

**A MODEL FOR RESOURCE ALLOCATION FOR
COMPUTATIONAL GRID USING GENETIC ALGORITHM**

*Dissertation submitted to Jawaharlal Nehru University
in partial fulfillment of the requirements
for the award of the degree of*

**MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE AND TECHNOLOGY**

**KRISHAN VEER SINGH
ENROLLMENT NO.11/10/MT/19**



**SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
INDIA
2013**

SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI, 110067 (INDIA)



CERTIFICATE

This is to certify that the dissertation entitled “A Model for Resource Allocation for Computational Grid using Genetic Algorithm” is being submitted by Mr. Krishan Veer Singh to School of Computer and Systems Sciences, Jawaharlal Nehru University New Delhi-110067, India in the partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Technology. This work has been carried out by him in the School of Computer and Systems Sciences under the supervision of Dr. Zahid Raza. The matter personified in the dissertation has not been submitted for the award of any other degree or diploma.

A handwritten signature in black ink, appearing to read 'Zahid Raza', with the date '22.7.13' written below it.

DR. ZAHID RAZA
(SUPERVISOR)

A handwritten signature in black ink, appearing to read 'C. P. Kott', written in a cursive style.

DEAN, SC&SS

JNU, NEW DELHI
प्रोफेसर सी. पी. कोटी/Professor C. P. Kott
डीन/Dean
कंप्यूटर और प्रणाली विज्ञान संस्थान
School of Computer and Systems Sciences
जवाहरलाल नेहरू विश्वविद्यालय
Jawaharlal Nehru University
नई दिल्ली / New Delhi-110067



DECLARATION

I hereby declare that the dissertation work entitled “**A Model for Resource Allocation for Computational Grid using Genetic Algorithm**” in partial fulfillment for the requirements for the degree of “**Master of Technology in Computer Science and Technology**” and submitted to School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi-110067, India, is the authentic record of my own work carried out during the time of Master of Technology under the supervision of Dr. Zahid Raza. This dissertation comprises only my original work. This dissertation is less than 100,000 words in length, exclusive tables, figures and bibliographies.

The matter personified in the dissertation has not been submitted for the award of any other degree or diploma.



Krishan Veer Singh

Enrollment No. 11/10/MT/19

M. Tech (2011-13)

SC&SS, JNU

New Delhi India -110067

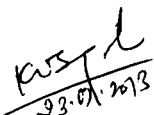
ACKNOWLEDGEMENT

I am very glad to express my sincere gratitude and thanks to my supervisor **Dr. Zahid Raza** for his guidance. I would like to express special thanks to Dr. Zahid Raza for many helpful discussions and his intellectual input to make dissertation work worthy. His extensive and invaluable research experiences were very helpful in my dissertation and the most important thing was the helping nature of him that contributes an important share in fulfillment of this work. The mythology, philosophy and problem solving methods learned by him have been very beneficial in this work and would be afterward.

I would like to express my thanks to Dean SC&SS JNU, Prof. Karmeshu in support to pursue my work in the School. Also my thanks go to School administration and librarian of software library and main library for supporting me, in whatever way they can, to make dissertation a success. Their support has been a real emphasize in completing this dissertation.

I would like to accord my sincere thanks to Mr. Mohammad Sajid, Mr. Taj Alam, Mr. Mohammad Shahid, Mr. Sumit Kumar, Mr. Yogendra Meena, Mr. Gagandeep Singh, Mr. Deepak Gupta, Mr. D. P. Sahu and Mr. Shiv Prakash Tiwari for their valuable suggestions for my dissertation work.

My parents and family members have been my strength through the long hours of study and research. I especially thank my father and mother for their patience, unconditional love and economical as well as moral support for completing this dissertation. Finally I would like to express thanks to each person & thing which is directly or indirectly related to my dissertation work.


23.07.2013
Krishan Veer Singh

Dedicated to my Parents

Table of Contents

Abstract	(i)
List of Acronyms	(iv)
List of Figures	(v)
List of Tables	(vi)
Chapter 1: Grid Computing	1
1. Introduction	1
1.1 Grid Computing	2
1.1.1 Grid Components	3
1.1.1.1 Remote Computing Resources	3
1.1.1.2 Heterogeneous Resources	3
1.1.1.3 Untrusting Resources	3
1.1.1.4 Dynamic Computing	3
1.1.2 Layered Architecture for Grid Computing	4
1.1.2.1 Fabric Layer	4
1.1.2.2 Connectivity Layer	4
1.1.2.3 Resource Layer	4
1.1.2.4 Collective Layer	5
1.1.2.5 Application Layer	5
1.2 Types of Grid	5
1.2.1 Clubby Analytics	5
1.2.1.1 Compute Grid	5
1.2.1.2 Service Grid	5
1.2.1.3 Information Grid	5
1.2.1.4 Intelligent Grid	6
1.2.2 Sun Microsystems	6
1.2.2.1 Cluster Grid	6

1.2.2.2	Campus Grid	6
1.2.2.3	Global Grid	6
1.2.3	IBM	6
1.2.3.1	Computational Grid	6
1.2.3.2	Scavenging Grid	7
1.2.3.3	Data Grid	7
1.3	Application of Grid Computing	7
1.4	Challenges in Grid Computing	8
1.5	Grid Computing Versus Other Technology	8
1.5.1	Grid Computing Versus Distributed Computing	8
1.5.2	Grid Computing Versus Cluster Computing	9
1.5.3	Grid Computing Versus Cloud Computing	10
1.6	Grid Scheduling	10
1.6.1	Components of Grid Scheduling	11
1.6.1.1	Resource Discovery	11
1.6.1.2	Resource Selection	11
1.6.1.3	Job Execution	12
1.6.2	Types of Grid Scheduling	12
1.6.2.1	Local Vs Global	12
1.6.2.2	Static Vs Dynamic	12
1.6.2.3	Cost constrained techniques	13
1.6.2.4	Hybrid of static and Dynamic techniques	13
1.6.2.5	Optimal Vs Suboptimal	14
1.6.2.5.1	Approximate Vs Heuristics	14
1.6.2.5.2	Distributed Vs Centralized	15
1.6.2.6	Adaptive scheduling	16
1.7	Scheduling –NP hard problem	16
1.7.1	Class of problems	16
1.7.1.1	P class of problems	16

1.7.1.2	NP class of problems	16
1.7.2	Approaches to scheduling Problem	16
1.7.2.1	Approximate Algorithm	17
1.7.2.2	Heuristic Algorithm	17
Chapter 2:	Genetic Algorithm	18
2.	Soft Computing	18
2.1	Genetic Algorithm	20
2.1.1	History	21
2.1.2	Biological Background	21
2.1.3	Fitness Function	22
2.1.4	Search Space	22
2.1.5	Termination Condition	23
2.2	Structure of Genetic Algorithm	23
2.2.1	Steps of Genetic Algorithm	24
2.2.2	Operators of Genetic Algorithm	25
2.3	GA Parameters	27
2.3.1	Encoding	28
2.3.2	Selection	29
2.3.3	Crossover and Mutation	31
2.3.3.1	Binary Encoding	31
2.3.3.2	Permutation Encoding	33
2.3.3.3	Value Encoding	33
2.3.3.4	Tree Encoding	34
2.4	Travelling Salesman Problem	34
2.4.1	Solution of TSP using Genetic Algorithm	35
2.4.2	Algorithm	35
2.4.3	Conclusion	38
Chapter 3:	The Proposed Model	39
3.1	Proposed Scheduling Strategy using GA	39

3.1.1	Data Structure used in the Model	42
3.1.2	Notation used	43
3.1.3	Fitness Function	44
3.2	The Proposed Algorithm	44
3.3	Illustrative Example	47
3.4	Simulation Experiment	51
3.5	Observation	56
Chapter 4: Conclusion and Future Scope		58
References		60

Abstract

As the science and technology advances the need for high computation power is also witnessing an increase day by day. Since science is based on analysis, visualization and collaboration of available data so that useful information can be extracted, high computation power is needed. Also, since scientific and engineering problems are getting more and more complex, user needs them to be solved precisely and accurately within the limited time. Due to this feel need of high computational power, the term parallel computing comes in to the picture. In parallel computing, multiple computer or processors work together to solve a single problem or to achieve a goal. This meets the requirement of improved performance and also the need of memory is satisfied. Parallel computing is of two types, parallel processing and distributed computing. In parallel processing, several no. of processors work together to solve a problem with each processor handling a section of code and is allowed to exchange the information between them. In distributed computing system there are multiple computers with multiple software components that are working together to achieve a single goal. In distributed system the computers can be at same physical location or globally distributed and connected via high speed network. These distributed systems include cluster computer, super computer and storage systems etc.

Grid computing is a huge collection of computers that are globally distributed and connected via high speed internet connections working together to achieve a single goal. Grid is a distributed system in which workloads are non-interactive involving a large no. of files. Grid computing is different from cluster computing and conventional high performance computing system as the grid systems are loosely coupled, heterogeneous and dispersed across the globe. Grid computing is used in various scientific problems like protein folding, earthquake and financial modeling simulation etc. In grid the individuals can also share their processing cycles voluntarily in projects like SETI@home and folding@home etc.

In Grid, the resources are dynamic in nature, heterogeneous and globally distributed across the globe having different ownership. Therefore any resource can leave or join the system at any time. This makes the scheduling of job in the grid to belong in the category of NP-hard problems having very large search space with changing environment accordingly as the resources leaves or join the system. In this category, exact solution cannot be determined but approximate solution can be obtained which is acceptable and considered as good as the exact solution. Such type of problem cannot be solved by traditional method because mathematical modeling is not easy. To handle NP-hard problems soft computing techniques are used which includes Neural Network, Fuzzy Systems, Probabilistic Reasoning and Evolutionary Computing.

Among all soft computing techniques, evolutionary computing is considered as the good one as closely related to the nature. Evolutionary computing, a global search paradigm, includes Evolutionary Strategies, Evolutionary Programs, Genetic Algorithms and Genetic programming. Genetic Algorithm was inspired by the Darwin theory of evolution i.e. "fittest of the survival". Genetic Algorithms (GA) are most common in all evolutionary paradigms. GA mimics the process of natural evolution and finds its use in solving computing and optimization problems. In GA, a population of chromosomes, generally a sequence of bits is randomly selected. This population is then transformed into some new population by the use of some methods which are similar to the natural selection by the use of operators which are inspired by the natural genetic operators like crossover, mutation and inversion operator.

Fitness function is the deciding criteria for the natural selection of a population. According to that, the chromosomes having optimum fitness value can survive and are allowed to reproduce offspring. Among all chromosomes that survive the fittest chromosomes can reproduce to produce new offspring than the less fit chromosomes. Then the crossover operator performs crossover operation on the selected chromosomes based on certain features like bit location in the parent chromosomes to produce new offspring having same size. The mutation operator flips/replaces the bits at selected locations by a certain value. The inversion operator reverses the order of a subsequence in a chromosome.

When the new generation of a population is completed, then it is checked for the stopping criteria. If stopping criteria is met then the algorithm is stopped otherwise the fitness value is again evaluated for the chromosomes of this generation and the whole process is repeated till the stopping criteria is not met.

For any job execution the minimization of the TAT is of utmost importance and is expected from any scheduling scheme to meet this objective. In this work we have extended the work done in [28] to minimize the TAT of a job for a computational grid. The same is done using GA which is an established soft computing tool for such kind of combinatorial problems. The model analyses the performance of the scheduling schemes viz. rank selection and roulette wheel selection for various mutation instances. The simulation study reveals the effectiveness of the model.

List of Acronyms

API	Application Programming Interface
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DNA	Deoxyribonucleic Acid
GA	Genetic Algorithm
GHz	Gigahertz
GIS	Grid Information Service
GP	Genetic Programming
GS	Grid Scheduler
HPC	High Performance Computing
IaaS	Infrastructure as a Service
IBM	International Business Machines Corporation
IMC	Inter Module Communication
LAN	Local Area Network
LHC	Large Hadron Collider
MATLAB	Matrix Laboratory
MCT	Minimum Completion Time
MET	Minimum Execution Time
NEC	Node Execution Cost
NP	Non-deterministic Polynomial time
OLB	Opportunistic Load Balancing
PaaS	Platform as a Service
RS	Rank Selection
RW	Roulette wheel Selection
SaaS	Software as a Service
SDK	Software Development Kit
SETI	Search for Extraterrestrial Intelligence
TAT	Turnaround Time
TSP	Travelling Salesman Problem
VO	Virtual Organization

List of Figures

Figure 1.1: A Generic view of Grid Computing	3
Figure 1.2: Hour Glass Model (Layered Architecture Model) for Grid Computing	4
Figure 1.3: A logical Grid Scheduling Architecture	12
Figure 1.4: Euler diagram for P, NP, NP-complete, and NP-hard set of problems	17
Figure 2.1: Steps for Genetic Algorithm	24
Figure 2.2: Crossover in Tree encoding	34
Figure 3.1: Computational Grid	40
Figure 3.2: Directed Acyclic Graph for a Job	47
Figure 3.3: Mutation after 5 th Generation	53
Figure 3.4: Mutation after 10 th Generation	54
Figure 3.5: Mutation after 15 th Generation	54
Figure 3.6: Mutation after 5 th Generation	55
Figure 3.7: Mutation after 10 th Generation	55
Figure 3.8: Mutation after 15 th Generation	56

List of Tables

Table2.1: Showing Cost Matrix of TSP	36
Table 3.1: Job j_0	48
Table 3.2: Node Attributes in a Cluster	48
Table 3.3: Hamming Distance (D) between different Nodes	49
Table 3.4: IMC Matrix	49
Table 3.5: Processing Time of Various Modules on Various Processors	50
Table 3.6: A Typical Chromosome	50
Table 3.7: Calculated Values1	50
Table 3.8: Calculated Values2	51
Table 3.9: Parameters Used	52

Chapter 1

Grid Computing

1 Introduction

Around 80's when there was huge demand for computational power, the birth of parallel computing was witnessed. Parallel computing is of two types, Parallel Processing and Distributed Computing. In Parallel Processing several CPU's or computers for which the frequent communication between the processors is needed are used to solve scientific problems. In Distributed Computing several computers are used to solve these problems. In order to solve a problem using Distributed Computing, the problem is divided in to several independent tasks that can be executed in parallel. Because of the availability of high speed internet and low cost commodity components makes the scientist and engineers to do their computation over these distributed systems, which includes cluster computer, super computer, storage systems, data sources and some other devices being used as unified resources. The high speed internet connection has made possible the seamless access to distributed resources and interaction among them, giving birth to a new computing era known as "GRID COMPUTING".

Since independent codes of same application can be executed in parallel on different machines it can achieve end to end system capabilities of distributed networked system. This is known as "METACOMPUTING". Later on as the technology advanced and the new programming language tools were designed several new projects were carried out by many universities, laboratories and other organization. In 1996, the researcher in NSF supercomputing centers agreed on a new name "POWER GRID" or simply "GRID". Since then the term "METACOMPUTING" has got replaced by GRID. Because of the growing interest in Grid some standard was needed so that products can be produced in industry. This leads to the formation of Global Grid Forum [1].

1.1 Grid Computing

The term “Grid Computing” was first given by Ian Foster and Carl Kesselman in his book “The Grid: Blueprint for a New Computing” in 1998. Accordingly it is defined as “Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.” Later on in 2000 Steve Tuecke and Foster published an article named “The Anatomy of the Grid” in which Grid Computing is redefined as “coordinated resources sharing and problem solving in dynamic, multi-institutional organization”. In this definition the key concept is the ability of negotiation of sharing the resources among the participating parties [2]. In this way in literature Grid Computing is defined in so many ways, some of the other definitions are as below:

The Globus Alliance defined the GRID as, “The infrastructure that enables the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organizations.”

In general Grid Computing uses the shared resources of huge computer network to solve a single problem either of scientific, research or technical domain where large numbers of processing cycles are needed.

Initially it was known as “GRID” as it shows resemblance with the “Electrical Grid”. Electrical Grid shows analogy with the Computational Grid in terms of structural design. Electrical Grid provides power to thousands of devices via complex physical connection. In terms of physical characteristics Electrical Grid is highly heterogeneous, managed by different organization also they differ in terms of power consumption by a consumer, duration and quality of services they needed and the amount they can pay.



Figure 1.1: A Generic View of Grid Computing [6].

1.1.1 Grid Components

In Grid Computing possibly the remote, heterogeneous, untrusting and dynamic computing resources are seamlessly accessed. These terms can be defined as [3]:

1.1.1.1 Remote Computing Resources: Resources which are connected through a LAN are said to be local resources and resources which are widely distributed across the globe are said to be remote resources.

1.1.1.2 Heterogeneous Resources: The resources are said to be heterogeneous since they may or may not be using same operating system, it may have same or different kind of hardware configuration, it may be a single resource or a group of resources of same or different kind forming a cluster or it may be a supercomputer etc.

1.1.1.3 Untrusting Resources: The Grid user may not trust the user on the other side whose system is a part of the Computational Grid, since he/she may alter the executing programme, so security issue is also there.

1.1.1.4 Dynamic Computing: Since Grid Computing uses the unused processing cycles of the systems that are part of the Grid System. It might be possible that some systems are free at a particular time but remain busy for some other time. This dynamism is owner dependent (owner of the individual systems) [3].

1.1.2 Layered Architecture for Grid Computing

The most popular representation of Grid Computing has been proposed by Ian Foster because of its shape it is often referred as “The Hour Glass Model”, which is wider at the top and bottom and thinner at the middle part which plays the role of middleware in the Grid System. Grid System architecture provides services and includes several protocols which are shown below in the Figure 1.2 [4].

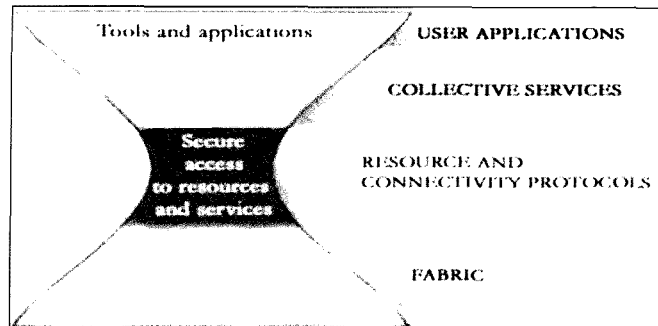


Figure 1.2: Hour Glass Model (Layered Architecture Model) for Grid Computing

1.1.2.1 Fabric Layer: This is the bottom most layer of this Layered Architecture.

This layer acts as an interface and makes available the shared resources such as network bandwidth, memory and CPU time etc. Resources are managed by some standard protocols. This layer allows the sharing of resources among the virtual organization.

1.1.2.2 Connectivity Layer: This layer provide the secure and easy access to the data exchange between the Resource Layer and the Fabric Layer, for this purpose this layer specifies some protocols regarding authentication and communication. Communication Protocol permits the exchange of data between different layers and Authentication Protocol are meant for secure cryptographic mechanisms for authenticate the users and resources.

1.1.2.3 Resource Layer: This Layer is responsible for managing the resources by specifying the protocols that handles the shared resources. Since this layer is defined over the Connecting Layer, it also defines API (Application Programming Interface) and SDK(Software Development Kit) for secure navigation, control, billing and accounting etc.

1.1.2.4 Collective Layer: This Layer lies just above the Resource layer. This Layer handles access and manage the coordination among the multiple resources, allocation and scheduling of task on multiple resources, data replication etc.

1.1.2.5 Application Layer: This is the top most layer which provide interface to the users and admin to control/operate the Grid Computing Environment.

1.2 Types of Grid

Grids have so many variations as The Grid Technology or Grid Architecture is in its evolving stage. Therefore on the basis of one's need, its functionality and architectural design it can be of several kinds [5, 29].

No boundary exists between various forms of Grids. A Grid can be a combination of one or more of these. According to different organization, the Grids can be classified as:

1.2.1 Clubby Analytics: Clubby Analytics, a research organization categorizes the Grid in to four types and they are as described below:

1.2.1.1 Compute Grid: These Grids are designed to exploit the unused processing cycle of the large pool of the shared CPU. These kinds of Grid are used in scientific, engineering, space research programs to meet their high throughput and computational needs.

1.2.1.2 Service Grid: It provides the reliable, fault tolerance and high speed communication among the different Grids to facilitate the environment that allows the interoperability and different applications to run on varied operating environment.

1.2.1.3 Information Grid: These Grids provides peer to peer services, acts as collaborative computing and also allows file sharing or handling the data that are distributed or stored over the heterogeneous databases. Sometimes these Grids are also known as "Data Grids".

1.2.1.4 Intelligent Grid: These kinds of Grids automatically manage themselves. These are the basic network of different Grid interconnection that

manages the storage, network hardware management and software enhancement etc.

1.2.2 Sun Microsystems: Sun Microsystems classified the Grids on the basis of geographical dispersion/location of server. They classified the Grids in three types and they are Cluster Grid, Campus Grids and Global Grids as described below:

1.2.2.1 Cluster Grid: This kind of Grids consists of one or more systems working together and is used to satisfy the needs of a user. They are used whenever high throughput and high performance job are needed to be executed. These kinds of Grids are used and managed by small no. of users such as a department and in a project etc.

1.2.2.2 Campus Grid: This kind of Grid are used as the demand of high computation is needed, then the different cluster from different organization are combined to form a Campus Grid. Multiple projects can be executed on Cluster Grid and share the resources in a cooperative fashion.

1.2.2.3 Global Grid: As the requirement increases beyond the availability of resources, Campus Grids are combined to form a Distributed arrangement of Campus Grid known as Global Grid. In Global Grid the shared resources are used in a cooperative way to satisfy the needs of the application and the user as well.

1.2.3 IBM: IBM a leading research organization categorized the grid in to three types as listed below:

1.2.3.1 Computational Grid: In this kind of Grids the resources are clubbed together to satisfy the needs of high computation. In Computational Grids most of the machines are high computation server. For high throughput and better response time Computational grids are used.

1.2.3.2 Scavenging Grid: This kind of Grids consist of large number of desktop machines that are used or shared by the users. The desktop owner has full

control over their resources. Scavenging Grid uses the unused processing cycles and other resources of the shared machine/desktop.

1.2.3.3 Data Grid: This Grid are used when huge amount of data are needed to be processed, analyzed which may be of same or different format, may or may not be located at different geographical locations [5].

1.3 Applications of Grid Computing

Grid makes an optimized use of the shared resources. It uses the unused processing cycles of the CPU that are shared to the network which otherwise could be wasted and help achieving the high computation resources which can be used to solve a complex problem within the limited time. It can find its use in many organization, some of them are listed as below [6]:

- **Government Organization:** These organizations use Grid Computing during disaster management like earthquake, Tsunami and situation like flooding so that they can minimize the loss. These organizations also use the Grid Computing for urban planning and economic modeling etc.
- **International organization:** International Organization like World Bank also uses Grid Computing so that they can share their data archive more effectively and simply.
- **Military:** Countries like United States of America already started using Grid Technology for simulating artificial military operations, for simulating different kind of environment needed for training, for military application like designing an automated tank and simulation and effect of nuclear bomb if it gets detonated etc.
- **Scientist and Researcher:** These people use the Grid Computing for weather forecasting, for development of drugs, recently it is used in the fight against dangerous H5N1 avian flu virus, and Researcher uses the Grid technology for knowing the fact that how universe comes in to picture, they used it in LHC Experiment also they used in many projects like SETI@home, also used in World Community Grid etc.

- **Industry:** This technology can also be used in industry to carry out large scale modeling or computation where resources are distributed at many sites etc.

1.4 Challenges in Grid Computing

As the need of computation increases in the industry, this lead to excessive use of resources which are very hard to manage in terms of cost as wells as environment perspective. Since large numbers of system are working in shared manner in Grid to satisfy the requirement, resources are located at different locations and under different administration which make the problem more challenging. The process of managing and scheduling the resources over the Grid is very challenging as they are globally distributed, having different kind of hardware and software and also owned by different individuals or organizations with their own policies, having varying load/availability and different cost models. Since resources are distributed and may be owned by individuals or organization, they may leave or join the system at any time causing extra burden on the Grid System i.e. resources are dynamic in nature in Grid System. This introduces problem like policy extensibility, online control of the resources, resource allocation, scalability of resources and interoperability among the heterogeneous resources along with security issues since jobs are executed over different systems having multiple administrative control. These are some challenges that are needed to be resolved for smoother working of the Grid.

1.5 Grid versus Other Technology

Similar to Grid Computing there are some other contemporary technologies like Distributed Computing, Cloud Computing, Cluster Computing etc. which have some similarities and dissimilarities. Some of them are mentioned below:

1.5.1 Grid Computing Vs Distributed Computing

Around 80's Distributed Computing came in existence when increased network bandwidth and CPU computation power were two advances in technology helping in solving complex problem by putting together a large number of systems connected through high communication network. It is actually a subset of Grid Computing. In distributed Computing the systems have limited memory and

processing power, also manages thousands or some hundreds of computers. On the other hand Grid Computing is concerned with efficient utilization of resources with optimal workload management. Further there is no restriction on the number of resources i.e. Grid Systems are scalable without any restriction on number of user.

Grid Computing includes computation over multiple administrative domains and also the systems or resources are distributed across the globe which makes the Grid Computing a step ahead of Distributed Computing. Grid Computing also includes issues like multiple administration policies, security issues for authentication and authorization.

1.5.2 Grid Computing Vs Cluster Computing

In terms of architecture and computational ability, Grid is far ahead of Cluster Computing. Grid Technology includes varied computing resources which are globally distributed. Cluster can be a part of a Grid System. In Grid, sharing, selection and aggregation of resources is there, which are distributed across the globe. Also they are different in terms of architecture and computational capabilities. This includes storage systems, data sources, supercomputers and specialized machines owned by particular organization for solving complex problems which needed huge resources. Cluster Technology is used for specific purpose and objective such as database services etc. Cluster can be scalable up to 100's of computer whereas Grid can be scalable up to millions of computer. In Cluster the systems are of same kind and are located at same physical location connected with high speed connections. It appears as a single system image. In Grid, systems are loosely connected, distributed job management and scheduling is there since systems are distributed whereas in case of Cluster the systems are tightly coupled, centralized job management and scheduling is there [7].

Initially the term Cluster is used for group of servers. The server is the platform where the services associated with database and application resides. Therefore server plays a vital role in providing availability and high performance.

Since, Cluster includes multiple servers, provides uninterrupted services thereby increasing the performance.

1.5.3 Grid Computing Vs Cloud Computing

Cloud Computing allows the user to free from the burden of handling and managing the huge hardware and software in terms of cost and labor. Cloud provides the researcher or industrialist the three elements of IT being IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service). In Cloud Computing, the business is executed on virtual resources and the users have to pay according to their requirement of hardware, software or platform [8, 32, 33, 34].

In Grid Computing the resources are shared and globally distributed having multiple administrative domains. It provides the suitable environment for running the application or executing the problems. Since, in Grid Computing the resources are shared therefore any complex problem first broken in to smaller module in such a way that they can be executed in parallel on distributed machine. The Grid Computing environment is more flexible and viable than Cloud. In Grid there is no problem of load server collapse. Even if it occurs then other distributed systems are there to handle it this feature is not there in Cloud until load balancer is not used, also upgrading of software and hardware is easier in Grid Computing, since resources are distributed and managed by independent owner.

Cloud has many advantages over Grid Computing. In spite of these none can replace the Grid because all these technologies (Cloud Computing, Grid Computing and HPC) have their own place.

1.6 Grid Scheduling

Grid Computing is a form of distributed computing making computer power as easy to access as power grid and work like a super computer. In grid computing the computers are loosely coupled network computers which executes very large jobs s.a. DNA analysis, earthquake, flood and analysis of protein structure etc. These are very large applications requiring huge computer resources and a huge amount of time, sometime

taking a few days or even a few weeks to complete their execution. This execution time can be minimized by proper task scheduling since the delay in one task can affect the completion time of the entire application. Scheduling is the way processes are assigned to run on different resources. The main goal of job scheduling is to minimize the execution time and minimize/avoid the wastage of the CPU cycles. A good job scheduling strategy should be taken into account since the grid possesses the characteristics like geographically distributed computer systems, heterogeneity of hardware and software resources, dynamicity of shared resources, multiple administrative control and resource coordination [29, 36].

1.6.1 Components of Grid Scheduling

Since in Grid Computing the resources are heterogeneous and are globally distributed, it is necessary to know what kind of resources and how many of them are available beforehand. After collecting all information about the resources they are selected on the basis of the requirements [9, 10].

1.6.1.1 Resource Discovery: Since in a Grid, the resources are heterogeneous, globally distributed and dynamic in nature, it is of great importance to know the status of the available resources beforehand. Grid Information Service (GIS) collects all information about the available resources and makes a list of them. This includes CPU capacity i.e. the computing power, memory size, network bandwidth etc. This information is needed to be updated at regular intervals of time. On the basis of the above information the Grid Scheduler takes the decision.

1.6.1.2 Resource Selection: This phase of selecting the resources comes after the discovery of them which filters the unwanted resources. This phase of selecting the resources first collects the information about the application, its requirements and then on the basis of these chooses the best resources from the large list of available resources. This method of selecting the resources is quite easier for sequential jobs but gets more and more complex for complex jobs that need to be executed in parallel.

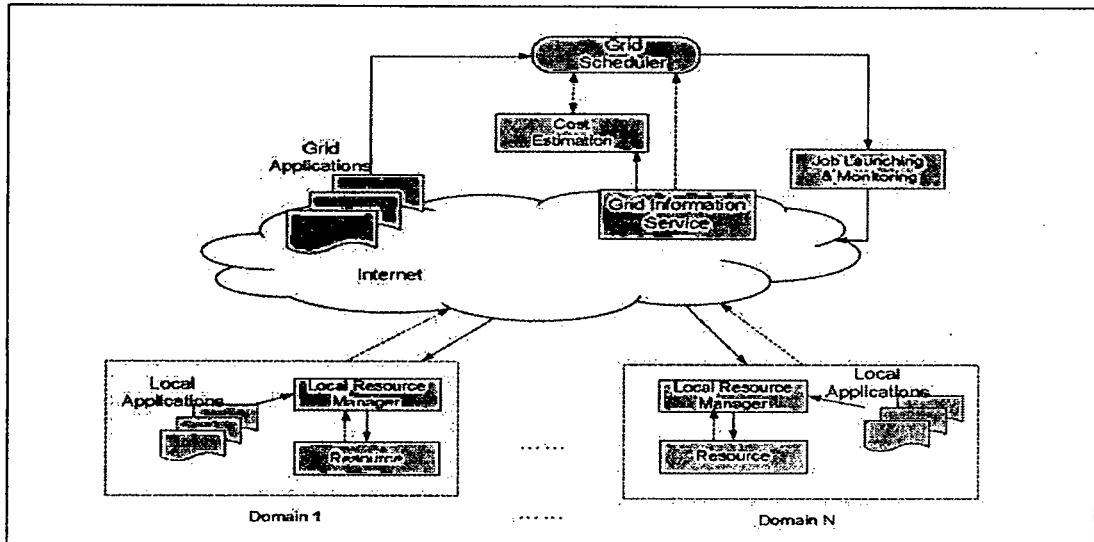


Figure 1.3: A logical Grid Scheduling Architecture [9].

1.6.1.3 Job Execution: The last phase of job scheduling is the execution of the jobs. This phase is the most complex step during the whole process because it needs to be monitored as the resources are dynamic in nature. Also it might happen that job execution stops due to hardware or software failure or may be due to network problems. This demand for the job to be scheduled again.

1.6.2 Types of Grid Scheduling

In Grid, the resources are heterogeneous, globally distributed around the globe, dynamic in nature, having different ownership etc. therefore depending on the criteria, different kind of scheduling methods are there. There could even be some overlap and may not have clear distinction among them. Some of them are discussed below [9,10,11, 30]:

1.6.2.1 Local Vs Global: Local Scheduler decides how the process which resides on a single CPU is scheduled on a single machine whereas in Global Scheduling the scheduler has all the information about the available resources in the system. On the basis of that information, it allocates the process to multiple resources.

1.6.2.2 Static Vs Dynamic: These Scheduling policies indicate the time at which the scheduling decisions are made. In static scheduling prior to scheduling the application or job, the systems have all information about the available

resources and the job requirement. In case of dynamic scheduling, task allocation is on the fly as the job executes.

In Static Scheduling every task is once assigned to a resource. Thus this arrangement of assigning a resource to a task is static. It makes possible the estimation of cost of computation prior to the actual execution of task. From the programmer point of view it is beneficial to know in advance the scheduling arrangement so that he/she can make the cost of computation simpler. This model allows a “global view” of task and costs. But this estimation of cost is not adaptive since based on static scheduling because it might happen that any number of the nodes or network may get fail during the task execution leading to higher response time than expected. To avoid such kind of scenarios rescheduling mechanisms are introduced at the cost of overhead for task migration.

Dynamic Scheduling is used when estimation of computation cost is difficult since jobs are arriving dynamically in the system. This kind of scheduling is also known as “Online Scheduling”. There are two major components in scheduling, one is system state estimation and another is decision making. Prior to taking any estimation decision about the assigning of task to a resource the scheduler has information about the system state throughout the Grid. To make the system work properly, dynamic load balancing is used.

1.6.2.3 Cost constrained techniques: This approach is an improvement of balanced-constrained approach as this not only considers the balance among the resources but also considers the communication cost. This approach is more flexible and efficient but objective of this approach is to release the running jobs and assign the resources to the waiting jobs rather than load balancing and cost optimization.

1.6.2.4 Hybrid of static and dynamic techniques: This technique is better since it takes the advantage of static scheduling and at the same time captures the

dynamicity of the task and resources and takes the appropriate decision of scheduling.

1.6.2.5 Optimal Vs Suboptimal: Optimal assignment can be made if all information about the resources and the job is known, based on some criteria such as minimum make span and maximum resource utilization. Since scheduling is NP-complete, it is difficult to find the optimal solution leading to accepting the suboptimal ones. Accordingly the methods can be divided as:

1.6.2.5.1 Approximate Vs Heuristics: Approximate algorithm uses formal method of computation which gets satisfied when a sufficiently good solution is obtained. In this they do not search the entire search space. Where any metric is available it reduces the time to find an optimal solution. Some of the approximate algorithms are described below.

- **OLB (Opportunistic Load Balancing):** This is the simplest approach of assigning the task to a resource. It does not consider the expected execution time.
- **MET (Minimum Execution Time):** This approach considers the expected execution time of each task on each machine and on the basis of that selects the one having minimum execution time.
- **MCT (Minimum Completion Time):** This considers the task for scheduling having the minimum completion time.
- **Min-min:** This approach executes the task fastest and for this first it selects the best machine for each task and then it selects the task with minimum completion task.
- **Max-min:** In this approach first the best machine is selected for each task and then the task with maximum completion time is

selected for execution. This leads to better load balancing and better total execution time.

- **GA (Genetic Algorithm):** GA is an evolutionary technique for larger search space. The general procedure includes, generation of a set of initial population by heuristic methods which is a set of chromosomes represents the mapping between the task and the machine. Fitness value of each chromosome is then evaluated which is the total completion time. Goal of GA is to find the chromosomes with optimal fitness value. Crossover and mutation are the other operators used on the selected chromosomes. Crossover is defined as the swapping of certain subsequences in the selected chromosomes. Mutation includes the replacing certain subsequences with new task machine mapping choices that are new to the chromosomes. After performing these operations the new generated population is evaluated for fitness value. This process is repeated over and over until the stopping criteria are met. Stopping criteria can be all chromosomes converge to the same mapping, cost bound is met or no improvement in recent evaluation.

1.6.2.5.2 Distributed Vs Centralized: In the case of dynamic scheduling the decision for global scheduling can be handled either by centralized scheduler or decentralized scheduler. In the case of grid scheduling, where any number of resources or jobs can leave or join the system at any time, rescheduling is needed dynamically. In such cases centralized scheduling shows some advantages but at the same time suffer from scalability, fault tolerance etc. Decentralized or distributed scheduling is preferred over centralized scheduling instead of weak efficiency.

1.6.2.6 Adaptive Scheduling: An adaptive scheduling is one in which the scheduling decision is changed dynamically depending on the previous, current or future status of the resources.

1.7 Scheduling - NP hard problem

The Grid computing includes large no of resources that are globally distributed and are used for satisfying the need. Since the usage cost of these resources are very high sometimes it reaches to some millions and sometime higher so it is better to use the resources in optimized fashion so that we can reduce the cost. In case of Grid, scheduling problems are optimization problem i.e. need to generate a schedule that minimizes the certain objective function.

1.7.1 Class of problems: P or NP

A scheduling problem is said to be polynomial time algorithm if it is bounded by the input size n , the number of resources and the number of bits that are used to represents the largest integer i.e. its running time is bounded by a polynomial in input size. On the basis of computational complexity the problems can be classified as P class and NP class of problems [12,13,14].

1.7.1.1 P class of problems: P is the set of decision problem which can be solvable in polynomial time, P represents polynomial time.

1.7.1.2 NP class of problems: The set of problem whose solution can be verified for its correctness. The solution of NP class of problem can be verified in polynomial time if the correct answer can be guessed. Here NP means non-deterministic polynomial time. This NP class of problem can be further classified as:

- **NP complete class of problems:** A NP complete problem can be said to be solvable in polynomial time if and only if all other NP complete problem can be solved in polynomial time.
- **NP hard class of problems:** A problem is said to be NP hard, can be solvable in polynomial time if and only if all other NP

complete problem can be solved in polynomial time. For such kind of problem no such polynomial time algorithm is known. All NP complete problems are NP hard but vice-versa is not true. NP complete problem are said to be the hardest problems.

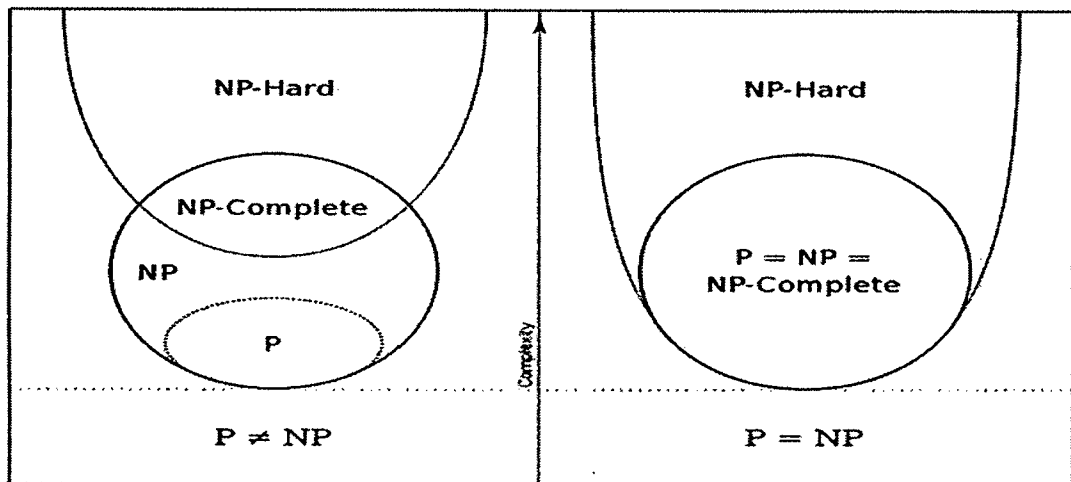


Figure 1.4: Euler Diagram for P, NP, NP-Complete, and NP-hard Set of Problems [15].

1.7.2 Approaches to scheduling problem

Given a scheduling problem we need to determine its computational complexity either by designing its polynomial time algorithms to find its solution or by proving that the problem is NP hard problem. In the case of Grid Computing the scheduling problem is NP hard problem, which can't be solved by polynomial time algorithm but the sub optimal solution can be found by using the following two methods:

1.7.2.1 Approximate Algorithm: These algorithms help to prove that obtained solution is close to the actual solution. They produce the solution in polynomial time. The calculated solution may or may not be the optimal one.

1.7.2.1 Heuristic Algorithm: These methods are used to find the feasible solution in reasonable time by performing the experiment a number of times. They are not guaranteed to be the optimal solution. These methods are simple and effective. Some of the heuristic methods are Genetic Algorithm, Tabu Search and Simulated Annealing etc.

Chapter 2

Genetic Algorithm

2 Soft Computing

Prof. Lofti Zadeh, who developed the concepts of fuzzy logic, first introduced the term Soft Computing. Soft computing is used where approximate solution is considered as good as the exact solution, for example NP complete problems, for which accurate or precise solution cannot be obtained in polynomial time.

Soft Computing is tolerant of uncertain, imprecise, approximate and partially true results whereas Hard Computing involves a precisely stated analytical model and computational time. Soft Computing takes the human mind as a role model [18].

Soft Computing includes:

- **Fuzzy Systems:** The concept of Fuzzy Systems is given by Dr. Lofti Zadeh. The behavior of Fuzzy Systems is described by Fuzzy Set Theory and Fuzzy Logic. With the help of Fuzzy Systems it is quite possible to translate the subjective human knowledge which is impossible to quantify, using mathematical calculus with some level of uncertainty because of imprecise tools used for measurement or vagueness in the language. It is used for modeling real world problem which are inherently imprecise. It can manage the numerical data and linguistic knowledge both. Fuzzy System finds its various applications like Robot Vision, Information Retrieval and Navigation System etc.
- **Neural Network:** Architecture of Neural Network is inspired by the functioning of human brain. It is an attempt to adapt the way the brain function such as process the input information and produces the output accordingly also it possess learning capability. They acquired the knowledge through learning. It is a powerful tool for modeling the complex non-linear real world problem. It can identify and learn correlated patterns between input and output set. Also it possesses the capacity to predict after being trained.

Probabilistic Reasoning: It uses the concepts of probability theory and deduction logic to exploit the problem. It tries to find the natural extension of logical truth table the results they define are derived through probability expression. Problem with this is that to compute the complexity of probabilistic reasoning it tries to multiply the complexities of both the probability and logical components

Evolutionary Computing: A technique inspired from natural evolution process. This method is widely used on a variety of problems having very large search space. It is not possible to obtain an exact solution but suboptimal solution is considered as the best solution obtained within the limited time or under certain constraints. Evolutionary Computing uses the evolution algorithm over a set of population to obtain the suboptimal solution. This algorithm includes the natural selection of the child population followed by crossover and/or mutation under the environmental condition/pressure, which increases the fitness of the obtained child, over the time these steps are repeated until the termination condition is not met, to give the best child and hence the suboptimal solution is obtained.

Soft Computing is a totally new multi-dimensional field, which is used to design a new kind of Artificially Intelligent machines, known as computationally intelligent machine.

Soft computing techniques can be used to serve the following goals [18]:

- To design an intelligent machine capable of solving a real world problem which otherwise cannot be solved efficiently by mathematical modeling.
- To make use of approximate, uncertain, imprecise and partially true result in order to attain human like decision power.

In approximation, the model features are similar to the real one but not same. In uncertainty we are not sure that feature of model is similar or same as that of entity. Imprecision tells that the model features (quantities) are not same as the real one but close to that of the real one.

2.1 Genetic Algorithm

Genetic Algorithm (GA) is a general purpose search algorithm based on the process of evolution observed in the nature. It is a heuristic search algorithm, a part of evolutionary computing and a booming area of Artificial Intelligence, inspired by the Darwinian theory of evolution “survival of the fittest”, for getting optimum solution of a problem, which is not solved by traditional methods. In other words, GA is an adaptive heuristic search algorithm which uses the idea of natural selection and genetics. Genetic Algorithm directs the search to the region of the sample space where better results can be obtained. Genetic Algorithm can be applied to wide variety of application, for ex-computer games, scheduling, transportation problem, TSP, medical, adaptive control and stock market trading etc [19]. In general, GA can be classified into two categories viz. Single optimization GA and Multi-objective GA. In the case of single objective optimization GA, one tries to obtain the best solution among all alternative solution by optimizing the single objective function whereas in the other case, there are multiple conflicting objectives that are to be optimized to produce a set of optimal solutions using the Pareto optimality theory, the optimal set of solutions must satisfy all the objectives as best as possible [24].

Some of the advantages of GA are as follows [21], [26]:

- Easy to understand and covers the large search space.
- Support multi-objective optimization.
- Good for noisy environment.
- GA is as good as the objective function is.
- GA gives the solution of a problem through evolution process.
- It can solve any optimization problem, which can be encoded using chromosomes
- Easy to exploit the result.
- GA is good as good as the objective function is.
- Runs in parallel.
- The fitness function can be changed at each iteration if needed to increase the performance.

The important disadvantages of GA could be [26]:

- GA is very slow.
- We have to choose mutation rate, population size wisely; if the mutation rate is too high then they never converge to an optimal solution and if the population size is too small then search space is very small to find the solution.
- Designing the fitness function for any problem is very crucial.
- Convergence is always there, but time taken is uncertain depending on the objective function chosen.
- It does not always give the exact solution.

2.1.1 History

The term evolutionary computing was first coined by I. Rechenberg in 1960's in his work "Evolutionsstrategie". His idea was further used and developed by other researcher. The term Genetic Algorithm (GA) was first introduced by John Holland while studying cellular automata, developed by him, his students and colleagues at the University of Michigan. Holland incorporated his work in a book named "Adaption in Natural and Artificial Systems" published in 1975. Actually, Holland's goal was not to design algorithm for any specific problem but to understand the phenomenon of adaptation that occurs in nature, and how to incorporate this way of adaptation in computer science.

John Koza, in 1992 used the GA to evolve programs to perform certain task. This was named as Genetic Programming (GP) [16].

2.1.2 Biological Background

Cell is the basic building block of living organisms. Each cell has same set of chromosomes, which consists of genes, the basic unit of DNA. A string of DNA in a gene forms a particular trait. Each gene has its position in the chromosomes, called locus. The solution to a problem in Genetic Algorithm is called chromosomes, the parameter that is to be optimized.

TH22 862

Evolution is a method of searching the best among the huge number of possibilities. In biological terminology the huge possibility is the availability of set of possible genetic sequence and the desired solution is the fittest organism among all available possibilities. In other words, solving a problem using GA can be seen as looking for a solution which is the best among the others.

2.1.3 Fitness Function

To solve any problem using GA, we first require formulating its mathematical model in terms of a function. Then parameters that optimize the function of the model are determined. This is known as fitness function also known as the Objective function. Fitness function basically determines or analyzes the genes holding the data and returns the fitness value to quantify the fitness or suitability of the chromosomes. This results in selection of chromosomes having higher fitness value for producing the next generation. The better is the fitness, more chances are there of selecting those chromosomes to survive. The chromosomes having poor fitness value are discarded. The fitness function varies from problem to problem. The effectiveness of the fitness function determines how well a problem can be solved [25].

2.1.4 Search Space

If we are looking for solution to a problem which will be best among the other, then search space comes into picture. Search space is the set of all possible solutions that can exist for a given problem. Search space is defined by the domain where all feasible/possible solution (or the object among those desired solutions) are present, also known as state space. Each solution or point in the search space is known as the feasible solution and is marked by its fitness value. Search space changes at each step of evolution.

2.1.5 Termination Condition

There could be various terminating conditions for the GA to stop. A programmer can use either one or multiple terminating conditions as per the domain requirements. Some of these are listed as follows [20]:

- **Generation number:** A solution is obtained when maximum numbers of iterations have been run.
- **Evolution time:** when the specified time limit exceeds the process can be terminated.
- **Fitness threshold:** when the best fitness in the current population is less than the user specified fitness threshold value, when the aim is to minimize the fitness value. This also stops when the best fitness value in the current population becomes more than the user specified threshold value with the objective being maximization of the fitness.
- **Fitness convergence:** The evolution process stops when fitness converges.
- **Population convergence:** The evolution process also stops when population converges in next generations.
- **Gene convergence:** A termination method that stops the evolution process when a user-specified percentage of the gene of a chromosome is greater than the percentage of genes in a chromosome of current population.
- **Maximum iteration without termination:** when further specified number of iteration does not improve the specified result the evolution process is terminated.

2.2 Structure of Genetic Algorithm

GA is a method that produces or evolves to a better population or children at each step by choosing the best parent chromosomes from the available set of chromosomes. The newly generated chromosomes have better rate of survival than the previous generation as they have come from the fittest parents of the previous generation. This can be done by using methods of natural selection and other genetic inspired methods like cross-over, mutation and inversion etc. Genetic Algorithm evolves until a certain termination condition is met.

Genetic algorithm has the following elements [16]:

- A population of chromosomes
- Selection according to fitness (Fitness function, which is applied over the population)

- Cross-over to produce new children
- Random mutation of newly generated children/solution.

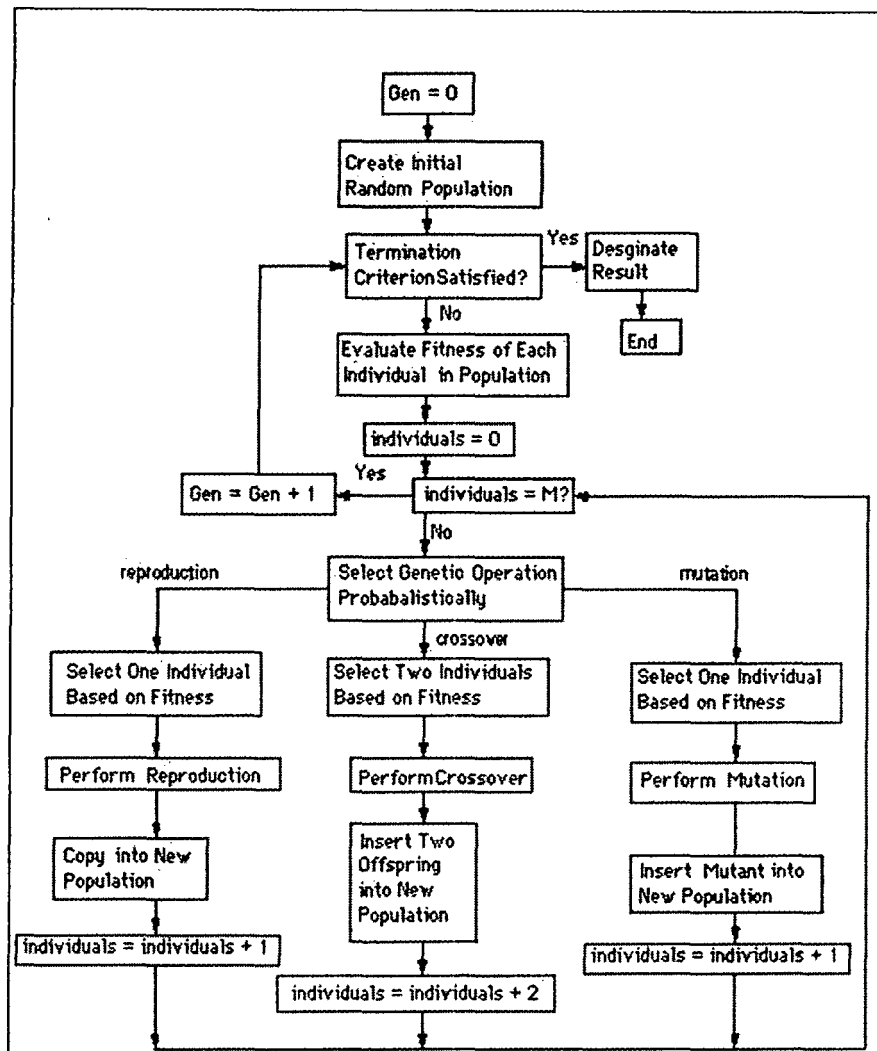


Figure 2.1: Steps for Genetic Algorithm [27]

2.2.1 Steps of Genetic Algorithm

The step by step process of GA can be represented as shown in Figure. The method presented can be summarized into various steps as shown below [16].

1. Start with the randomly generated population of n , l bit chromosomes.
2. Calculate the fitness $f(x)$ of each chromosomes x in the population.
3. Repeat the following steps until population of size n is created.
 - a) Select a pair of parent chromosomes from the current population. Selection is made on the basis of fitness i.e. higher the fitness more is the

probability of selecting the chromosomes. Same chromosomes can be selected more than once.

- b) With a crossover probability (p_c) crossover the parents to form new children. If no crossover was performed exact copies of the children is obtained. Crossover can be single point or it can be a multi-point.
 - c) Mutate the two offspring at each locus with probability p_m , known as mutation probability or mutation rate and place the mutated children in new population. If n is odd, a randomly selected offspring is deleted or discarded.
4. Replace the new population with the current one.
 5. If the termination condition is obtained then stop, otherwise repeat steps 2-4.

2.2.2 Operators of Genetic Algorithm

In order to solve a problem by GA sometimes it is not necessary to always encode the variable. But some problems can be only solved by encoding the variables. The most common method used for encoding is binary encoding. The length of string is decided by the accuracy needed. Encoding is done either by 0 and 1, real no or integer, depending on the problem. For e.g. after binary encoding chromosomes may look like [16]:

Chromosomes1	1100100101100001
Chromosomes2	1000010110001011

(a) Selection Operator

Selection determines which chromosome is to be chosen and how many offspring are to be generated. Selection of chromosomes can be done, first by assigning fitness value to each chromosome. Then on the basis of their fitness value they are selected with both assigning fitness value to each chromosome and selection being done on the basis of certain algorithm [25].

(b) Crossover operator

After encoding certain operation is performed on the chromosomes so that new offspring is generated/produced. One of them is crossover, in which two chromosomes

are selected for producing offspring. To accomplish this crossover point is selected randomly in each at same locus point. After this, in new offspring first part of chromosomes is selected from the first part of first parent and second part from second parent and so on. It can be shown below:

Chromosome1	11010 11000001110
Chromosome2	10010 00101010100
Offspring1	11010 00101010100
Offspring2	10010 11000001110

Here crossover is performed at a single point. It can even be multi-point depending on the encoding of chromosomes in a problem. Specific crossover operator is used for specific type of problem. This increases the performance of the GA. Generally, single point crossover is used when size of the string is small and multipoint crossover is used when the size is large.

(c) Mutation operator

After crossover is performed, mutation can be done. It is used to avoid the process of getting trapped in a local optimum. Using mutation new offspring is randomly changed at any location. For binary encoding, in mutation, bits are randomly flipped i.e. at some position 1 flips to 0 and vice-versa. Mutation randomly changes the genetic information. When operated at bit level it is possible that mutation occurs at each bit but this has very lower probability as defined by mutation probability (P_m). It can be shown as:

Offspring1 before mutation	0101101001011011
Offspring1 after mutation with $P_m=25\%$	1001001001001011

On the basis of the encoding as well as the crossover, mutation is performed. For example in permutation encoding bits are exchanged. Mutation maintains genetic diversity and at the same time inhibits premature convergence. During evolution the mutation occurs according to the mutation probability, usually set to fairly low value

(0.01 is a good choice). If it is set to very high then the search will become a primitive random search.

Some of the types of mutation operator are as follows [20]:

- **Flip bit:** Simply selected bits are inverted. This mutation operator is used in Binary encoding scheme.
- **Boundary:** In this the randomly selected genes are replaced by the lower or upper bound for that gene.
- **Non uniform:** This mutation operator increases the probability that the amount of mutation will be close to 0 (zero), which is a good choice. This also keeps the population from stagnating in early stage of evolution and therefore gives the fine solution at later stage.
- **Uniform:** In this, the selected genes are replaced by the uniform random value chosen between user defined upper and lower bound for that genes.
- **Gaussian:** In this a unit Gaussian distributed random value is added to the chosen genes. If the newly generated gene falls outside the upper and lower bound for those selected genes, it is clipped.

All above mutation operators are used for integer or float genes except the Flip bit mutation operator.

2.3 GA Parameters

The basic parameters of GA are crossover and mutation probability [16].

- **Crossover Probability:** It shows how frequent the crossover occurs. If there is no crossover then children are the exact copy of their parents i.e. crossover probability is 0% whereas if crossover probability is 100% then all offspring are formed after crossover. Crossover is done so that new generation has better fitness value to survive. Crossover rate should be high about 80%-95% but sometimes it appears that 60% of crossover rate can also serve well.
- **Mutation Probability:** This shows how often the part of chromosome is mutated. If no mutation is there, the chromosome remains exact copy of their parent. If mutation is there part of the chromosome is mutated. If mutation is 100% entire

chromosome is changed, 0% means nothing is changed. Mutation rate should be very low, 0.5%- 1% is considered as best.

Other parameters of GA are as follows:

- **Population Size:** It shows that how many chromosomes are there in the search space. If the population size is very small then after performing certain operation we have very small search space. If the population size is very large then GA slows down. Good size of population is 20-30 whereas sometimes 50-100 gives better result. Research shows that appropriate population size depend on the encoding and the size of encoding string.
- **Encoding:** The type of encoding used is decided by the type of the problem and its instances.
- **Selection:** Select chromosomes for further operation. Generally Roulette wheel selection method is used but sometimes Rank selection can be better.
- **Crossover and Mutation Type:** This is decided by the type of encoding used.

2.3.1 Encoding

Encoding of chromosomes greatly depends on the problem. Some of the popular encoding schemes are presented below [16]:

- **Binary Encoding:** Most commonly used encoding techniques uses strings of 0 and 1.

Chromosomes1	1100100101100001
Chromosomes2	1000010110001011

For small number for alleles it gives a large number of chromosomes. This coding is not good as it faces many problems and sometime correction is needed after crossover and/or mutation. For e.g. Knapsack Problem uses Binary Encoding, where each bit of a chromosomes represent whether the particular thing is present in the knapsack or not.

- **Permutation Encoding:** When focus is on ordering in a problem then Permutation Encoding is used. For example in Travelling Salesman Problem ordering of cities is the key thing. In Permutation Encoding the fitness of

chromosomes is decided by the position of genes. In this encoding scheme some type of correction in crossover and/or mutation is needed to make the chromosomes consistent.

Chromosomes111	1 5 3 2 6 4 7 9 8
Chromosomes2	8 5 6 7 2 3 1 4 9

- **Value Encoding:** This Encoding is used where some complicated values such as real no. character or letter or words, is used in the problems. This can be used for developing some specific crossover and/or mutation depending on problem.

Chromosomes	1.2324 5.8243 0.4556 2.7293
Chromosomes	ABDJEIFJDHDIERHFNCJKNJW
Chromosomes	HI, WHAT DO YOU WANT?
Chromosomes	1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 1

An example of the use of Value Encoding method is for finding weights for neural network where real value in chromosomes represents weights in the neural network.

- **Tree Encoding:** This encoding is mainly used for evolving programs where the gene represents programming language commands, mathematical operations and other components of the program. Programming language LISP is used since in this the program can be easily parsed which makes the crossover and/or mutation relatively easier.

2.3.2 Selection

Chromosomes are selected from the population for producing children. Selection of chromosomes is done keeping in mind “survival of the fittest” i.e. the chromosomes having high fitness value are selected to produce new offspring with expectedly high fitness value. Selected chromosomes then undergoes crossover and/or mutation to produce offspring for new generation. This selection operation is governed by various methods; some of them are shown below [16], [23]:

- **Roulette Wheel Selection:** Chromosomes with higher fitness value is selected. In roulette wheel all population is placed according to their fitness and then a

chromosome is selected at random. Chromosomes having higher fitness value are selected quite often. In this case the other chromosomes will get very few chances to get selected. This method is also known as stochastic sampling with replacement.

- **Rank Selection:** In rank selection method first the population is ranked, then the fitness value is decided by the rank associated with each chromosomes. The worst chromosomes have fitness 1; second worst have fitness 2 and so on. The best chromosomes have fitness N. After this all chromosomes have a chance to get selected but this can lead to slower convergence since the best chromosomes have very little difference between them.
- **Steady state selection:** In this method the chromosomes with high fitness value are selected to produce offspring. These newly generated offspring replaces chromosomes having smaller fitness value to form a new generation. In this selection method the main focus is on surviving a large part of chromosomes for next generation.
- **Elitism:** In this method we preserve the best chromosomes i.e. chromosomes having high fitness are copied to the next generation, as they may lose during crossover and/or mutation. The rest is done in usual way. Since Elitism preserved the best chromosomes, this greatly increases the performance of GA.
- **Deterministic sampling:** It is very much similar to the roulette wheel selection method. In this selection method, the mean fitness is used instead of assigning probabilities proportional to the fitness value. In this the value obtained after dividing the individual fitness value by the average fitness is taken and only its part is considered for selecting the chromosomes for the next generation. The fraction part is used to sort the individuals. In the newly generated population the free places is filled by the chromosomes having higher fraction value in the sorted list.
- **Stochastic Remainder Sampling:** It is very much similar to Deterministic Sampling. In this case the population is generated by using the integer part of f_i/f (f_i is the individual fitness value and f is the mean fitness value). The free places in the new population are filled by using roulette wheel selection method. In

remainder selection mechanism, expected number of copies of a string is calculated as $m_i = f_i/f$. It assign parent deterministically from the integer part of each individuals scaled value (m_i) then roulette selection is used on the remaining fraction part. For ex- if the scaled value of an individual is 3.4 then that individual is listed thrice as a parent because the integral part is 3. The fractional part gives the probability of choosing the chromosomes as parent.

- **Linear Ranking selection:** In this method the individuals are ranked according to their fitness value. Individuals having higher fitness value have high rank and individuals with lower fitness have lower rank. Then the probability of selection of individuals is linearly dependent on their rank.
- **Truncate selection:** In this method the candidate solution are ordered by fitness and some proportion, p (for ex- $p= 1/3, 1/2$ etc.). This method is less sophisticated therefore it is not often used in practice.
- **Binary tournament selection:** In this method two individuals are chosen at random and better of them is selected with fixed probability $p, 0.5 < p < 1$.

2.3.3 Crossover and Mutation

Crossover and Mutation are two operators of Genetic Algorithm. The type of operator used in a particular problem is decided by the encoding scheme used and the problem itself. There are number of operator, few of them are discussed below based on the encoding scheme used.

2.3.3.1 Binary Encoding

When Binary Encoding techniques is used for encoding the chromosomes, then following operations can be performed on the available population of chromosomes, as the encoding schemes are the governing factor for the operators to be used [16], [20].

(a) Crossover

- **Single point crossover:** A crossover point is selected in chromosomes, from first chromosomes first part is taken and the rest is taken from the second chromosomes.

Chromosomes1	Chromosomes2	Offspring
10010 110	01010 011	10010011

- **Two point crossover:** In this kind of crossover, two crossover points are selected. Till first crossover point bits are copied from first chromosomes, from first to second crossover point bits are copied from second chromosomes again first chromosomes is used for copying the bits after the second chromosomes.

Chromosomes1	Chromosomes2	offspring
10 010 110	01 011 011	10 011 110

- **Uniform crossover:** It can be considered as the generalization case of single point and two point crossover. In this case bits are randomly copied from the two chromosomes to produce an offspring since each bit has an equal chance of being chosen from either parent. Uniform crossover generates a random value between 0 and 1 for each gene. If the value exceeds the locus crossover probability then only the genes are exchanged otherwise they are just copied from their parents.
- **Arithmetic Crossover:** In this case some arithmetic operation is performed to produce a new offspring.
- **Heuristic crossover:** This uses the fitness of the two parent chromosomes to determine the search direction. The offspring's are generated according to the following equation.

$$\text{Offspring1} = \text{Best parent} + r * (\text{Best parent} - \text{Worst parent});$$

where r is a number randomly chosen between 0 and 1.

$$\text{Offspring2} = \text{Best parent.}$$

(b) Mutation

In mutation selected bits are inverted. The bits that are highlighted are inverted to generate a new offspring. As shown in the following example.

Chromosomes	100101000110
Offspring after mutation	101001010110

2.3.3.2 Permutation Encoding

Permutation Encoding is used where ordering is the key factor, in this encoding scheme following operator can be used [16]:

(a) Crossover

Single point crossover: In this a crossover point is selected randomly, till this crossover point everything is copied in the offspring and for second part, the second chromosomes is scanned and value which is not there in the offspring is copied.

$$(5\ 4\ 3\ 2\ 1\ |6\ 7\ 8\ 9) + (4\ 5\ 3\ 6\ 8\ 9\ 7\ 2\ 1) = (5\ 4\ 3\ 2\ 1\ 6\ 8\ 9\ 7)$$

(b) Mutation

Bit exchange: When mutation is done in permutation encoding, selected two numbers are exchanged.

$$(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8) \rightleftharpoons (1\ 2\ 7\ 4\ 5\ 6\ 3\ 8)$$

2.3.3.3 Value Encoding

When some real numbers, words or characters etc are used for encoding a chromosome, following operators can be used.

(a) Crossover

All crossovers from Binary Encoding are used such as single point crossover, two point crossover, uniform crossover, arithmetic crossover etc.

(b) Mutation

A small number (for real value encoding) or value is added/subtracted to the selected value.

$$(1.29 \ 5.68 \ 4.32 \ 5.66 \ 8.98) \Longrightarrow (1.29 \ 5.63 \ 4.32 \ 5.66 \ 9.03)$$

2.3.3.4 Tree Encoding

Tree encoding is used where evolutionary programming comes into picture where the gene represents programming language commands, mathematical operations etc.

(a) Crossover

A crossover point is selected in both the parents, to create an offspring. The two parts below the crossover point are then exchanged.



Figure 2.2: Crossover in Tree Encoding

(b) Mutation

In Tree encoding the mutation can change the selected nodes.

2.4 Travelling Salesman Problem

Travelling Salesman problem is simple to describe but difficult to solve. TSP is a NP-hard problem which cannot be solved in polynomial time. In TSP, the salesman has to find a route in such a way that he visits each city exactly once and return back to the starting place keeping in mind that the cost should be as least as possible. Many algorithms have been designed to solve it. It is a suitable candidate to be solved using GA [17], [22].

2.4.1 Solution of TSP using Genetic Algorithm

Using GA for solving TSP gives the optimum solution. TSP can be classified on the basis of the structure i.e. whether symmetric or asymmetric method of representation is used. Symmetric if $C_{ij} = C_{ji}$ and asymmetric if $C_{ij} \neq C_{ji}$ for all i, j , where C_{ij} , C_{ji} represents the cost of path from i to j and vice-versa. For asymmetric representation of matrix there are $(n-1)!$ Combination can be there and for symmetric representation $(n-1)!/2$ possible solution can be there. Here we are using asymmetric matrix for representing the cost matrix and applying crossover and mutation till we didn't reach the optimal solution. A sequential constructive crossover operator is used and a crossover point is selected with common length, the information is just swapped after that crossover point. If already visited node appears again then replaces it with the unvisited node.

First generate the population or the search space in which all cities are connected to each other by a shortest possible edge, this can be done by using greedy approach. Now choose any two routes having minimum cost/distance, and combine them to create a child. These mutated children are then placed in the original population and replaces the longer routes in the population, since population size remain the same. The new children are then created until the desired goal is reached.

The accuracy of TSP is decided by the two factors- first is Population size and second is Speed. After comparing these two factors in each solution the next iteration is created by choosing the best one.

In TSP the crossover is the main operator for giving the solution since the characteristics are exchanged between the individuals of the population.

2.4.2 Algorithm

- Randomly create the population of individuals and arrange the cost of the path in the matrix form.
- Assign the fitness value to each chromosomes using the fitness function as shown below:

$F(x) = 1/f(x)$; where $f(x)$ is the objective function representing the total cost of the tour.

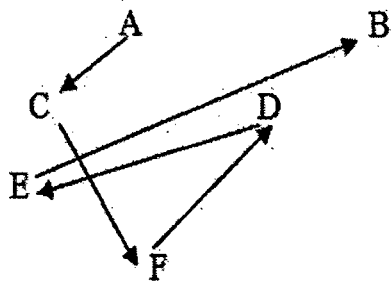
- Create the offspring by choosing the two parent chromosomes from the existing population by using the crossover operator.
- Mutate the chromosomes obtained after the crossover if needed, mainly used to avoid getting trapped in the local minima.
- After repeating above steps we get a new population having fitness value higher than the parent chromosomes.
- Stop when the desired constraint is met and then choose the chromosomes having highest fitness value and decode it to obtain the required solution.

Let us consider an example:

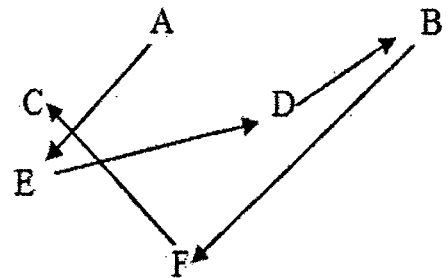
Distance	A	B	C	D	E	F
A	100	10	65	36	9	7
B	50	100	85	45	87	21
C	49	5	100	15	27	25
D	21	43	10	100	60	48
E	87	64	33	75	100	72
F	35	30	42	51	59	100

Table2.1: Showing Cost Matrix of TSP

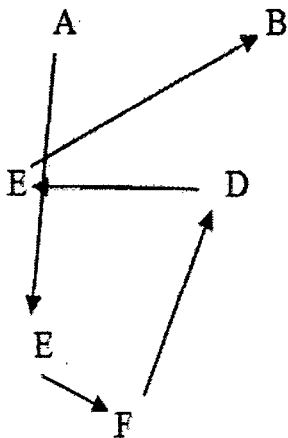
Select two parent chromosomes with value P1(A,C,F,D,E,B) and P2(A,E,D,B,F,C) with cost 265 and 190. Select first node from both the chromosomes. Since it is the same so go to next node i.e. C and E respectively. Now consider this C and E as the crossover point. Since cost (AC) = 65 and cost (AE) = 9 i.e. cost (AC) > cost (AE). So E is considered as the nearest node [28]



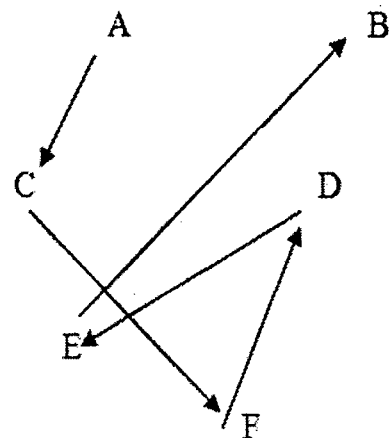
P1(A,C,F,D,E,B)



P2(A,E,D,B,F,C)

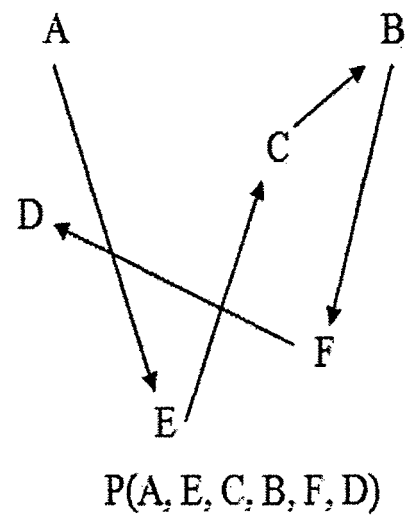
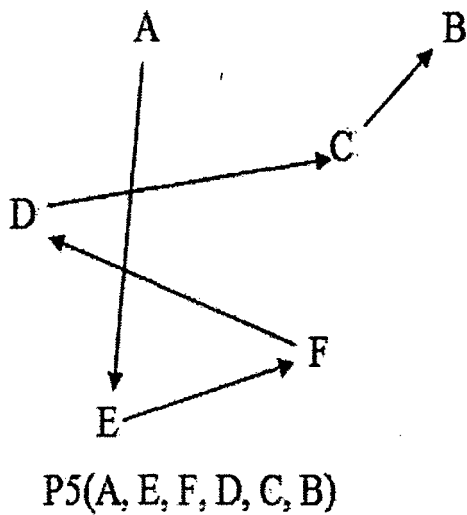


P3(A,E,F,D,E,B)



P4(A,E,C,B,F,D)

for both the chromosomes and the crossover point becomes after C and E in the respective chromosomes. Therefore, the new chromosomes created are now P3 (A,E,F,D,E,B) and P4 (A,C,F,D,E,B). But E is appearing twice in the first offspring so it is replaced by unvisited node C for that chromosome making it P5 (A, E, F, D, C, B) as the new offspring. This process is repeated in the similar fashion till we get a *better* solution. In this case the optimal solution comes out to be P (A, E, C, B, F, D). If the starting node is different then we can randomly select a crossover point and obtain an offspring after the crossover. Following this, we can reach an optimal solution.



2.4.3 Conclusion

Soft computing tools can be used in such situations when we have the appetite of accepting sub optimal solution owing to the fact that the search space is very large and the problem belongs to NP class. Genetic algorithm is used in a wide range of optimization problems like TSP, inductive learning, scheduling etc. The usefulness of GA in solving the problems to give optimized solution in efficient time makes it popular and preferred over traditional methods. The limiting conditions in the GA can be the fact that in some cases it can give better results but the same is not true in all the cases depending on how we are using the various operators of GA. Therefore, the performance observed with GA is highly dependent on the way it has been tuned.

Chapter 3

The Proposed Model

Grid computing refers to the heterogeneous resources that are distributed across the globe having multiple administrative domains to satisfy the user's requirement. In Grid the resources are needed to be used in optimized fashion inside an organization. It makes possible the distributed computation making scheduling of jobs in Grid is an important and challenging issue. Grid Scheduling is proved to be an NP hard problem implying exact solution to a problem cannot be obtained and sub-optimal solution becoming acceptable [14]. There are various soft computing approaches like Tabu search, Ant Colony optimization, Particle Swarm optimization and Genetic Algorithm (GA) etc [31] that can be used for the same. Mostly GA is preferred for such kind of highly complex problem, since based on natural selection and evolution process, it does not need to know the rules about the problem but uses its own method and give a sub-optimal solution closer to the exact one. This chapter focuses on Genetic Algorithm based scheduler for Computational Grid [35]. The strategy suggested in this chapter focuses on allocating the resources on a Computational Grid in order to minimize the turnaround time for the jobs submitted using Genetic Algorithm. Here, the job is divided in to sub-jobs. In scheduling various sub-jobs are mapped on the available grid resources considering the precedence of the job module as indicated by its directed acyclic graph (DAG). The model uses a centralized scheduler for scheduling the jobs using Genetic Algorithm. The chapter starts with the presentation of the scheduler, the data structures used and the notation considered for design of the model. Later, the scheduling algorithm is presented followed by the simulation results and their analysis.

3.1 Proposed Scheduling Strategy using GA

The proposed strategy uses the Genetic Algorithm so that the turnaround time of the jobs that are submitted to the Grid can be minimized. This can be achieved by changing the chromosomes structure over the generations by considering the chromosomes having the reduced turnaround time for the next generation. The model assumes a distributed and

multi-point entry system for the job in which they can be submitted to any node of the Grid, and is an extension of the work [28].

A Computational Grid consists of several hundred or thousand's of computational nodes that are distributed across the globe. Each node has certain attributes which characterize the node like their processing speed and the time allocated to a job or module on that particular node. The proposed model assumes that each node has a single processor. Following Figure 3.1 gives the overview about the Computational Grid which can be viewed as collection of many virtual organizations or cluster. Here, Virtual Organizations (VO) has a number of computational nodes. Within the Grid each VO, Cluster, Supercomputer or Individual PC's are connected by a high speed internet.

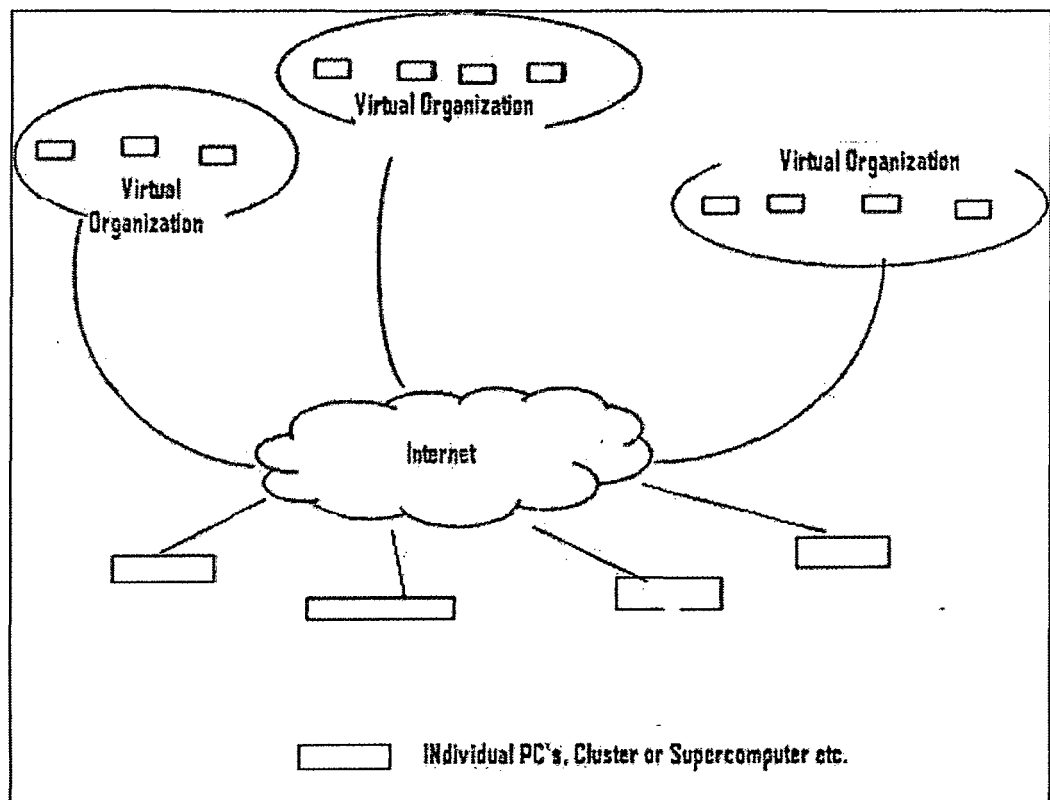


Figure 3.1: Computational Grid

The model permits each node to have its own scheduling policy since they can be managed by different individuals whether it is a cluster, supercomputer or individual PC's. The Grid Scheduler (GS) is loaded on all the nodes responsible for the maintenance, control and updating of their computational node and hence updating the

dynamic grid structure. The proposed scheduler uses distributed scheduling policy in which jobs can be submitted at any nodes and suitability of nodes for a particular job is evaluated by the dispatcher which is a part of GS loaded on each node which eventually dispatches the corresponding job modules to the suitable nodes. Thus the dispatcher is also responsible for scheduling of jobs. Since Grid is dynamic in nature so any number of jobs can arrive in the system sometimes forming a queue, so current state of the system is updated from time to time. For scheduling a job grid scheduler includes the need of computational nodes and the clock frequency to be stored in a table where it also stores the previous work load assigned on that particular node. Further, in the Grid, the arrangement is dynamic in nature so it needs to hold the current scenarios of the resources of the grid i.e. what kind of resources and how many are available at any instant of time and what is the status of the currently executing jobs. In short the GS hold information like:

- Processing speed of each node in the Grid.
- No. of resources/nodes available at any instant of time.
- Previously allocated workload on each node.
- Time to finish for a particular job.
- Nature of jobs i.e. specifying the type of cluster/resource requirement.

This information is eventually used in selecting the best resources for a particular job module. In the proposed model Genetic Algorithm is used to allocate/schedule the jobs to the grid. In this model each chromosomes holds the node number and the index of the array of chromosomes gives the job module that are executing on that particular node. In a chromosome the allocation of nodes for a particular job module is done under the consideration of the minimization of the TAT (Turnaround time). Therefore the arrangement of nodes in different chromosomes offers different execution cost. The chromosomes offering the minimum TAT is considered fittest for that generation. Using the GA operators like crossover and mutation, the model aims to further minimize the TAT over the generations. After meeting the stopping criteria we obtained an allocation pattern of nodes corresponding to the jobs i.e. chromosomes with least TAT obtained.

This allocation pattern defines the way in which the job module should be scheduled on the nodes in order to minimize the TAT for the jobs. Finally, allocates the resources to each job module according to the pattern suggested by the selected chromosome.

3.1.1. Data Structure used in the Model

The following data structures are used to meet the requirements:

- **g[no_module][no_module]:** This matrix represents sparse graph having values 0 and 1, where 1 means there is a connection between the nodes and 0 means there is no connection between the corresponding nodes.
- **IMC[no_module][no_module]:** It holds the inter-module communication cost between the different module in terms of bytes exchanged.
- **Chromosomes:** The value in a chromosome is a mapping of a node to a job module. This kind of node to job module mapping makes possible to form a large set of chromosomes known as population.
- **P[no_pop][size_chromosomes]:** This matrix is a population matrix having rows equal to the no. of population and column equal to the size of chromosomes .
- **p_TAT[no_pop][size_chromosomes+1]:** This matrix has rows equal to the no. of population and the no. of column is equal to the size of the chromosomes i.e. array size of chromosomes and one extra column is added to add the corresponding TAT value of each chromosomes. In this matrix, each row indicates the chromosomes and it's TAT.
- **D[no_resources][no_resources]:** This is an square matrix with size equal to the no. of resources available at a particular instant of time. It gives the distance traversed between the different nodes. The diagonal matrix has values equal to zeros, since there is zero distance traversed when going nowhere.
- **proc_speed:** It gives the processing speed of a job module on a particular node.

It can be calculated as:

$$\text{proc_speed} = \text{noins} / \text{cf}; \text{-----} (1)$$

Where noins is the no. of instruction and cf is the clock frequency of the node.

3.1.2. Notation used

The notation used in the proposed model is as follows:

no_module: No. of modules in the directed acyclic graph.

n: No. of edges in the directed acyclic graph.

n_x: No. of nodes in the cluster.

noins: No. of instruction in each module.

no_resources: No. of processor or nodes available in the grid system.

cf: Clock frequency of a node/processor.

new_population: This matrix gives the newly generated population obtained after operations like crossover and/or mutation.

RT_{jk}: Workload that is previously allocated to a particular node/processor.

no_generation: Variable tells how many generation the program is executing.

temp_pop: Temporary population i.e. population generated after selection is stored in this matrix.

child_pop: Population generated after crossover is stored in this matrix.

TAT_col: A column that holds the TAT obtained from the matrix having no. of column equal to size_chromosomes+1.

TAT_sorted_col: Sorted TAT_col (in ascending order) is stored in this column.

newgen_pop: The population that acts as a parent population for the next generation.

diff_chromosomes_same_TAT: This holds the population having different chromosomes but having same TAT.

pop_concat: Populations i.e. new population and the parent population are concatenated.

3.1.3. Fitness Function

The model considers the job module to be allocated on those nodes which offer minimum TAT. The cost of execution of job module 'm_i' on a node 'n_k' is given by NEC_{kifn}. The node execution cost depends on the three factors as follows:

- The processing speed 'E_{ijk}' of the module on a node,
- The inter-module communication 'IMC' in terms of bytes exchanged between modules 'm_i' and 'm_h' of the job allocated on node 'n_k' and 'n_l'. The nodes are separated by a distance 'D_{kl}', and
- The prior workload 'RT_{jk}' on node 'n_k'.

Considering these factors, the execution cost can be calculated as:

$$NEC_{kifn} = E_{ijkn} + \sum_{g=1}^{i-1} \max(B_{igj} \cdot D_{kr}) + RT_{jk} \quad \text{----- (2)}$$

Here E_{ij} gives the time taken by the node for executing the given job module. RT_{jk} is the resource characteristics (previously allocated workload) on which the job is executed. The second factor corresponds to the job characteristic which gives the communication cost between the two corresponding module as per its directed acyclic graph (DAG). The cost depends on the degree of interaction between a module with the preceding module of the same job. If there is no interaction then the communication cost remains zero. This NEC value is calculated for each node in a chromosome. The maximum value obtained in a chromosome gives the execution cost, which is also the TAT value offered by that chromosome as represented by equation (3).

$$TAT = \max(NEC_{kifn}) \quad \text{----- (3)}$$

For each chromosome in the population the TAT value is calculated. For a population of chromosomes the chromosome with minimum TAT value is considered as the best solution for that generation and the jobs are scheduled accordingly.

3.2. The Proposed Algorithm

The proposed algorithm schedules the jobs in the grid environment where both resources and jobs are dynamic in nature so that TAT (Turnaround time) can be minimized. The proposed strategy being GA (Genetic Algorithm) is based on using two

selection methods viz. Rank selection and Roulette wheel selection for selecting the chromosomes followed by performing the operations like crossover and mutation, if needed, on the population of chromosomes generated randomly. After performing the above mentioned operations the chromosome with minimum TAT value is obtained, which offers the way, scheduling is done so as to optimize the resources. The algorithm is described in detail as below.

- The grid is considered to be having many VO's or clusters.
- For any job submitted these clusters are evaluated for the TAT they offer to the jobs. The scheduler finally schedules the job on that cluster which offers the minimum TAT to the job.
- For each job submitted a population of chromosomes is generated.
- For the population of chromosomes generated NEC_{kifn} is calculated which gives the TAT for each chromosome of the population.
- Randomly select two chromosomes from the parent population depending on the strategy viz. rank selection or roulette wheel selection. If the selected chromosomes are same then repeat the selection process till chromosomes obtained remain the same. Crossover operation is performed next on the selected chromosomes to produce two child chromosomes. Here, crossover performed is single point crossover. These children are then stored in a new variable named `new_population`. If child chromosomes generated after crossover are already there in the `new_population` then do not store these child chromosomes but repeat the same process again. This gives a new population of child chromosomes. Now calculate the NEC_{kifn} and TAT of each chromosome. Next, pool the parent and child chromosomes together and out of them select the best chromosomes to form the next generation population having same no. of chromosomes as that of parent chromosomes.
- Perform mutation on the population if it satisfies the necessary condition. Mutation is performed after a fixed no. of generations. In this experiment the scheduling strategy is evaluated for mutation being performed after 5th, 10th and 15th generation respectively. Not all population is selected for mutation. The no.

of chromosomes being mutated is varied from 25%, 50% and 100% population respectively. This means for a generation when mutation is being performed every 5th generation at first 25% population is mutated and the mutated chromosomes replaces its parent chromosomes. Similarly, it is done using 50% and 100% of the population. If child generated after mutation is already there in the new population then repeat that step again until different chromosome is not obtained. This step is repeated for all generations NEC_{kifn} and TAT is repeatedly calculated for the new population.

- The population generated in this generation is used as parent population for the next generation and so on.
- Repeat the above steps for the no. of generations needed to optimize the obtained result, till the terminating condition is reached.

The algorithm for the same is given below:

```

Alloc(job)
{
  Submit the job in the form of modules
  Calculate the processing time of each job module on each resource for each cluster.
  Randomly generate a population of chromosomes.

  While (terminating condition is not met)
  {
    Calculate  $NEC_{kifn}$  and TAT value for each chromosome in the population set.
    Perform selection
    // Roulette wheel selection & rank selection alternatively.
    Perform crossover
    // Single point crossover.
    Calculate  $NEC_{kifn}$  and TAT value for each chromosome in the new population set.
    Perform mutation
    // Every 5th, 10th and 15th generation for 25%, 50% and 100% of population.
    // No. of genes mutated is equal to 20% of the size of the chromosomes.
    Calculate  $NEC_{kifn}$  and TAT value for each chromosome in the population set.
  }
  Record the best chromosome for each cluster.
  Allocate the job to the cluster offering the minimum TAT as per the pattern suggested by its best chromosome.
}

```

3.3. Illustrative Example

An illustrative example is given here so that the model can be easily understood. This shows the basic working of the model in terms of calculating the NEC_{kifm} and finally TAT for that chromosome for available grid resources. The same method is used to calculate NEC_{kifm} and finally TAT for other chromosomes as well. Here in this illustrative example parameter have been scaled down for better understanding of the working of the model. Though the simulation experiment have been performed with dynamic scheduling environment but for the purpose of illustration static grid environment is assumed. The illustration explains the analysis of the suitability of the jobs on one such cluster. The same method can be adopted to find the TAT offered to the job by other clusters.

Let us consider that initially the grid comprises of certain set of resources in various virtual organizations comprising it. The job submitted at any time at any node can be represented in the form of a DAG as shown below. A typical job for execution in the form of DAG is represented in Figure 3.1 with its attributes shown in Table 1.

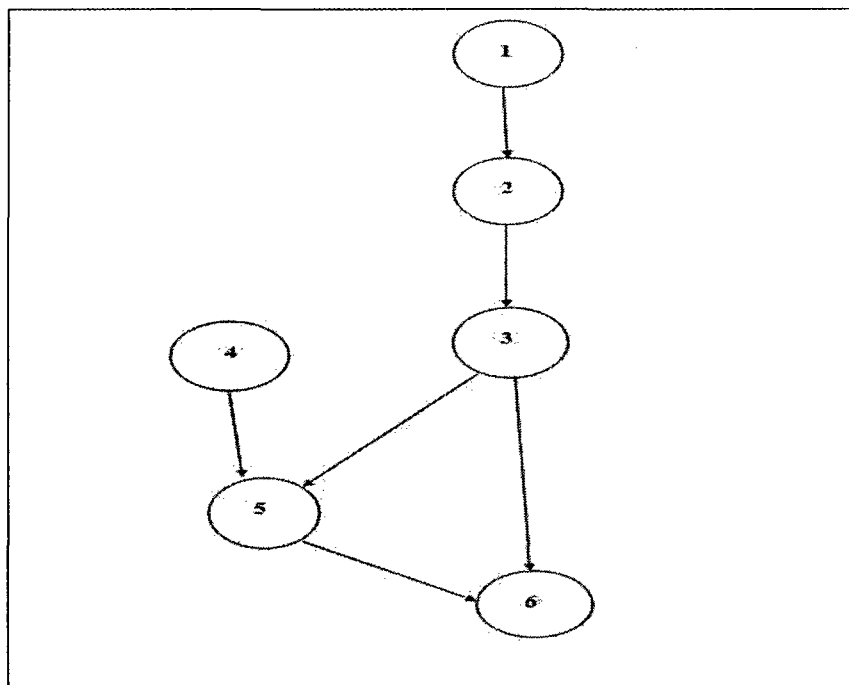


Figure 3.2: Directed Acyclic Graph for a Job

Job specialization	No of instructions(noins)
j_0	481
j_0	173
j_0	499
j_0	325
j_0	132
j_0	237

Table 3.1: Job j_0

Further, each resource available in the grid also has certain characteristics like clock frequency and the previous workload. This information can be shown in the matrix below as Table 2.

Node no.	Clock frequency(cf)	Previous workload(RT)
P_1	19	3
P_2	16	3
P_3	15	3
P_4	10	2
P_5	11	5

Table 3.2: Node Attributes in a Cluster

Table 3 shows the hamming distance between the different processor. This actually refers to the number of link traversed between two corresponding nodes, if any data is exchanged between them.

	P_1	P_2	P_3	P_4	P_5
P_1	0	2	2	5	5

P_2	2	0	4	4	2
P_3	2	4	0	3	2
P_4	3	3	3	0	5
P_5	5	4	3	2	0

Table 3.3: Hamming Distance (D) between different Nodes

The inter module communication (IMC) cost between the different job module in terms of bytes exchanged can be represented in the form of matrix given below as Table 4. Since the grid comprises of various virtual organizations called clusters the job can be evaluated for the TAT offered by the cluster to eventually submit it on the cluster offering minimum TAT. The illustration explains the analysis of the suitability of the jobs on one such cluster. The same method can be adopted to find the TAT offered to the job by other clusters.

	M_1	M_2	M_3	M_4	M_5	M_6
M_1	0	4	0	0	0	0
M_2	0	0	4	0	0	0
M_3	0	0	0	0	5	4
M_4	0	0	0	0	4	0
M_5	0	0	0	0	0	0
M_6	0	0	0	0	0	0

Table 3.4: IMC Matrix

Table 5 summarizes the processing time of a processor for various job modules calculated by using equation (1).

	M ₁	M ₂	M ₃	M ₄	M ₅	M ₆
P ₁	25.31	9.10	26.26	17.10	6.94	12.47
P ₂	30.06	10.81	31.18	20.31	8.25	14.81
P ₃	32.06	11.53	33.26	21.66	8.80	15.80
P ₄	48.10	17.30	49.90	32.50	13.20	23.70
P ₅	43.72	15.72	45.36	29.54	12.00	21.54

Table 3.5: Processing Time of Various Modules on Various Processors

Typical chromosomes may look like:

1	4	1	2	3	2
---	---	---	---	---	---

Table 3.6: A Typical Chromosome

This means that module 1 is allocated to processor 1, module 2 is allocated to processor 4, module 3 is allocated to processor 1 and so on.

Since module m_1 is allocated on processor 1 the node execution cost NEC_{kifn} on the processor 1 is can be calculated as shown below:

Module 1	Processor 1
E_{ij11}	25.31
$\sum \max(B_{igj} * D_{kr})$	0
RT_{jk}	3
NEC_{11fn}	28.31

Table 3.7: Calculated Values1

From the Table 5 it can be seen that the processing speed E_{0j10} for module m_1 on node/processor 1 comes out to be 25.31. As this node does not have any predecessor so $\sum \max(B_{igj} * D_{kr})$ has value equals to zero. This node has some previous workload assigned to it, has value equals to 3 as seen from the Table 2. Using all these values the NEC_{10n0} can be calculated as 28.31 this value then becomes the RT_{jk} for processor 1. Similarly, the

NEC_{kifn} values for module 2, module 3 and module 4 can be calculated and that come out to be 65.61, 91.87 and 23.31 respectively. For e.g. the NEC_{kifn} values calculated for module 5 can be calculated as shown below:

Module 5	Processor 3
E_{sj35}	8.80
$\sum \max(B_{ij} * D_{kr})$	16.00
RT_{jk}	91.87
NEC_{35f5}	126.67

Table 3.8: Calculated Values2

Since module 5 is allocated on processor 3, for this arrangement the processing speed i.e. E_{sj35} is 8.80 as seen from the Table 5. It can be seen from the DAG that module 5 have two predecessor module 3 and 4 with whom it needs to communicate. This communication cost for module 5, can be calculated as

$$\sum \max (B_{ij} * D_{kr}) = \max(B_{45j} * D_{23}, B_{35j} * D_{13}) = \max(4 * 4, 5 * 2) = 16.$$

Similarly, for module 6 the NEC_{kifn} values can be calculated and it comes out to be 149.48. The maximum value of NEC_{kifn} gives the TAT value for that chromosome. In this case, the TAT comes out to be 149.48. Similarly, we calculate the NEC_{kifn} and finally TAT values for all other chromosomes of the population and then performs other operation as per requirement. Using GA we try to evolve to a population comprising of improved chromosomes offering lesser TAT than its predecessors. This process is repeated till termination condition is not met

3.4. Simulation Experiment

Simulation experiments were conducted to observe the allocation of the jobs on the grid. The experiment is conducted on Intel Core-2 Duo @1.97GHz using MATLAB 7.6.0 (R2008a). The data values taken in the experiment are generated dynamically during execution.

S.No.	Parameter	Notation Used	Range
1	No. of Resources/Processor	no_resources	5-20
2	Clock frequency of processor	cf	10-20
3	Distance between two processors	D_{kr}	2-5
4	Time to finish previous workload	RT_{jk}	2-5
5	No. of Module in a job	no_module	5-20
6	No. of instructions in a module	noins	100-500
7	Inter module communication between	IMC	2-5
8	Population Size	no_pop	100-200
9	Size of chromosome	size_chromosomes	5-20
10	No. of generation	no_generation	100,200
11	Crossover considered during experiment	Single point	--
12	Mutation performed after no. of generations	mu	5,15,20
13	% of Population selected for Mutation	--	25%,50%,100%
14	Rank selection method	rank_selection	--
15	Roulette wheel selection method	rw_selection	--

Table 3.9: Parameters Used

The experiment is performed for 200 generations for getting better results but the results seems to be converging mostly after 100 generations. The experiment is performed for size of the population varying from 100- 200 in order to observe the behavior if the model. Also, the model is analyzed for variation in the generation no. after which mutation is performed in order to know the suitable value for optimized results.

The variation of the turnaround time over various no. of generations with the population size of 100-200 for roulette wheel selection scheme and rank selection scheme for different no. and percentage of population on which mutation is performed are discussed and shown below:

RS 25: It represents the graph when the percentage of population selected for mutation is 25% and rank selection method is used for selection operation.

RS 50: It represents the graph when the percentage of population selected for mutation is 50% and rank selection method is used for selection operation.

RS 100: It represents the graph when the percentage of population selected for mutation is 100% and rank selection method is used for selection operation.

RW 25: It represents the graph when the percentage of population selected for mutation is 25% and roulette wheel selection method is used for selection operation.

RW 50: It represents the graph when the percentage of population selected for mutation is 50% and roulette wheel selection method is used for selection operation.

RW 100: It represents the graph when the percentage of population selected for mutation is 100% and roulette wheel selection method is used for selection operation.

Figure 3.2 - 3.4 are plotted when population size is taken as 100 and figure 3.5 - 3.7 are plotted when population size is 200.

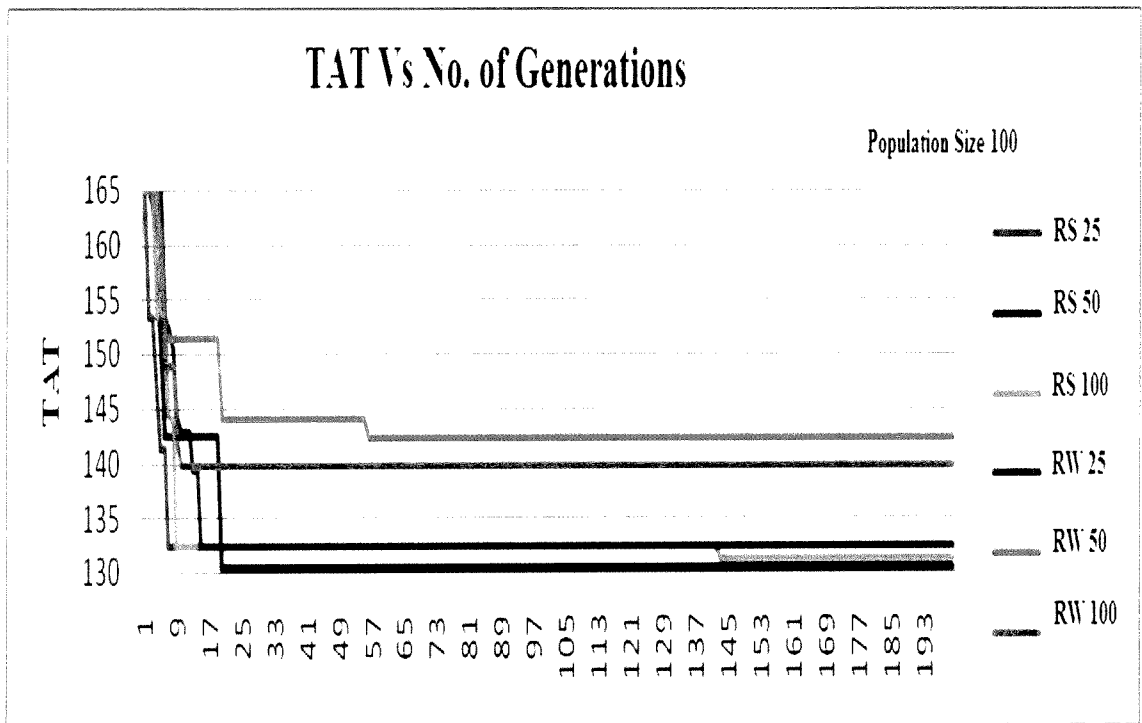


Figure 3.3: Mutation after 5th Generation

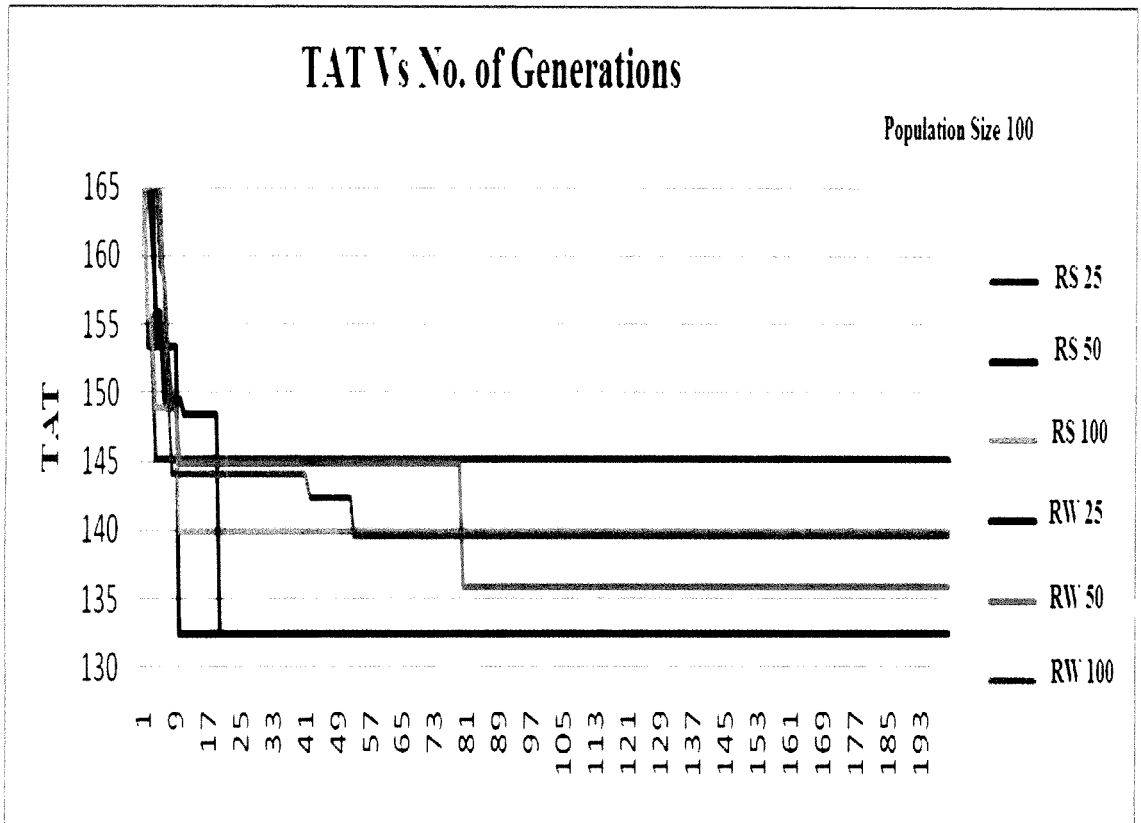


Figure 3.4: Mutation after 10th Generation

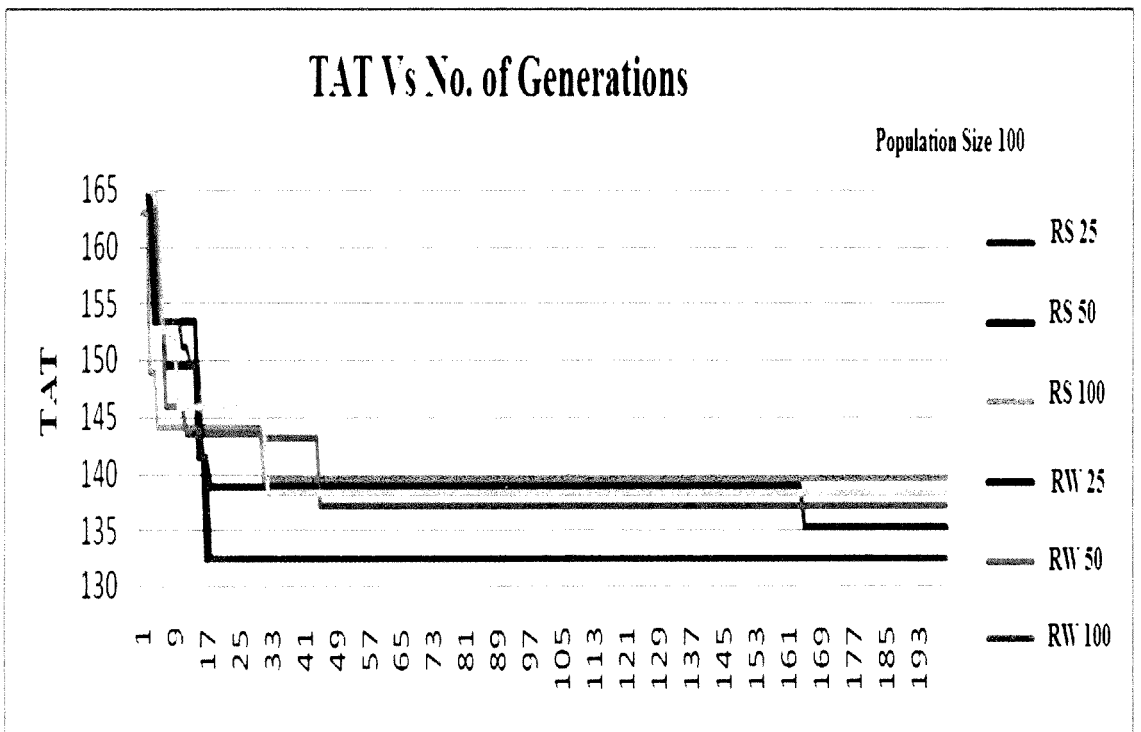


Figure 3.5: Mutation after 15th Generation

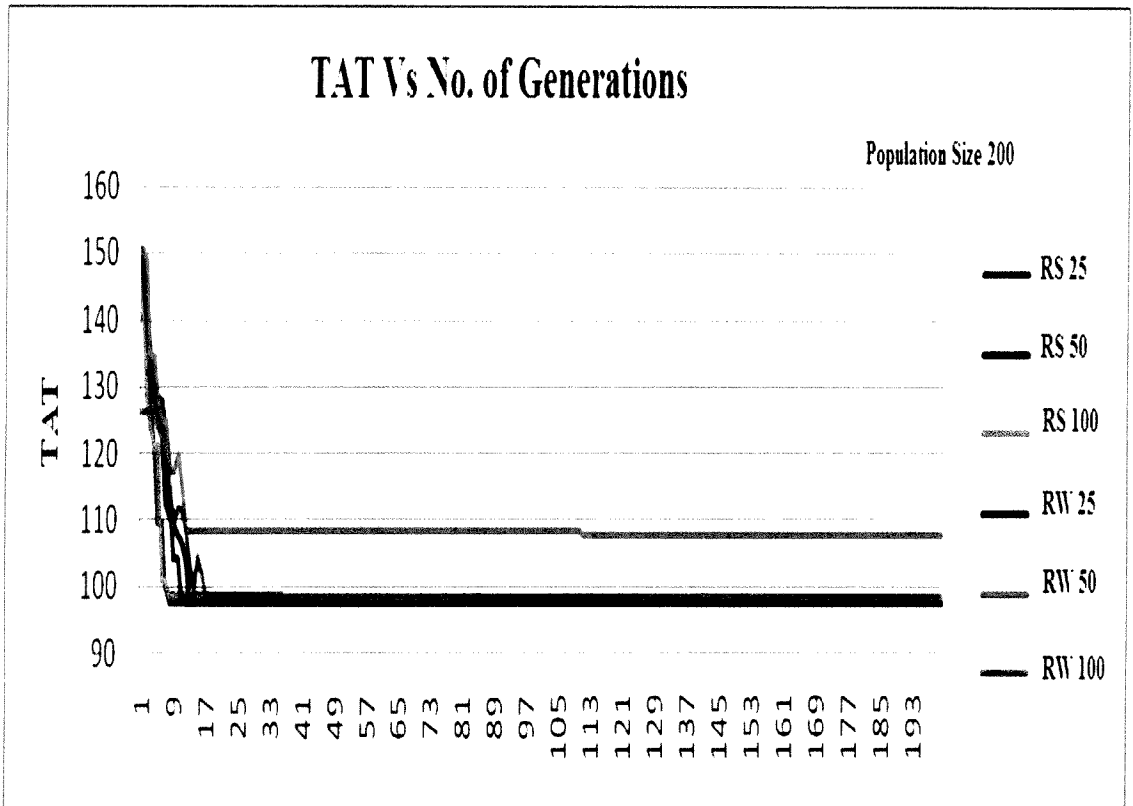


Figure 3.6: Mutation after 5th Generation

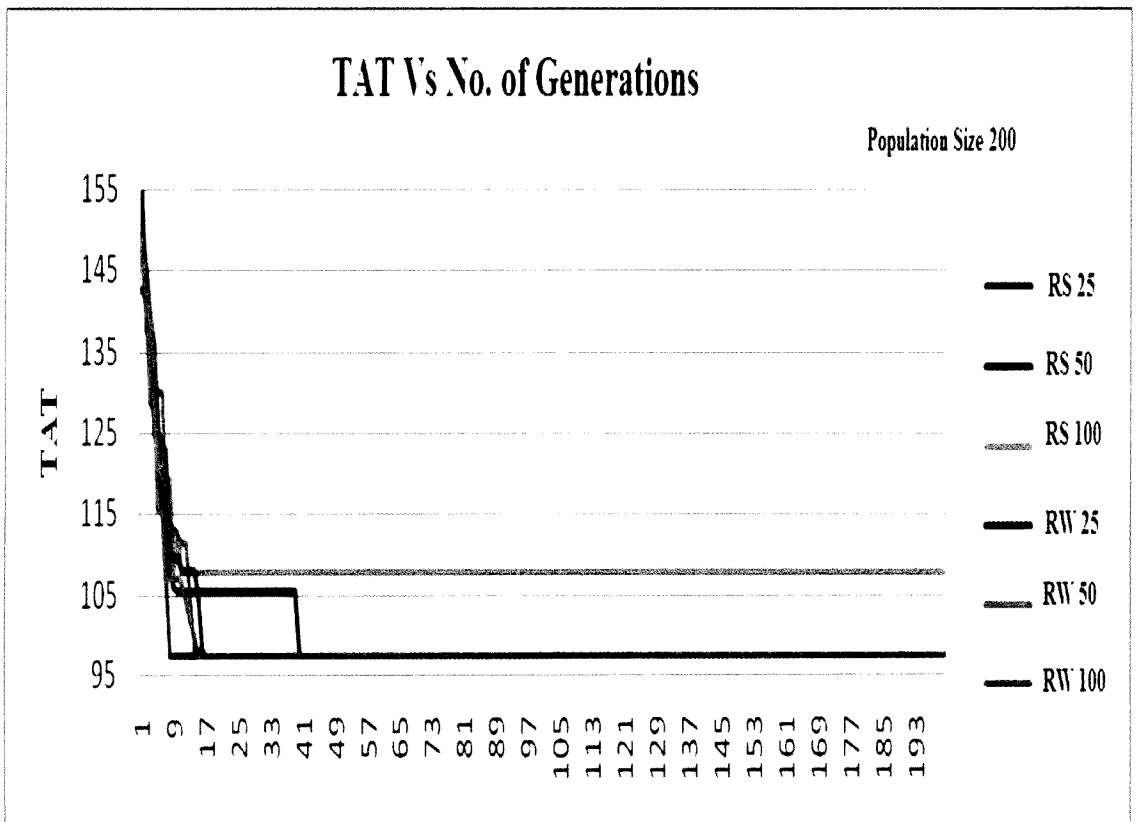


Figure 3.7: Mutation after 10th Generation

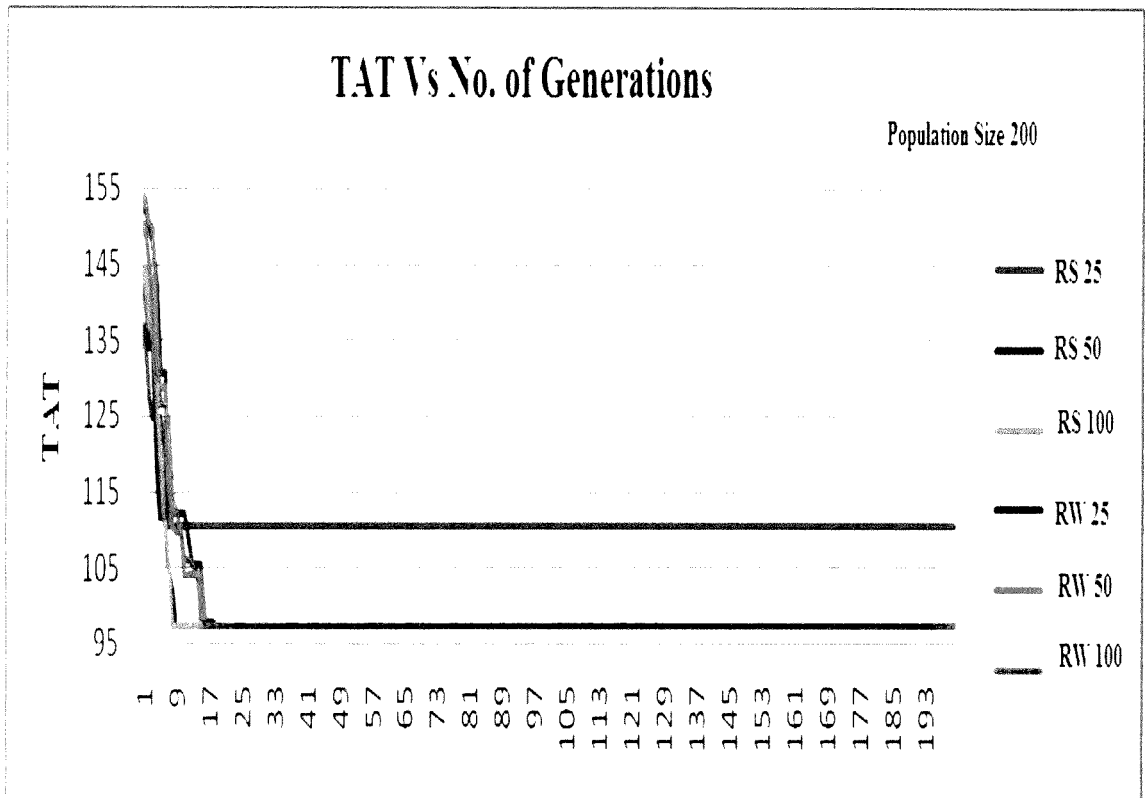


Figure 3.8: Mutation after 15th Generation

3.5. Observations

- From the graph it has been observed that in mostly all the cases rank selection with case RS 50 shows better performance at both the population size of 100 and 200, as we move to the larger population size of 200, the performance of roulette wheel selection increases coming at par with RS 50 to give better results.
- From the experiment it has been observed that the TAT value decreases over generations and an optimum value is obtained owing to the reason that the better resources i.e. resources with higher clock speed and lesser previous workload are selected for executing the job.
- When the population size is taken as 100 and the no. of generation after which mutation is performed is taken as 5 then rank selection methods perform better at all values with the percentage of population selected for mutation varying as 25%, 50% and 100% referred as case RS 25, RS 50 and RS 100 respectively. Only in this case roulette wheel perform at par or better when the percentage of

population selected as 25% for case RW 25 with even faster convergence over rank selection.

- There are three factors which affect the allocation of resources are the processing speed of the node, time to finish previous allocated workload and IMC. From the graph it can be observed that the lower TAT (turnaround time) is obtained when allocation is made on nodes having lesser previous workload and having high clock frequencies.
- In general when the population size is 100 and the no. of generation after which mutation is performed is either 5, 10 or 15, rank selection method with case RS 50 gives better result than roulette wheel selection with a faster convergence over roulette wheel selection.
- When the size of the population increases from 100 to 200 roulette wheel selection shows improved performance in all cases (i.e. when mutation performed after 5th, 10th, 15th generation and in each case the percentage of population selected is 25%, 50% and 100% respectively) in comparison to rank selection matching RS 50 case. Further, roulette wheel selection exhibits a little slower convergence as compared to rank selection method.
- Overall it can be concluded that rank selection gives better performance than roulette wheel selection specially when the percentage of population mutated is 50% for any no. of generation after which mutation is performed i.e. 5th, 10th or 15th, whereas roulette wheel selection gives better result when the population size is 200 with the best result exhibited in RW 100 case.

Chapter 4

Conclusion and Future Scope

Grid system is a dynamic environment in which resources can leave or join the system at any time. This behavior makes the scheduling a tough job. Job scheduling in grid system is an NP-Hard combinatorial optimization problem. Therefore, traditional methods are not suitable for such kind of complex problems as they have certain constraints such as they do not exploit the tolerance for imprecision and may not give the solution within the time constraints. Therefore, soft computing techniques like genetic algorithm, fuzzy logic and artificial neural network finds a wise use for such type of problem.

The proposed model uses genetic algorithm (GA) for scheduling a job on the grid system so that the turnaround time of the jobs that need to be executing on the system can be minimized. Here, the study is based on two selection methods viz. rank selection and roulette wheel selection method for a single point crossover and mutation effectuated after certain no. of generations for certain percentage of the population. From the obtained results it can be said that the allocation is made keeping in mind that the jobs are scheduled on resources having higher clock frequencies and lesser previous workload with a minimum communication cost. Accordingly, the jobs are assigned on the resources on the grid system so that turnaround time can be minimized.

From the simulation results it can be observed that rank selection shows better performance when the percentage of population mutated is 50% for any no. of generation after which mutation is performed i.e. 5th, 10th or 15th, whereas roulette wheel selection gives better results when the population size is 200 with mostly 100% of the population being mutated.

Since scheduling is an NP-hard problem, the results obtained cannot be treated as the most optimized results. This gives us option to use some other soft computing approaches for the same. Multiobjective function or parameter can be explored by using genetic

algorithm thereby optimizing more than one parameter. Further, some other selection methods and other crossover operator can also be tested for the performance evaluation to the model. Some other optimization methods like particle swarm optimization or ant colony optimization method etc. can be explored for a comparative study of the work.

References:

1. Parashar, M. and Lee Craig A., "Grid Computing: Introduction and Overview". Proceeding of the IEEE, Special Issue on Grid Computing Seattle, WA: IEEE Press.
2. Foster, I., "What is the Grid? A Three Point Checklist", Argonne National Laboratory & University of Chicago. July 20, 2002. <http://dlib.cs.odu.edu/WhatIsTheGrid.pdf>
3. A. Roxburgh, K. Pawlikowski and D. McNickle, "Grid Computing: The Current State and Future Trends". TR-COSC 01/04, 2004. <http://hdl.handle.net/10092/3060>
4. Dabas, P. and Arya, A., "Grid Computing: An Introduction", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, 2013.
5. http://www.dbaoracle.com/real_application_clusters_rac_grid/types_of_grid_computing_oracle.htm
6. www.gridcafe.org
7. http://www.dba-oracle.com/real_application_clusters_rac_grid/grid_vs_clusters.htm
8. <http://www.cloudways.com/blog/cloud-computing-vs-grid-computing-differentiated>
9. Dong, F. and Akl, Selim G. "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", School of Computing, Queen's University, Kingston, Ontario, Jan-2006.
10. Malathi, G. and Sarumathi, S., "Survey on Grid Scheduling", Journal of Computer Application, Vol-III, No.3, July-Sept 2010.
11. Elzeki, O. M., Rashad, M. Z., Elsoud, M. A., "Overview of Scheduling Tasks in Distributed Computing Systems", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-3, July 2012.
12. http://drona.csa.iisc.ernet.in/~gsat/Course/DAA/lecture_notes/jeff_nphard.pdf
13. <http://www.cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture10-NPcomplete.pdf>

14. Wigderson, A., "P, NP and Mathematics – a Computational Complexity Perspective", December 21, 2006.
15. <http://en.wikipedia.org/wiki/NP-hard>
16. <http://www.obitko.com/tutorials/genetic-algorithms/index.php>
17. Dwiedi, V., Chauhan, T., Saxena, S. and Agrawal, P. "Travelling Salesman Problem using Genetic Algorithm". DRISTI 2012.
<http://research.ijcaonline.org/dristi/number1/dristi1007.pdf>
18. <http://www.soft-computing.de/def.html>
19. http://www.myreaders.info/html/soft_computing.html
20. <http://www.nd.com/products/genetic.htm>
21. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html
22. <http://www.hermetic.ch/misc/ts3/ts3demo.htm>
23. Shivraj, R. and Ravichandran T. "A Review of Selection Method in Genetic Algorithm". International Journal of Engineering Science and Technology, Vol. 3, No. 5, pp. 3792–3797, 2011
24. Pizzuti, C., "A Multiobjective Genetic Algorithm to find Communities in Complex Networks," IEEE Transactions on Evolutionary Computation, Vol. 16, No. 3, June 2012.
25. Mathew, Tom V., "Genetic Algorithm". http://www.civil.iitb.ac.in/tvm/2701_dga/2701-ga-notes/gadoc.pdf
26. <http://www.scribd.com/doc/46961967/25/Advantages-and-Disadvantages-of-Genetic-Algorithms>.
27. <http://www.geneticprogramming.com/Tutorial>
28. Raza, Z., Vidhyarthi, D. P., "GA based Scheduling Model for Computational Grid to Minimize Turnaround Time", International Journal of Grid & High Performance Computing, Volume 1, Issue IV, 2009, pp 70-90.

29. Xhafa, F. and Abraham, A., "Computational Models and Heuristic Methods for Grid Scheduling Problem", *Future Generation Computer Systems*-2009.
30. George, D. I., Muthulakshmi, P., "An Overview of the Scheduling Policies and Algorithm in Grid Computing", *International Journal of Research and Reviews in Computer Science*, Volume 2, No. 2,, April 2011.
31. Jiang, C., Wang, C., Liu, X. and Zhao. Y., "A Survey of Job Scheduling in Grids", *LNCS 4505*, pp. 419-427, 2007.
32. Foster, I., Zhao, Y., Raicu, I. and Lu, S., "Cloud Computing and Grid Computing 360-Degree Compared". *Proc. IEEE Grid Computing Environments Workshop*, IEEE Press, 2008, pp. 1-10.
33. Zhang, S., Chen, X., Zhang, S., Huo, X., "The comparison between cloud computing and grid computing", *International Conference on Computer Application and System Modeling (ICCASM)*, 2010, vol.11, 22-24, Oct. 2010, p.V11-72-V11-75.
34. Gandotra, I., Abrol, P., Gupta, P., Uppal, R. and Singh, S., "Cloud Computing over Cluster, Grid Computing: a Comparative Analysis", *Journal of Grid and Distributed Computing*, Volume 1, Issue 1, pp. 01-04, 2011.
35. Aggrawal, M., Kent, R. D. and Ngom, A., "Genetic Algorithm Based Scheduler for Computational Grids", *Proceedings of the 19th Symposium on High Performance Computing Systems and Applications*, pp. 209-215, 2005.
36. Vidyarthi, D. P., Sarker, B. K., Tripathi, A. K. and Yang, L.T., "Scheduling in Distributed Computing Systems- Analysis, Design and Models", Springer 2009.