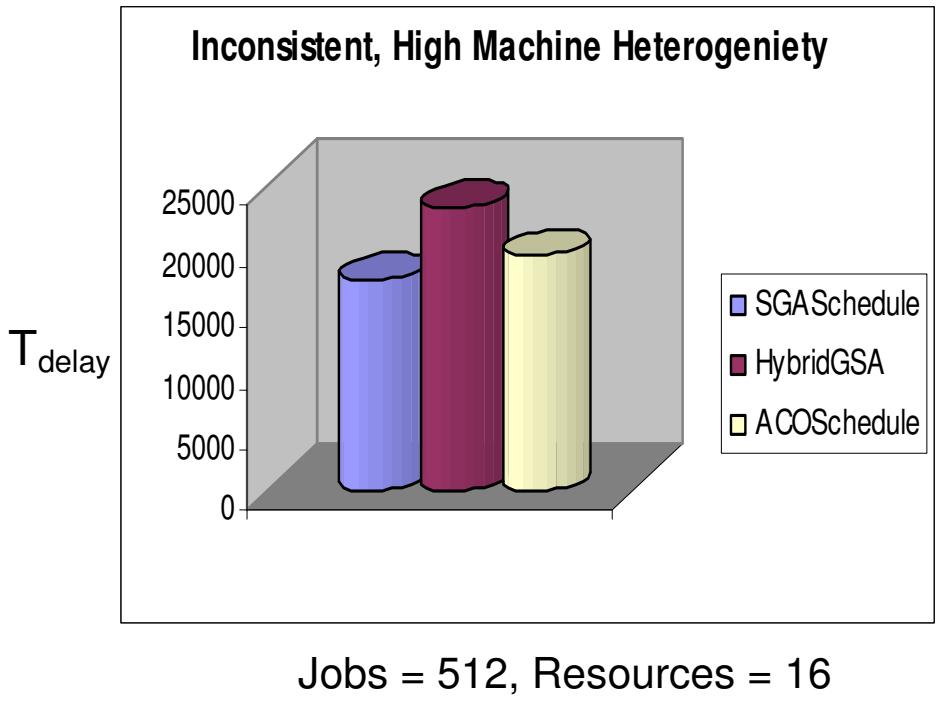


Figure 6.5: Comparison of cumulative time delay (T_{delay}) for proposed algorithms for inconsistent, high machine heterogeneity.

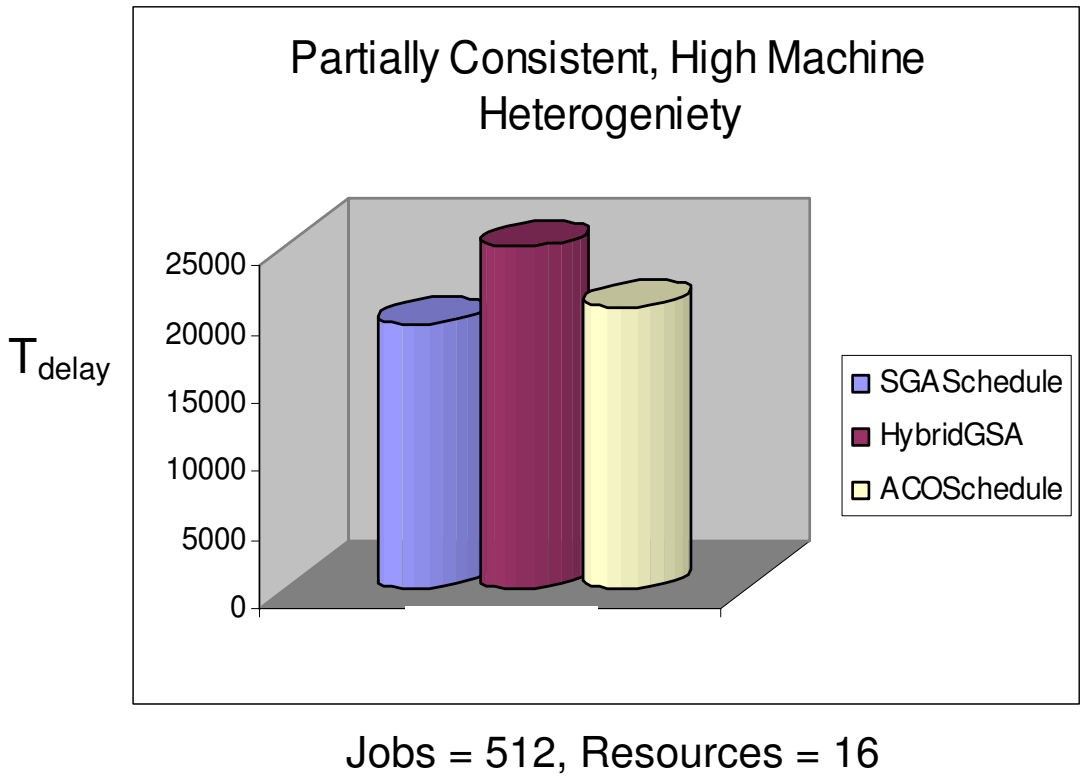


Figure 6.6: Comparison of cumulative time delay (T_{delay}) for proposed algorithms for partially consistent high machine heterogeniety.

It is observed from the above results that SGASchedule (SexualGA based scheduling algorithm) outperform all other proposed as well as existing algorithms. SGASchedule uses SexualGA based selection mechanism, where both parents are selected using different selection schemes. It uses roulette wheel selection to select one parent and n-tournament selection as the selection mechanism for selecting second parent. Hence it truly reflects the selection in the living beings by correctly simulating the male vigor and female choice. This SexualGA based selection mechanism reduces the selection pressure for the individuals and hence provides the better exploration of search space. SGASchedule performs best for minimizing both of the objectives: minimizing the makespan and minimizing the cumulative time delay. ACOSchedule is the second best in achieving the objectives. It uses pheromone deposited by ants as the learning mechanism to schedule the jobs on to the resources

which is further improved by applying the genetic operators. Application of genetic operators (crossover and mutation) reduces the probability of ACO based algorithm to saturate without proper exploration of search space. It is clear from the results that ACOSchedule performs almost same (sometimes better) as existing GA based algorithm which was considered as best scheduling heuristic till now. HybridGSA (a hybrid approach based on Genetic Algorithms and simulated annealing) performed better than existing simulated annealing based scheduling algorithm. But still existing GA based algorithm is better than the hybrid approach for grid scheduling problem.

6.4 Experimental Results on GridSim based Grid Simulator

Grid simulator, as mentioned in the previous chapter, is a simple centralized broker that has been realized on the top of a well-known grid simulation toolkit 'GridSim' to validate the efficacy of proposed scheduling algorithms. Since grid is very dynamic and heterogeneous environment, hence it is very difficult to perform experiments with large size grids. Hence GridSim based grid simulator provides a controlled environment. Also comparison is done with the Nimrod/G implementation on GridSim.

To compare the results with Nimrod/G [103], experiments have been performed with grid simulator. Results have been obtained by scheduling independent sets of jobs generated randomly on Nimrod/G set on time sharing mode and same set is scheduled using the SGASchedule (proposed algorithm) on grid simulator. Three different experiments were performed with *30* resources / machines. Number of jobs / gridlets is taken as *500, 800* and *1000* for three experiments. (A Gridlet is a package that contains all the information related to the job and its execution management details such as job length expressed in MI (Millions Instruction), the size of input and output files, and the job owner id. Individual users model their application by creating

Gridlets for processing them on Grid resources.). In GridSim based simulations, a machine / resource can have more than one processing elements. But for the experiments, all the machines are assumed to have one processing element only. Hence the terms “machine”, “resource”, “processor” are same.

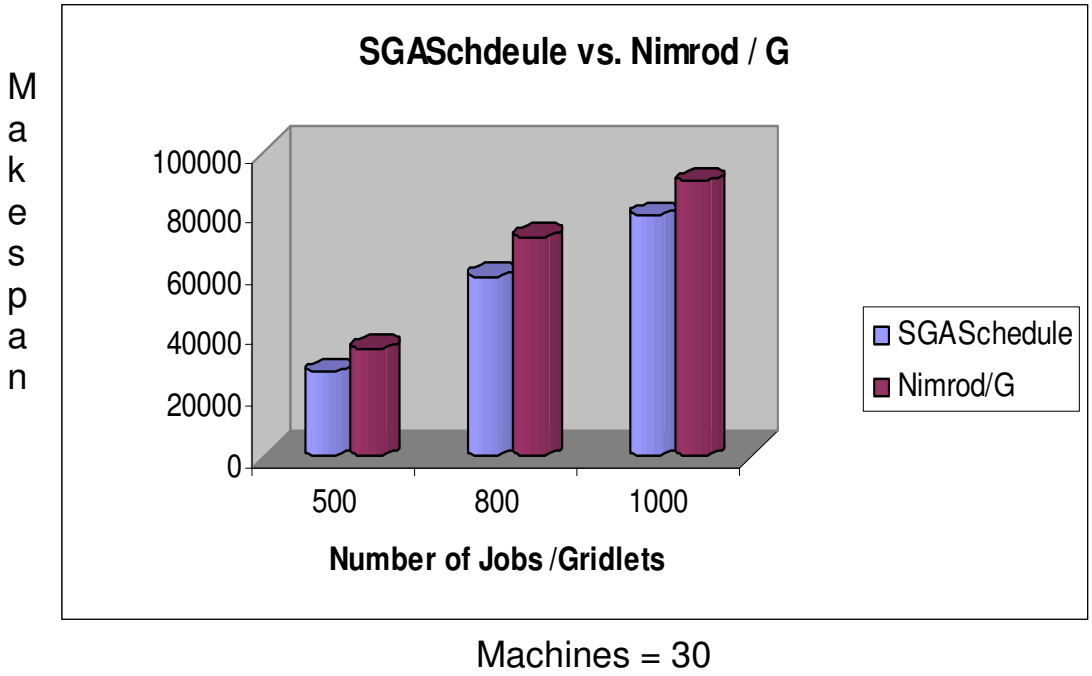


Figure 6.7: Comparison of SGASchedule Vs. Nimrod/G

It is clear from the Figure 6.7 that SGASchedule performs better than Nimrod/G implementation [103] for minimizing makespan. As the number of jobs / Gridlets increase, the SexualGA based proposed scheduling algorithm, SGASchedule, performs better than the Nimrod/G. Nimrod/G schedules the jobs using first come first served (FCFS) basis initially and switches to GA based scheduling at later stage. On the other hand the proposed algorithm uses GAs for scheduling from start to end. Also the use of two different selection schemes, as mentioned earlier, aids more exploration of search space. The combined effect of these two factors makes SGASchedule perform better as compared to Nimrod/G.

6.5 Experimental results for Proposed Resource Discovery Algorithm

Experimental details and results for proposed grid scheduling algorithms are discussed in previous section. This section discusses the experimental details and results for proposed ACO guided mobile agents based resource discovery algorithm (ACORD). Implementation details are provided in the Chapter 5. To perform the experiments and ACO based multi-agent environment is implemented by integrating proposed algorithm in the Grid Test bed.

Experiments are done by simulating the grids of different sizes varying from *200* nodes to *1000* nodes (node represents a machine that can host one or more resources for sharing). Each node is having varying degree of resources shared on grid. Resources were selected from the list of *100* resource IDs and these are assigned randomly to all the nodes. Nodes were added and removed during simulation runs to simulate the dynamic nature of the grid. User requests are generated using non-uniform (Gaussian) distribution. This distribution displays the patterns very similar to real world systems. A sample non-uniform distribution of user requests is shown in Figure 6.8.

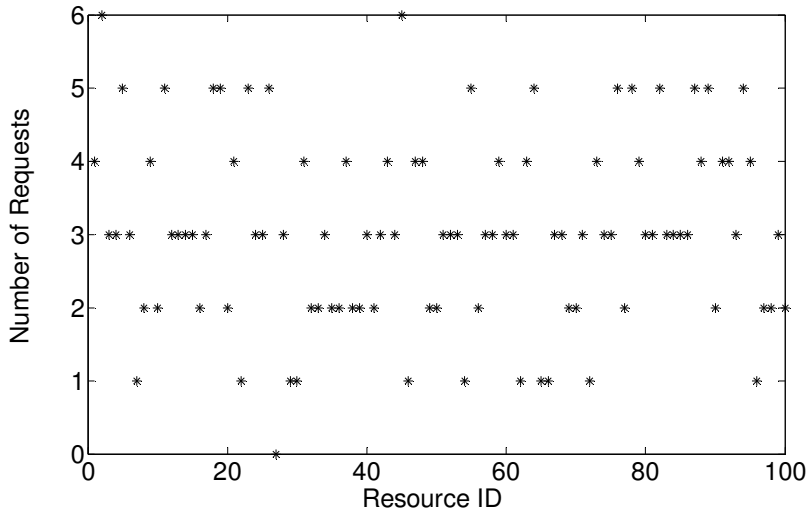


Figure 6.8: User Requests Profile (Gaussian distribution)

First experiment is carried out to find out the appropriate number of ants (BranchingFactor) to be created corresponding to the resource query. Same experiment is repeated by varying the number of ants from *1* to *50* and number of ants vs. percentage query hits is plotted. Values of α and β (Equation 4.4) are taken as *0.6* and *0.4* respectively. The plot in Figure 6.9 shows that as we increase the number of ants, query hits also increase. For lesser number of ants there is lesser pheromone deposited to guide the ants and hence less hits. Also for number of ants above *20,* there is not much increase in the number of query hits. So *15* seem to be the appropriate number of ants. For all other experiments, *15* ants are taken per query.

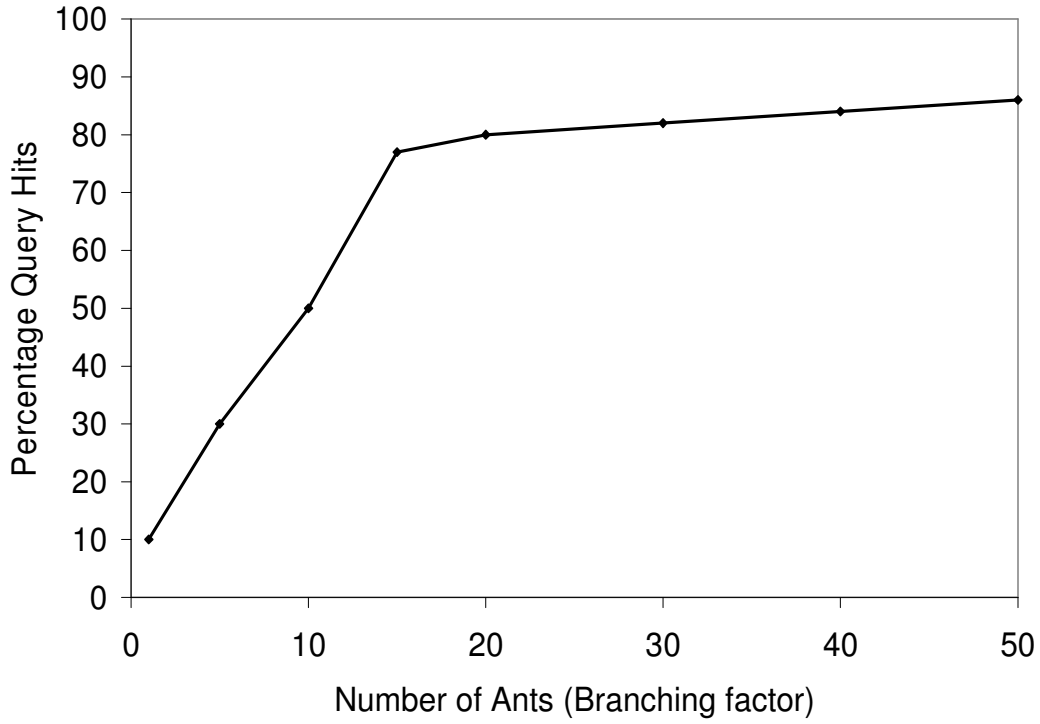


Figure 6.9: Number of Ants Vs Query Hits

Finally, the performance of ACORD is tested by varying the number of nodes and hence the network size. *MaxNodes* is taken as *10*, means an ant is allowed to visit at most *10* nodes for its search. Percentage of Query hits Vs. Network size is depicted in the plot given in Figure 6.10. This can be concluded from the Figure 6.10 that on an

average there are *80%* of query hits. There is a small decrease in the query hits as the network size grows. As the network size grows *MaxNodes* needs to be adjusted accordingly. This will increase the search periphery of the ant before it actually declares the query miss. It is established from the results shown in plot (Figure 6.10) that as we increase the *MaxNodes* from *10* to *15*, the number of query hits also increase at increased network sizes. The average query hits percentage increases to approximately *85%*.

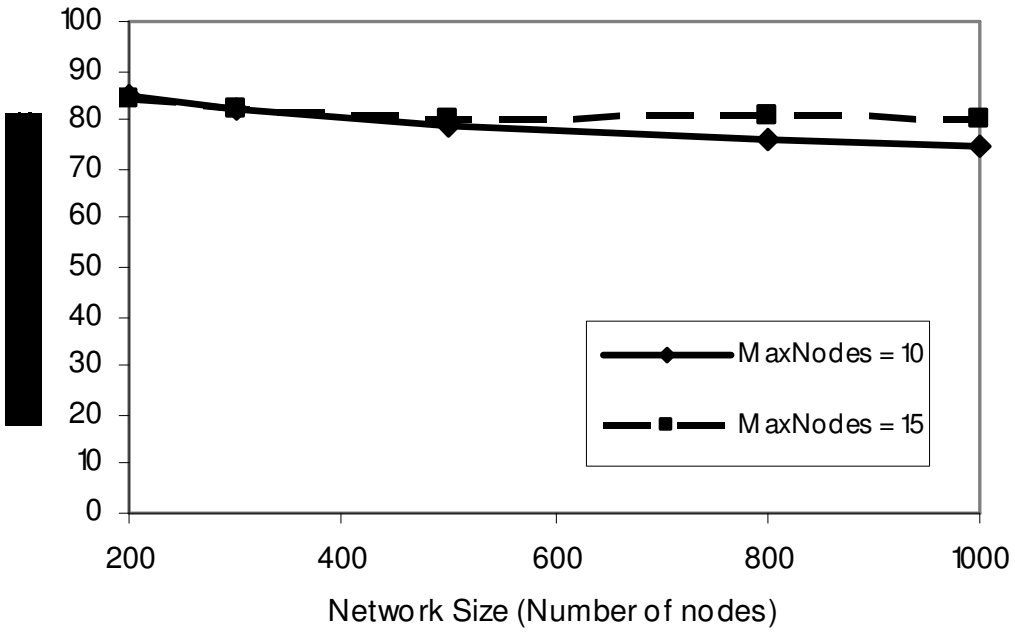


Figure 6.10: Network Size Vs. Query Hits

Experimental results show that the ACORD is scalable and can cope with the increasing network sizes with the adjustment of few parameters mentioned in the algorithm. Also the use of mobile agents shifts the actual processing to the different nodes rather than having the whole processing on a centralized machine.

6.6 Summary

This chapter provides the experimental details and results of the proposed algorithms in a simulated environment. Experiments are performed based on the inconsistent and partially consistent completion time matrix with high task and machine heterogeneity. Results show that proposed meta-heuristic based algorithms perform better than the existing algorithms for grid scheduling. Also the experimental details about ACO guided mobile agent based proposed resource discovery algorithm are given and results show that with the adjustment of few parameters this noble approach is able to cope with the growing network sizes in a grid environment.

7

Conclusions & Future Scope

7.1 Conclusions

Grid computing has emerged as an important discipline through an increased focus on resource sharing, co-ordination, manageability and high performance. Computational Grid is a distributed and heterogeneous computing system that combines open, shared, geographically distributed and heterogeneous resources to achieve high computational performance. Since a World Wide Grid can span over million of computers connected over the internet, hence resource management becomes quite challenging and complex. To find a set of resource that match the job

requirements and then actually scheduling the jobs on to the resources is a compute-intensive problem and the complexity grows as the size of grid increases. This pushes a strong need for meta-heuristic based algorithms to be devised for grid resource management systems that can explore the large search space and can quickly converge to an optimal or near optimal solution.

In this thesis three new meta-heuristics based algorithms for grid resource scheduling have been proposed, implemented and validated. These are: SexualGA based Resource Scheduling Algorithms (SGASchedule), ACO based Resource Scheduling Algorithm (ACOSchedule) and Hybrid Genetic Simulated Annealing based Grid Scheduling Algorithm (HybridGSA). It is clear from the experimental results that SGASchedule performs best among all three algorithms and also from existing algorithms. Genetic Algorithms (GA) based scheduling algorithm is the best known heuristic till now. SGASchedule further improves over GA by using SexualGA based selection mechanism, which uses two different selection mechanisms to select different parents. This truly mimics the selection in the human genetics by correctly simulating the concept of male vigor and female choice. This helps to maintain the genetic diversity of the population and enables better exploration of search space. ACOSchedule comes second best in the list. It performs a little better over GA based algorithm in some cases and is almost approaching the existing GA based algorithms in the other. ACOSchedule uses the ACO based algorithm in the first place to find the schedule. This result is further enhanced by applying genetic operators: crossover and mutation. HybridGSA uses SGASchedule as the base algorithm where the probability of acceptance or rejection of new offsprings is based on the system temperature that decreases after every iteration. Even though it is assumed the Hybrid approach based on GA and SA can harness the parallel

exploration of search space and better convergence properties of GA and SA respectively but still the experimental results show that HybridGSA performs poor as compared to proposed algorithms SGASchedule and ACOSchedule and also to existing GA based algorithm.

A new ACO guided Mobile agents based resource discovery algorithm (ACORD) has also been proposed. This algorithm uses ACO based learning mechanism for resource discovery. Mobile Agents are used to simulate the ants in the actual resource discovery process. Hence this approach reduces the network traffic by encapsulating and moving the processing to the data than to have data transferred to a central machine and processed there. ACO guided search enables the Mobile Agents to quickly reach to the fruitful nodes. It has been established from the experiments that for finding the exact match that this approach gives approximately 85% query hits, which could increase if we go for approximate match instead of exact match. Moreover, it has been observed that with the adjustment of few parameters in this novel approach it is possible to extend the scalability of the grid size effectively.

7.2 Future Scope of work

This thesis demonstrated the usability of meta-heuristic techniques for Grid resource management. This work has laid down the foundation for using meta-heuristic techniques for grid environment but also opens up new avenues for future work in this direction.

7.2.1 Supporting Parallelization of Grid Scheduling Algorithms

While the meta-heuristics based algorithms perform well for grid scheduling, a further enhancement that can be done to the proposed algorithms is the evaluation of parallel version of these algorithms. Since all scheduling algorithms are population

based algorithms, hence the actual process of match-making and scheduling itself can be parallelized and distributed to more than one processor. Research can also be focused on designing the scheduler / broker itself as a grid application and distribute the task of schedule generation to the systems registered with the grid with in a virtual organization. ACO Guided Mobile agent resource discovery algorithm can also be further enhanced by integrating with Dynamic Hash Tables to speed up the resource discovery process.

7.2.2 Data intensive Programming and Scheduling Framework

Further research can also be focused on developing the programming and scheduling frameworks for data intensive applications from commerce, science and medicine that require need large processing power as well as data analysis / processing on large distributed data bases. Mobile Agents is another area of interest which can be further explored. Mobile agent based grid application framework can be developed for application that have little computation part but huge data that needs to be analyzed / processed. Mobile agents can travel to the data store with computation code encapsulated with them and can move the processing part rather than moving the huge data to a processing node.

7.2.3 Scheduling Framework for Real Time applications

Another area that needs attention is to develop grid based framework for real time applications where a quick response to an event is required rather than only optimally utilizing the grid resources. An example is a social community web site where one user needs to search for some specific person in the whole user database, where as the other user may query for other set of people who shared the same school with him/her. These types of requests generally require a soft real time response.

7.2.4 Grid based Generic Mathematical / Software libraries

Mathematical libraries for matrix operation are a good choice to be developed on a grid based application frameworks. Similarly, Linear Programming (LP) or Mixed Integer Linear Programming (MILP) solvers can be a good choice to be developed as grid enabled software libraries. Future research can be focused on designing grid based architectures for these libraries.

7.2.5 Grid based DWDM Network Design and Planning Tool

Designing Dense Wave length Division Multiplexing (DWDM) based network design and planning has many problems like wavelength assignment, Amplifier placement, link budget design etc. that are computationally hard and require lot of optimization. A good DWDM design tool tries to optimize the hardware placement so as to reduce the overall cost of network equipment, thus helps sales personnel to quickly place cost effective bids to the customers. Designing large and mesh DWDM networks generally take 2-3 days or more on single desktop computer with some sub-optimal designs that may result in lose of business. A grid enabled distributed architecture for DWDM design tool may solve all these problems.

References

[1] I. Foster and C. Kesselman (editors), "*The Grid: Blueprint for a Future Computing Infrastructure"*, Morgan Kaufmann Publishers, USA, 1999.

[2] R. Buyya, S. Venugopal, *"A Gentle Introduction to Grid Computing and Technologies"*, CSI Communications, India, 29(1), 2005, pp 9-19.

[3] J. Joseph, C. Fellenstein, "*Grid Computing*", Pearson Education, 2004.

[4] I. Foster, C. Kesselman, S. Tuecke, "*The Anatomy of Grid: Enabling Scalable Virtual Organizations"*, International Journal of Supercomputer Applications, 2001.

[5] "*Fundamentals of Grid Computing*", http://www.redbooks.ibm.com/ abstracts/ redp3613.html.

[6] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. "*The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets",* Journal of Network and Computer Applications, 23, 2001. pp. 187-200 (based on conference publication from Proceedings of NetStore Conference 1999).

[7] M. Dias, R. Buyya, S. Venugopal. "*InterGrid: A case for internetworking islands of Grids*", Concurrency and Computations - Practice and Experience, 20(8), ISSN: 1532-0626 Wiley Press, New York, USA, 2008, pp 997-1024.

[8] Ekmecic, I. Tartalja, V. Milutinovic, "*A survey of heterogeneous computing:Concepts and Systems"*, Proceedings of the IEEE, 84(8), Aug 1996, pp. 1127-1144.

[9] M.J.Quinn, "*Parallel computing theory and practice*" McGraw Hill International edition. 1994.

[10] J. M. Crichlow, "*An Introduction to Distributed and Parallel Computing*", Second edition, PHI, 2003.

[11] I. Foster, C. Kesselman. "*Computational Grids, Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure"*, Morgan-Kaufman, 1999.

[12] T. Kindberg, G. Coulouris, J. Dollimore. "*Distributed Systems: Concepts and Design*", Addison-Wesley, 2001.

[13] J. Michael, B. Lewis Nael, A. Ghazaleh, V. Iyengar, S. Tilak. "*Non-Uniform Information Dissemination for Dynamic Grid Resource Discovery*", 3rd IEEE International Symposium on Network Computing and Applications, ISBN 0-7695-2242-4, 2004.

[14] T. G. Ramos, A. Cristina, M. A. de Melo. *"An Extensible Resource Discovery Mechanism for Grid Computing Environments",* Proceedings of 6th IEEE International Symposium on Cluster Computing and the Grid ISBN: 0-7695-2585-7, 2006.

[15] R. Belmans, K. Vanthournout, G. Deconinck. "*A Taxonomy for Resource Discovery"*, Journal of Personal and Ubiquitous Computing, ISSN:1617-4909, 9(2), 2005.

[16] H. Ham, I. Jeong, T. Kang. *"Serving Reliable Resource Discovery in Grid environment"*, ETRI, APNOM, 2002.

[17] Y. Liu. "*Grid Scheduling*", http://www.cs.uiowa.edu/~yanliu/QE/ QEreview.pdf.

[18] O. Beaumont, A. Legrand, Y. Robert, "*Scheduling divisible workloads on heterogeneous platforms*", Parallel Computing, 29(9), September 2003, pp 1121-1152.

[19] T. Robertazzi, "*Ten Reasons to Use Divisible Load Theory*", Computer, 36(5), May 2003.

[20] V. Bharadwaj, D. Ghose, V. Mani, T. Robertazzi, "*Scheduling Divisible Loads in Parallel and Distributed Systems*", IEEE Computer Society Press, Los Alamitos CA, Sept. 1996.

[21] T. D. Braun, H. J. Siegel, N. Beck, "*A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous*

Distributed Computing Systems", Journal of Parallel and Distributed Computing, 2001.

[22] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, D. Zagorodnov, "*Adaptive Computing on the Grid Using AppLeS*", IEEE Transactions on Parallel and Distributed Systems, 14(4), April 2003.

[23] Elisa Heymann, Miquel A. Senar, Emilio Luque, Miron Livny, "*Adaptive Scheduling for Master-Worker Applications on the Computational Grid*", Proceedings of the First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Bangalore, India, December 17, 2000.

[24] L. Yang, J. M. Schopf, I. Foster, "*Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments*", SuperComputing 2003, November 2003.

[25] R. Buyya, et al, "*An Economy Driven Resource Management Architecture for Global Computational Power Grids*", Proc. Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), CSREA Press, USA, 2000.

[26] R. Buyya, "*Economic-based Distributed Resource Management and Scheduling for Grid Computing*", Ph.D. Thesis, Monash University, Melbourne, 2002.

[27] R. Buyya, D. Abramson, S. Venugopal. "*The Grid Economy*", Special Issue on Grid Computing, Proceedings of the IEEE, Manish Parashar ,Craig Lee, editors, 93(3), IEEE Press, New York, USA, March 2005, pp 698-714.

[28] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. *"Economic Models for Management of Resources in Grid Computing"*, Technical Track on Commercial Applications for High-Performance Computing, SPIE International Symposium on The Convergence of Information Technologies and Communications (ITCom 2001), August 20-24, Denver, Colorado, USA, 2001.

[29] "*Meta Heuristics*", http://en.wikipedia.org/wiki/Metaheuristic.

[30] X. Yao, "*A New Simulated Annealing Algorithm*", International Journal of Computer Mathematics, 56: 1995, pp 161-168.

[31] D. Goldberg, "*Genetic Algorithms in Search, Optimization and machine learning*", Addison-Wesley Publishing Company, Inc., 1989.

[32] S. Bawa, G.K. Sharma. "*Search space pruning for faster test generation based on parallel and adaptive GA*". Progress in VLSI Design & Test. Elite publishing, 2005, pp 444-448.

[33] S. Bawa, "*Parallel and Distributed Algorithms for VLSI Test Generation*" Ph.D Thesis. Thapar Institute of Engineering and Technology, Patiala (India). 2004.

[34] M. Dorigo, L.M. Gambardella, "*Ant Colonies for the traveling salesman problem*", BioSystems, 43, 1997, pp 73-81.

[35] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. "*A Security Architecture for Computational Grids*", Proc. 5th ACM Conference on Computer and Communications Security Conference, 1998, pp 83-92.

[36] K. Maghraoui, T. J. Desell, B.K. Szymanski, C. A. Varela. *"The Internet Operating System: Middleware for Adaptive Distributed Computing"*, International Journal of High Performance Computing Applications (IJHPCA), Special Issue on Scheduling Techniques for Large-Scale Distributed Platforms, 20(4), 2006, pp 467-480.

[37] C. A. Varela, P. Ciancarini, K. Taura. *"Worldwide computing: Adaptive middleware and programming technology for dynamic Grid environments"*, Scientific Programming Journal, 13(4), December 2005, pp 255-263.

[38] T. Casavant, J. Kuhl. "*A taxonomy of scheduling in general-purpose distributed computing systems*", IEEE Transactions on Software Engineering. 14(2), 1988, pp 141–154.

[39] H. Rotithor. "*Taxonomy of dynamic task scheduling schemes in distributed computing systems*", IEE Proceedings on Computer and Digital Techniques. 141(1), 1994, pp 1–10.

[40] K. Krauter, R. Buyya, M. Maheswaran."*A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing*", Software: Practice and Experience (SPE), ISSN 0038-0644 32(2) Wiley Press, New York, USA, 2002. pp. 135-164.

[41] I. Foster. *"Globus Toolkit Version 4: Software for Service-Oriented System"*, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2006, pp 2-13.

[42] I. Foster, C. Kesselman. **"***Globus: A Metacomputing Infrastructure Toolkit*", International Journal of Supercomputer Applications, 11(2), 1997, pp 115-128.

[43] I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke. "*Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment*", Proc. 5th IEEE Symposium on High Performance Distributed Computing, 1997, pp 562-571.

[44] W. Allcock, J. Bresnahan, R. Kettimuthu, J. Link. **"***The Globus eXtensible Input/Output System (XIO): A protocol independent IO system for the Grid*", Proceedings of the Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models held in conjunction with International Parallel and Distributed Processing Symposium (IPDPS 2005), April 2005.

[45] K. Czajkowski, I. Foster, C. Kesselman. "*Resource Co-Allocation in Computational Grids*", Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8), 1999, pp 219-228.

[46] M. Feller, I. Foster, S. Martin."*GT4 GRAM: A Functionality and Performance Study***",** TERAGRID 2007 Conference, Madison, 2007.

[47] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke. **"***A Resource Management Architecture for Metacomputing Systems***",** Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998, pp 62-82.

[48] *"Software Components for Grid Systems and Applications"*, http://www-unix.globus.org /grid_software/.

[49] D. Thain, T. Tannenbaum, M. Livny, *"Distributed Computing in Practice: The Condor Experience",* Concurrency and Computation: Practice and Experience, 17(2-4), February-April, 2005, pp 323-356.

[50] D. Thain, T. Tannenbaum, M. Livny, "*Condor and the Grid*", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0.

[51] R.J.M. Boer, "*Resource Management in the Condor System*", Master Thesis, Delft University of Technology, Netherlands, 1996.

[52] T. Tannenbaum, D. Wright, K. Miller, M. Livny, "*Condor - A Distributed Job Scheduler"*, in Thomas Sterling, editor, Beowulf Cluster Computing with Linux, The MIT Press, 2002. ISBN: 0-262-69274-0.

[53] A. Roy and M. Livny, *"Condor and Preemptive Resume Scheduling"*, Jarek Nabrzyski, Jennifer M. Schopf and Jan Weglarz, editors. Grid Resource Management: State of the Art and Future Trends, Kluwer Academic Publishers. Fall 2003, pp 135-144.

[54] J. Meehean, M. Livny, *"A Service Migration Case Study: Migrating the Condor Sched*", Midwest Instruction and Computing Symposium, April 2005.

[55] R. Raman, M.Livny, M. Solomon, *"Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching",* Proceedings of the Twelfth IEEE International Symposium on High-Performance Distributed Computing, Seattle, WA, June 2003.

[56] N. Coleman, "*An Implementation of Matchmaking Analysis in Condor*", Masters' Project report*,* University of Wisconsin, Madison, May 2001.

[57] J. Frey, T. Tannenbaum, I.Foster, M. Livny, S. Tuecke, "*Condor-G: A Computation Management Agent for Multi-Institutional Grids*", Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10) San Francisco, California, 2001.

[58] R. Buyya, et al., *"Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*", Proc. 4th Int'l Conf. on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), IEEE CS Press, Los Alamitos, Calif., USA, 2000.

[59] R. Buyya, J. Giddy, D.Abramson. *"An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications"*, Workshop on Active Middleware Services (AMS 2000), (in conjuction with Ninth IEEE International Symposium on High Performance Distributed Computing), Kluwer Academic Press, Pittsburgh, USA, August 1, 2000.

[60] R. Buyya, D. Abramson, J.Giddy. *"Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid"*, HPC Asia 2000, Beijing, China, May 14-17, 2000, pp 283 – 289.

[61] D. Abramson, J. Giddy, L.Kotler. *"High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?",* International Parallel and Distributed Processing Symposium (IPDPS), Cancun, Mexico, May 2000, pp 520- 528.

[62] A.S. Grimshaw, A. Natrajan, *"Legion: Lessons Learned Building a Grid Operating System"*, Proceedings of the IEEE, 93(3), March, 2005, pp 589-603.

[63] A.S. Grimshaw, A. Natrajan, M.A. Humphrey,*"A philosophical and technical comparison of Legion and Globus"*, IBM Systems Journal, 48(2), March, 2004. pp 233-254.

[64] S.J. Chapin, D. Katramatos, J.F. Karpovich, A.S. Grimshaw *"Resource Management in Legion",* Journal of Future Generation Computing Systems, 15, 1999, pp 583-594.

[65] M. Lewis, A.S. Grimshaw. "*The Core Legion Object Model*", Proceedings of the Symposium on High Performance Distributed Computing (HPDC-5Syracuse, NY)), Aug 1996, pp 551-561.

[66] A. Natrajan, M.A. Humphrey, A.S. Grimshaw. *"Grid Resource Management in Legion",* Resource Management for Grid Computing, Jennifer Schopf , Jaroslaw ,editors, Nabrzyski, 2003.

[67] A.Luther, R. Buyya, R. Ranjan, S. Venugopal. *"Alchemi: A .NET-Based Enterprise Grid Computing System",* Proceedings of the 6th International Conference on Internet Computing (ICOMP'05), Las Vegas, USA, June 2005, pp 27-30.

[68] R.Buyya, A.Luther, S. Venugopal, R. Ranjan. "*Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids*", The University of Melbourne, 2004.

[69] C. Kruskal, A. Weiss. "*Allocating independent subtasks on parallel processors*", IEEE Transactions on Software Engineering, 1984.

[70] M. A. Arefin, M. S. Sadik, S.Coetzee, J. Bishop. "*Alchemi vs Globus: a performance comparison* ". Technical Report, Dept. of Computer Science and Software Engineering, University of Pretoria, South Africa, July 2006 (This report can also be found at following link: http://polelo.cs.up.ac.za/papers/ICECE2006_AlchemiGlobus_9July.doc).

[71] Eduardo Huedo, Rubén S. Montero and Ignacio M. Llorente "*The GridWay Framework for Adaptive Scheduling and Execution on Grids*" Scalable Computing - Practice and Experience 6 (3), ISSN 1895-1767, 2005, pp.1-8

[72] S.Venugopal, R.Buyya, L. Winton. "*A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids*". Concurreny and Computation - Practice and Experience. 18(6) ISSN: 1532-0626 Wiley Press, New York, USA, 2006, pp. 685-699.

[73] S. Bansal, P. Kumar, K. Singh. *"A Cost-Effective Scheduling Algorithm for Message Passing Multiprocessor Systems"*, Proc. 15th International Conference on Parallel and Distributed Computing System PDCS-2002), KY, USA, Sep 2002, pp 47-52.

[74] S. Bansal, P. Kumar, K. Singh. *"Duplication Based Scheduling Algorithm for Interconnection Constrained Distributed Memory Machines"*, Proc. 9th International Conference on High Performance Computing (HIPC-2002), 2552, Lecture Notes in Computer Science, Verlog, Dec 2002, pp 52-62.

[75] S. Bansal, P. Kumar, K. Singh, *"An Improved Duplication Strategy for Scheduling Precedence Constrained Graphs in Multiprocessor Systems",* IEEE Transactions on Parallel and Distributed Systems, 14(6) June 2003, pp 533-543.

[76] T. Leighton M. Harchol-Balter, D. Lewin. "*Resource Discovery in Distributed Networks",* 18th Annual Symposium on Principles of Distributed Computing, ACM-SIGACT/SIGOPS, Atlanta, May 1999, pp 229-238.

[77] K. Kise, H. Honda, S. Tangpongprasit, Takahiro. Katagiri,T. Yuba. "*A time-to-live based reservation algorithm on fully decentralized resource discovery in Grid computing",* Journal of Paraller Computing, Elsevier, 2005.

[78] N. Venkatasubramanian, Y. Huang. "*QoS-based resource discovery in intermittently available environments*", 11th International Symposium on High Performance Distributed Computing, IEEE, 2002, pp 50 -59.

[79] K. Siu, C. Law. "*An O(log n)randomized resource discovery algorithm*", 14th International Symposium on Distributed Computing Technical Report,Technical University of Madrid, 2000, pp 5-8.

[80] R. Shah, V. Bharadwaj, M. Mishra. *"Estimation Based Load Balancing Algorithm for Data-Intensive Heterogeneous Grid Environments"*, HiPC, Bangalore, 2006.

[81] H. Kasahara, S. Narita. *"Practical multiprocessing scheduling algorithms for efficient parallel processing",* IEEE Transactions on Computers, 33, 1984, pp 1023-1029.

[82] J.H. Holland. *"Adaptation in Natural and Artificial Systems"*, MIT Press, Cambridge, MA, USA, 1992.

[83] E. Hou, N. Ansari, H. Ren. *"A genetic algorithm for multiprocessor Scheduling",* IEEE Transactions on Parallel and Distributed Systems, 5, 1994, pp 113-120.

[84] A.Y. Zomaya, Y.H. The,"*Observations on using genetic algorithms for dynamic load-balancing",* IEEE Transactions on Parallel and Distributed Systems, 12, 2001, pp 899-911.

[85] Y.K. Kwok, I. Ahmad, M. Dhodhi. *"Scheduling parallel programs using genetic algorithms",* In: Solutions to Parallel and Distributed Computing Problems, chapter 9, A. Y. Zomaya, F. Ercal, S. Olariu, editors, John Wiley and Sons, New York, USA., 2001, pp 231-254.

[86] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, R.F. Freund. *"Dynamic mapping of a class of independent tasks onto heterogeneous computing systems",* Journal of Parallel and Distributed Computing, 59, 1999, pp 107-131.

[87] M.D. Theys, T.D. Braun, H.J. Siegal, A.A. Maciejewski,,Y.K. Kwok. *"Mapping Tasks onto Distributed Heterogeneous Computing Systems Using a Genetic Algorithm Approach",* chapter 6, John Wiley and Sons, New York, USA, 2001, pp 135-178.

[88] A.Y. Zomaya, R.C. Lee, S. Olariu. *"An introduction to genetic-based scheduling in parallel processor systems",* In: Solutions to Parallel and Distributed Computing Problems, A. Y. Zomaya, F. Ercal, S. Olariu, editors, John Wiley and Sons, New York, USA, 2001, pp 111-133.

[89] M. Affenzeller. *"SexualGA: Gender-Specific Selection for Genetic Algorithms",* Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics. In: International Institute of Informatics and Systemics, 4, 2005, pp 76-81.

[90] A. Colorni, M. Dorigo et V. Maniezzo. "*Distributed Optimization by Ant Colonies*", actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 1991, pp 134-142.

[91] M. Dorigo. "*Optimization, Learning and Natural Algorithms*", PhD thesis, Politecnico di Milano, Italie, 1992.

[92] V. Maniezzo, L.M. Gambardella, F. D. Luigi. *"Ant Colony optimization"*, http://www.idsia.ch/~luca/aco2004.pdf.

[93] M. Dorigo, L. M. Gambardella, *"Ant colony system: a cooperative learning approach to the traveling salesman problem",* IEEE Trans. Evolutionary Computation, 1(1), 1997, pp 53-66.

[94] G. Ritchie. *"Static multi-processor scheduling with ant colony optimisation and local search",* Master's thesis, University of Edinburgh. 2003.

[95] D. Schrage, P.G. Gonsalves. *"Sensor scheduling using ant colony optimization",* Proc. of the Sixth International Conference of Information Fusion, 1, 2003, pp 379-385.

[96] Y. Li, Z. Xul. *"An ant colony optimization heuristic for solving maximum independent set problems",* Proc. of Computational Intelligence and Multimedia Applications, 2003, pp 206-211.

[97] S. Kirkpatrick, C.D. Gelatt and P.M. Vechhi, *"Optimization by Simulated Annealing",* Science 220, 1983, pp 671-680.

[98] E. Arts, P.V. Losrhoven. *"Statistical cooling: A general Approach to combinatorial optimization prolems"*, Philips Journal of Research, 40, 1985, pp 193-226.

[99] C.Cerny. *"A Thermodynamical Approach to Travelling Salesman Problem: An efficient Simulated Annealing algorithm",* Journal of Optimization Theory and Applications, 45, 1985, pp 41-51.

[100] K. Dowsland. *"Variants of Simulated Annealing for practical problem solving",* V. Rayward-Smith, editor, Application of Modern Heuristic Methods, Henley-on-Thames: Alfred Walter Ltd. 1995.

[101] I.H. Osman and C.N. Pots, *"Simulated Annealing for Permutation Flow-Show Scheduling"*, Omega 17, 1989, pp 551-557.

[102] V.V. Rene. *"Applied Simulated Annealing"*, Berlin, Springer, 1993.

[103] A. Abraham, R. Buyya, B.Nath. "*Nature's Heuristics for Scheduling Jobs on Computational Grids*". The 8th IEEE Conference on Advanced Computing and Communications (ADCOM 2000). Cochin, India, 2000.

[104] A.I. Iamnitchi. *"Resource Discovery in large resource sharing environments",* Chicago Illinois. 2003.

[105] A.I. Iamnitchi, I. Foster. *"On Fully Decentralized Resource Discovery in Grid Environments",* IEEE International Workshop on Grid Computing, Denver, CO, 2001.

[106] Y. Gong, W. Li, Y. Sun, Z. Xu. *"A C/S and P2P Hybrid Resource Discovery Framework in Grid Environments"* International conference on Parallel Processing – '05, 2005, pp 261-268.

[107] C. Mastroianni, D. Talia, O. Verta. *"A P2P Approach for Membership Management and Resource Discovery in Grids",* ITCC-'05, 2, 2005 pp 168-174.

[108] S. Fitzgerald, I. Foster, C. Kesselman, G. Von Laszewski, W.Smith, S. Tuecke. *"A directory Service for Configuring High Performance Distributed Computation",* Proceedings of 6th IEEE Symposium on High Performance Distributed Computing, 1997, pp 365-375.

[109] S. Jayasena, C. Yee, J. Song, A. Stoelwinder, C. W. See, W. Wong. *"Data Resource Discovery in a Computational Grid",* Proceeding of the Grid and Cooperative Computing International Workshops, 2004, pp 303-310.

[110] R. Wolski, N. Spring and J. Hayes, *"The Network Weather Service: A Distributed Resource Performance Forcasting Service for Metacomputing",* Future Generation Computer Systems, 15(5), 1999, pp 746-768.

[111] Y. Gong, F. Dong, W. Li, Z.Xu. *"Vega Infrastructure for Resource Discovery in Grids",* Journal of Computer Science Technology, 18(4), 2003, pp 413-422.

[112] A.Sharma, S. Bawa. *"An Improved Resource Discovery Approach using P2P Model for Condor: A Grid Middleware",* International Journal of Transaction on Engineering, Computing & Technology, 17, 2006, pp 55-59.

[113] A. Caglayan, C. Harrison. *"Agent Sourcebook",* Wiley Computer Publishing. 1997.

[114] S. Green et al. *"Software Agents: A review: Department of Computer Science",* Trinity College Dublin, 1997.

[115] H.S. Nwana, "*Software Agents: An Overview," Knowledge Engineering Review*", Cambridge University Press, 11(3), Sept. 1996, pp 1 – 40.

[116] J. White. *"Mobile Agents",* White Paper, General Magic, 1998. www.cs.cmu.edu/~rwh/courses/mobile/Telescript/White-Telescript.ps.

[117] D. B. Lange,M. Oshima. *"Seven Good Reasons for Mobile Agents"*, Communications of the ACM, 42(3), 1999, pp 88-89.

[118] J.Baumann, F. Hohl, K. Rothermel, M.Straßer."*Mole–Concepts of a Mobile Agent System*", World Wide Web, Springer Netherlands, 1(3), September 1998, pp 123-137.

[119] A. Fuggetta, G.P. Picco, G. Vigna, "*Understanding Code Mobility*", IEEE Transactions on Software Engineering, 24(5), May 1998.

[120] R.S. Gray. "*Agent Tcl: A Flexible and Secure Mobile-Agent System*", Proceedings of the Fourth Annual Tcl/Tk Workshop (TCL 96), July 1996, pp 9-23. http://actcomm.dartmouth.edu/papers/#security.

[121] R. Agrawal, A. Ezzat,. *"Location Independent Remote Execution in NEST",* IEEE Transactions on Software Engineering, 13(8), August 1987, pp 905–912.

[122] J.L. Albin, J.A. Lorenzo, J. C. Cabaleiro, T. F. Pena, F.F. Rivera. *"Simulation of Parallel Applications in GridSim",* Proceedings of the 1st Iberian Grid Infrastructure Conference (IBERGRID), Santiago de Compostela (Spain), May 2007.

[123] *"GridSim: Java-based Modelling and Simulation of Computational Economy-based Scheduling for Grid Computing."* Poster Exhibit @ CCGrid 2001.

[124] A. Sulistio, R.Buyya. *"The GridSim: Real Tools for Simulated Parallel and Distributed Computing",* International Science Grid This Week (iSGTW), Feature on 17 October 2007.

[125] M. Murshed, R. Buyya. *"Using the GridSim Toolkit for Enabling Grid Computing Education"*, Proc. of the International Conference on Communication Networks and Distributed Systems Modeling and Simulation (CNDS 2002), San Antonio, Texas, USA. January 2002.

[126] R. Buyya, M. Murshed. *"GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing",* The Journal of Concurrency and Computation: Practice and Experience (CCPE), 14(13-15), Wiley Press, Nov.-Dec., 2002.

[127] A. Sulistio, C.S.Yeo, R. Buyya. *"Visual Modeler for Grid Modelling and Simulation (GridSim) Toolkit"* Proc. of the 3rd International Conference on Computational Science (ICCS 2003), Springer Verlag Publications (LNCS Series), Melbourne, Australia. June 2003.

[128] R.V. Engelen, "*gSOAP 2.7.11 User Guide*", Florida State University and Genivia, Inc., http://www.cs.fsu.edu/~engelen/soapdoc2.html.

Publications

[Pub1] Bhupinder Singh, Seema Bawa. *"ACO based Optimized Scheduling Algorithm for Computational Grids"*. Proceeding (559) Advances in Computer Science and Technology – 2007, ACTA Press (2007). Pp 283-286.

[Pub2] Bhupinder Singh, Seema Bawa. *"Meta-Heuristic Techniques for Grid Resource Management System"*. SIEMENS Information Systems Ltd. Technical Journal, Vol. 1 - December 2007. pp 20-25.

[Pub3] Bhupinder Singh, Seema Bawa. *"Grid Resource Discovery using Natural Mobile Agents"* International Journal of Computer Science and Network Security. Vol 8. No. 11, November 2008. pp 411-415.