Global Journal of Computer Science and Technology
Cloud and Distributed
Volume 13 Issue 2 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Maximizing Computational Profit in Grid Resource Allocation using Dynamic Algorithm

By K. Sathish & A. Rama Mohan Reddy
Sri Venkateswara University College of Engineering, India

Abstract - Grid computing, one of the most trendy phrase used in IT, is emerging vastly distributed computational paradigm. A computational grid provides a collaborative environment of the hefty number of resources capable to do high computing performance to reach the common goal. Grid computing can be called as super virtual computer, it ensemble large scale geographically distributed hetero- geneous resources. Resource allocation is a key element in the grid computing and grid resource may leave at anytime from grid environment. Despite a number of benefits in grid computing, still resource allocation is a challenging task in the grid. This work investigates to maximize the profits by analyzing how the tasks are allocated to grid resources effectively according to quality of service parameter and gratifying user requisition. A fusion of SS-GA algorithm has introduced to answer the above raised question about the resource allocation problem based on grid user requisition. The swift uses genetic algorithms heuristic functions and makes an effective resource allocation process in grid environment. The result of proposed fusion of SS-GA algorithm ameliorates the grid resource allocation.

Keywords : *grid computing, heterogeneous resources, resource allocation, QoS parameter, swift scheduler (SS), genetic algorithm (GA), fusion of SS-GA.*

GJCST-B Classification : *C.1.4*

Strictly as per the compliance and regulations of:

Maximizing Computational Profit in Grid Resource Allocation using Dynamic Algorithm

K. Sathish [α] & A. Rama Mohan Reddy [σ]

Abstract - Grid computing, one of the most trendy phrase used in IT, is emerging vastly distributed computational paradigm. A computational grid provides a collaborative environment of the hefty number of resources capable to do high computing performance to reach the common goal. Grid computing can be called as super virtual computer, it ensemble large scale geographically distributed hetero-geneous resources. Resource allocation is a key element in the grid computing and grid resource may leave at anytime from grid environment. Despite a number of benefits in grid computing, still resource allocation is a challenging task in the grid. This work investigates to maximize the profits by analyzing how the tasks are allocated to grid resources effectively according to quality of service parameter and gratifying user requisition. A fusion of SS-GA algorithm has introduced to answer the above raised question about the resource allocation problem based on grid user requisition. The swift uses genetic algorithms heuristic functions and makes an effective resource allocation process in grid environment. The result of proposed fusion of SS-GA algorithm ameliorates the grid resource allocation.

Keywords : grid computing, heterogeneous resources, resource allocation, QoS parameter, swift scheduler (SS), genetic algorithm (GA), fusion of SS-GA.

I. Introduction

The grid is the term defines based on the infrastructure of the distributed computing and offer the resources based on the client requirement. Grid technology can largely improve the virtual society's effectiveness and usefulness and supports the productivity. When going with improvement in grid technology, the grid facing many challenges by shared networking and collaboration and the main challenges by resource optimization and processes optimization. Grid computing technology is coordinated with the use of several numbers of servers; the grid server toil based on many applied techniques and methods. And these specialized gird servers in the network which acts together as a single logic integrated server system [1]. In 1969, Leonard Kleinrock was first visualized the concept of Grid in computing, he wrote: "We will probably see the spread of computer utilities, which, like present electric and telephone utilities, will service individual homes and offices across the country". In 1998, the redefinition of grid computing is evolved by Carl Kesselman and Ian Foster they wrote: "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [2]. The large-scale computation of the grid computing is the collection of heterogeneous autonomous system; systems are distributed throughout geographically, and the large number of heterogeneous networks is interconnected in the grid [3]. Fewer years back grid computing is formally defined as a technology that allows accessing, managing and strengthening the resources of IT in the environment of distributed computing. Grid computing is an advanced technology of distributed computing, that brings all databases, servers, infrastructures, applications and resources into a massive single system. The grid technology partnership among different enterprise organizations comprises the same organization and in addition external enterprise companies [1]. There are three reasons, which ensure grid computing is a promising technology [4]:

The availability of the number of efficient computer resources which brings time-consuming and cost-effective to all clients, Grid computing can solve what normal system doesn't have the capability to solve some problem that can be solved by the available cooperative resources with massive computing power in the grid and Grid system directs the job to the proper resources of numerous computers can be run cooperatively and it works towards common goal with the usefulness of available resource collaboration which results the less time consumption and cost effective.

In grid computing, there are lots of computers connected to grid to execute the jobs assigned by the clients, and among available computers at least a computer will perform as a server, this server takes all responsibility to allocate the client's jobs to the appropriate resource that are ready to execute [5]. Generally, grid resources can be divided into two types; one is software resources, and another one is hardware resources. In the category of the software resources which includes source of application pack, component services and data resources and the hardware resources which includes network resources, storage resources and computational resources [7]. The resources are distributed as grid geographically, unlike management policies are applied, heterogeneous

Author α : Research scholar, Srivenkateswara University, Tirupati chittoor (dt). E-mail : ksathishphd@gmail.com
Author σ : Professor, Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering, Tirupati.

resources are interconnected and all heterogeneous resources belong to various administrative domains. In grid computing, the term resource is defined as the capability that can be shared and utilized heterogeneous networked grid environment [6].

The pros of grid computing are

- Grid performance in processing data integration.
- Gird takes less time to solve more complex problems.
- Process the huge date sets into smaller one for faster execution.
- Put to good use the available heterogeneous hardware in the grid.
- Collaboration between the different organizations becomes easier.
- The cons of grid computing are
- Grid standards and software are still evolving one.
- The job submissions in grid network are non-interactive.
- Grid resource administrations are not properly controlled.
- No proper allocation of jobs to the appropriate resources.
- Due to the improper resource allocation leads to long execution time and added cost.

In grid computing, large scale powerful resource allocation is a main challenge in a grid that may be critical to task performance. In general, the resource allocation in the grid faces many challenges in adaptability, load balancing, scalability and reliability [7]. The resources in the grid network are not controlled centrally and the resources can enter grid network and may leave anytime from the network autonomously. The autonomous property of the grid resources in the network leads to vagueness. The competence of the grid system is totally dependent on the proficiency of the resource allocation. As per the grid network, there is a substantial change in availability of the resource which varies the computational performance of the network and so there is a need for scheduling and allocation algorithm to survive from the changing environment for this network. To gear up the grid resource allocation, there is a need to overcome the challenges to speed up the processing power and resource memory in order to process the job in minimal computational cost as well as minimal computational time [8]. The First Come First Serve (FCFS) based resource allocation [19] that allocates the jobs which comes first. The other jobs in the job pool may wait longer due to the job size and resource availability for that particular size of job, and so this leads to very high time and cost consumption. The Shortest Job First (SJF) also named as Shortest Job Next (SJN) or Shortest Process Next (SPN) based resource allocation [19] that allocates the job which has shorter length on the fastest resource. When the continuous arrival of shorter length job pushes the job with the longest length leads to longer waiting time in the job pool.

The Swift Scheduler (SS) based resource allocations [18] collects the job from the different users and swift the job using either by length or priority assigned. The resources are allocated based on the job swift by their priority in the job pool. The Genetic Algorithm (GA) based resource allocation is a population-based technique that allocates the jobs according to the fitness function evolved. The Genetic Algorithm (GA) performs crossover and mutation each in order to find the optimal solution. The Ant Colony Optimization (ACO) based resource allocation is a population-based technique which found the optimization problem solution using the pheromone values. Initially, the jobs are assigned to the resources randomly and each job verifies all the available resource using probability rule by assigning initial pheromone value and visibility value (i.e. simple heuristic value finds dividing one by distance between the job and the resource). After analyzing each job with each resource using probability rule, the process finds the reasonable solutions and stores values in the pheromone trails. The small amounts of pheromone trials are evaporated evaporation constant (ρ) and density of pheromone ($\Delta\tau_{ij}$). Finally the jobs are allocated to the resources which have the strong pheromone value and which found the optimal resource with less time consumption and low cost.

In this paper, the above all algorithms are compared with proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation as shown in fig.1. The Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation algorithm fuses prioritized jobs in the job pool using Swift Scheduler (SS), with the Genetic Algorithm (GA) based resource allocation, so it makes full use of advantages of the Swift Scheduler, estimated with a Genetic Algorithm, as well as ability provided the effective resource allocation. The swift scheduler arranges/schedules the jobs by the user's urgency (i.e. assigned as priority) or length of the job. And the arranged/scheduled jobs are allocated to the appropriate resources using Genetic Algorithm to obtain time-consuming and cost-effective as best. The Quality of Service (QoS) [14] parameter includes job execution time and cost efficiency are compared in First Come First Serve (FCFS) based resource allocation, Shortest Job First (SJF) based resource allocation, Swift Scheduler (SS) based resource allocation, Ant Colony Optimization (ACO) based resource allocation and AWOPF & AWPF Constraint checked Genetic Algorithm (GA) based Resource Allocation with the proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation algorithm. The performance

of the proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation algorithm compared with the above mentioned other algorithms by using simulation.

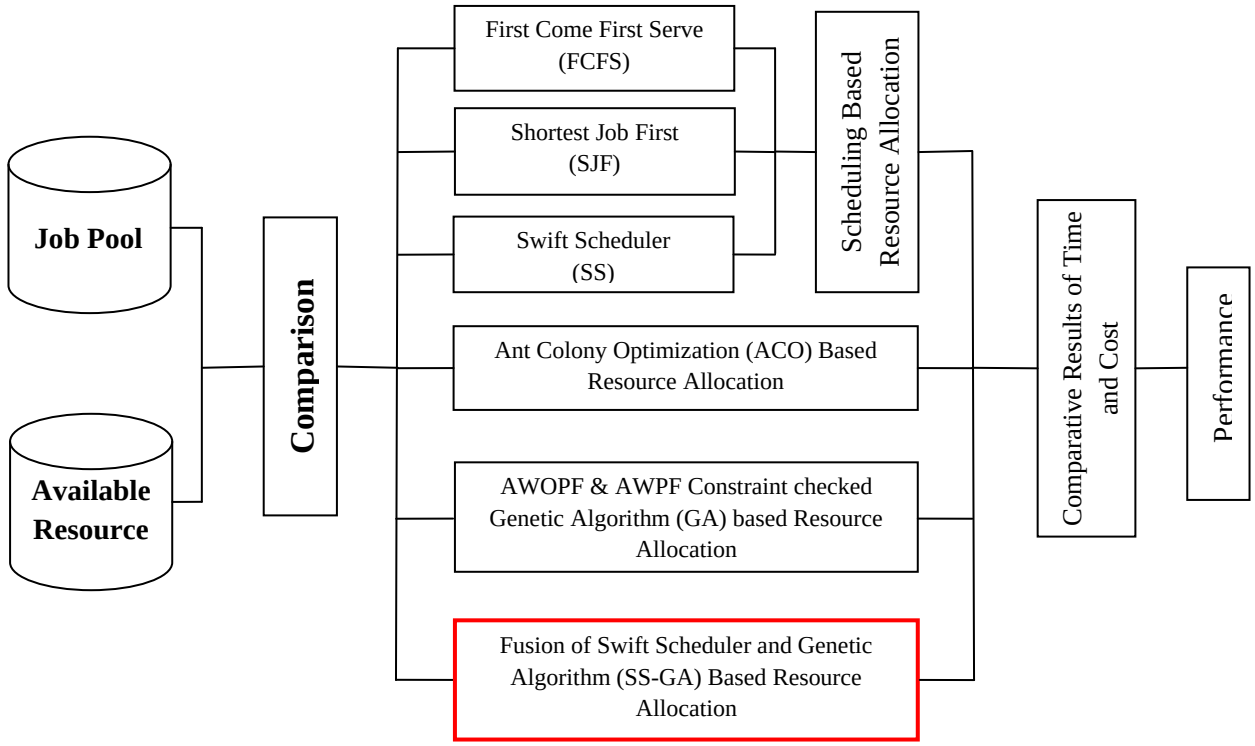

Figure 1 : The comparative model of various algorithms with proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based Resource Allocation

The paper is structured as follows: The related works are declared in Section 2. The proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation algorithm and its architecture are derived in Section 3. The experimental simulated resource allocation results are showed in Section 4. And Section 5 concludes the paper.

II. Related Works

A number of methods are being addressed by various researchers in the past; in this section some of the methods related to resource allocation are provided in this section.

Fatos Xhafa et al. [3] have presented for the minimization of makespan and flowtime by designing the efficient Grid Scheduler with the usefulness of Genetic Algorithm (GA). Along with Genetic Algorithm two encoding schemes are used separately with GA, empirically studied and implemented. The pervious works are based only with the makespan estimation of the resource and this experimental study in the Grid Schedulers surpasses than the previous one by adding flowtime minimization with makespan minimization in a contemporaneous optimization mode. Furthermore, this encoding based Genetic Algorithms versions can able to discover the best and useful in real grids, which gives the best grid characteristics. This GA based grid scheduler can schedule the job arriving in the grid system dynamically and very fast by executing the work in the short time.

Mayank Kumar Maheshwari et al. [9] have proposed the load balancing technique for proper distribution of the jobs. In general, Decision making, Information Collection and Data Migration are the three types of phases in Load Balancing. Load balancing algorithm which improves parallelism of work, proper distribution of task need to reduce the response time and resource utilization increases throughput management. Load balancing algorithm has two types of nature, static and dynamic. In this paper, they have proposed the optimal scheduling using Load balancing. This algorithm schedules the task by their minimum work completion time and to obtain load balance, it reschedules the waiting time of the works. This paper was based on the dynamic nature of the load balancing algorithm. By rescheduling the works according to the waiting time of the task tries to provide the best solution. It reduces the execution time of the job and with effective cost for the processing of all the jobs in the grid system.

Karthick Kumar [10] have proposed fair scheduling using Load balancing, and this fair

scheduling is compared with existing scheduling algorithms includes Adjusted Fair Task Order, Earliest Deadline First, Max Min Fair Scheduling and Simple Fair Task Order in the operational grids. It concentrated mainly the fairness issues by using mean waiting time. This algorithm schedules the task by their fair completion time and to obtain load balance, it reschedules using mean waiting time of the works. This fair scheduling algorithm scheme tries to reduce the execution time and minimization of cost for all jobs provides best solution in the computational grid. Adjusted Fair Task Order, Earliest Deadline First, Max Min Fair Scheduling and Simple Fair Task Order algorithms are compared with the proposed algorithm by using simulation.

Murugesan et al. [11] have proposed a resource allocation framework in a grid computing environment for heterogeneous workloads, subject to a set of conditions. The resource allocation approach can manage and assign the task to the available grid resources with the minimization of cost from user side. They proposed a mathematical model to allocate the user task in the job pool to the accurate resources in the grid resources pool with the purpose to reducing the grid user's costs with reference to the time limit and budget denoted by the grid user. In this paper, they presented a reasonable model to consider the Quality of Service (QoS) parameter requirements to complete the user task, and at the same time they examine the performance of this proposed algorithm.

Manavalasundaram et al. [12] have proposed Grid association states the method for synchronized sharing of distributed clusters based on the computational economy, permits the grid users to use the local resources from the grids association but doesn't satisfies the users requirement. The computational economy methodology used here as organizing the resource which not only satisfying the Quality of Service (QoS) parameters, but also the best performance of the resources. The entire self-provisioning for the each user's as autonomous world in Grid Association. The proposed efficient methodology for resource management can use by grid user effectively. The local resources in the grid association doesn't met the requirement of the grid user, so it the job migrate remote resources according to the user's condition in Quality of Service (QoS) requirements. The global scheduler manages and schedules the user job in the Virtual Organization (VO) and that imposes Virtual Organization (VO) - wide policies. The agents used in the grid association to access and maintain the shared directory of the association of the grid resources. The experimental results of this work shows that the resource which have high capability to execute the job in low mean time and cost effective, and so there is a very huge competitions from all other jobs in the association to that particular resource to satisfy their Quality of Service (QoS) requirements, the resource association provides a increased ability to satisfy Quality of service requirement, in general the resource allocation methodology provides an increased ability to satisfy Quality of service (QoS) parameter needs overall the grid users.

Navjeet Kaur et al. [13] have proposed managing the resource using inter-intra fairness scheduler in grid environment. In grid computing environment an essential role is resource allocation management. The grid system responsibility is to make sure weather the client's jobs/applications request are getting the best resources in cost effective and timely manner. In grid computing technology, there are many resource allocation algorithm are there making jobs allocate to the resources allocation decisions. In this paper, they describes the resource management with different proportional share scheduler with O(1) precision in grid environment. While allocating the resource to the job, the resource allocation's fairness and efficiency that ensure by proportional/fair share scheduler. The proposed inter - intra fairness scheduler is the integration efficient fair share scheduler features as well as managing resources and job scheduling issues in a soft way.

Rajkumar Buyya et al. [14] have identified challenges in grid resource management in the grid computing environment and proposed an effective resource allocation management and job scheduling as metaphor in the computational grid environment. Their projected algorithm analyzes the challenges and requirements in the distributed resource allocation of economy based grid system. The various agent economy based system considering both promising and historical. The CPU cycles, storage, and network bandwidth of the resources are considered by the various economy based system. It presented an extensible and leverage of existing resource management technology and service oriented grid architecture in the grid system. And it also presented auction models and merchandise for the resource management in the grid technologies. The use job scheduling and merchandise economy model for resource management in both data grids and operational is also presented.

Zne Jung Lee et al. [15] have proposed cost effective resource allocation process by allocating the job to optimal resources in the grid environment. In this paper, they proposed the hybrid search algorithm heuristics approach for the resource allocation problem is encountered in grid system. The proposed algorithm can explore the search space and exploit the best solution with the dual advantages of Genetic Algorithm (GA) and Ant Colony Optimization (ACO) of population algorithms. The well designed GA and ACO are implemented for resource allocation problem. In addition, heuristics in Ant Colony Optimization are used

to amend the search performance of the resource allocation problem. While testing this proposed algorithm by simulation, the results showed attractive performance for resource allocation problem.

Devaki et al. [16] have proposed population based Genetic Algorithm considering Quality of Service parameters for Job scheduling. The utilization of an idle resource and distributed resources present global grid system to solve the challenges which are computational was greatly encouraged in grid computing. The problems are either in two sides one in scheduling the job or another was executing jobs in available resource in the grid system. Here the main issue was scheduling the jobs in correct resource is considered as NP complete problem, and so it was essential to have an efficient job scheduling to be effective utilization of the resource in the grid system. Various heuristic algorithms are used to solve the issues which bring a nearby optimal solution. In this paper, they proposed an offline mode Genetic Algorithm evolution using Quality of Service (QoS) parameter satisfaction for scheduling the job to heterogeneous resources. The proposed algorithm mainly focused in makespan and Quality of Service (QoS) satisfaction, when selecting the optimal resource.

Prabhu et al. [17] have proposed the resources scheduling approach based on the multi-objective Genetic Algorithm (GA). This proposed multi-objective Genetic Algorithm (GA) focused on the mean waiting time, flow time and optimal resource allocation and batch mode methods used for the allocation of jobs to the favorable resource. They evaluate the performance using the methods with multi-objective parameters based on their computational results. The overall execution time is minimized by Genetic Algorithm (GA) based scheduling. This proposed algorithm was tested with up to 1000 generation, and the experimental performance shows the results achieved near optimal efficiency.

III. The Proposed Algorithm

a) *The basics of Swift Scheduler and Genetic Algorithm*

The swift algorithm and genetic algorithm is the key algorithm in problem solving methods. The swift focus and works according the users requirement and the genetic algorithm is a population based search algorithm which found the optimal solution to the problem.

i. *Swift Scheduler*

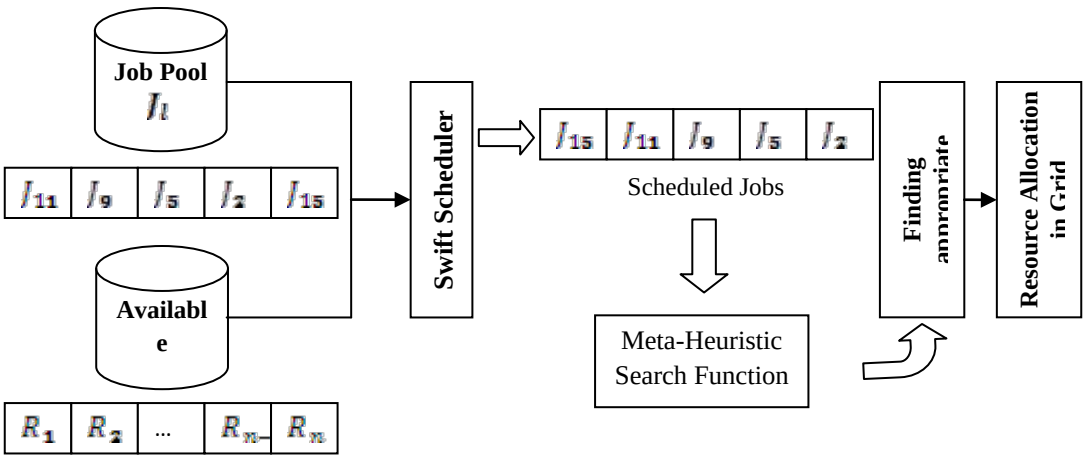

Figure 2 : Swift Scheduling based Resource Allocation

Swift scheduler based resource allocation defines the resource allocation process in the fast resource scheduling manner. Swift scheduler based resource allocation is the combination of the shortest job first and local meta-heuristic search function. Fig.2. shows the processing of swift scheduler based resource allocation. Swift scheduler gives the importance to the jobs, which schedules the jobs according to its length in ascending order (i.e. Shortest Job First (SJF) order). The resource allocation takes the advantage of meta-heuristic search function. The heuristic function searches the resource which has the capability to execute the job in minimal job computational time.

a. *Basic Structure*

The Swift Scheduler based Resource Allocation works as follows:

Begin
- **Step 1:** Initialize the N jobs in job pool;
- **Step 2:** Initialize the available M resource in Grid system;
- **Step 3:** Schedules the N jobs in ascending order based on the job's length;
- **Step 4:** Search the resource using meta-heuristic function based on less computational time;
- **Step 5:** Allocate the jobs to the appropriate resources;

End

Figure 3 : Swift Scheduler based Resource Allocation Structure

ii. *Genetic Algorithm*

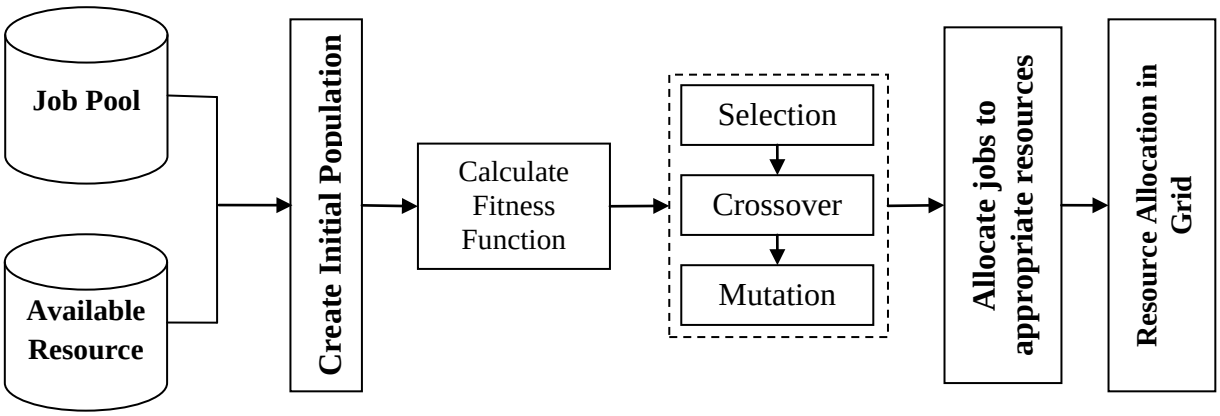

Figure 4 : Genetic Algorithm-based Resource Allocation

The Genetic Algorithm based Resource Allocation defines the resource allocation process by analyzing the compatibilities between jobs and available resources in the Grid system. The compatibility was analyzed by the job's computational time and if the expected computational time was satisfied, the jobs will be allocated to the available grid resources. Fig.3. shows the processing of genetic algorithm based resource allocation. The genetic algorithm can search the available resources in polynomial time by its meta-heuristic search technique. The algorithm initializes the population by random selection of chromosomes. It can explore the appropriate resource by calculating fitness function for each chromosome and selecting the chromosomes for mating for next generation. Crossover was done between randomly selected chromosomes with fixed probability rate, and mutation was also done in the random manner with some fixed probability rate. These operations are repeated every iteration while it satisfies the job with expected computational time in available resources.

a. *Basic Structure*

Begin

Step 1: Initialize the population with randomly generated N chromosomes;
Step 2: Initialize the available M resource in Grid system;
Step 3: Start the loop for finding appropriate resources;
Step 4: Fitness function evaluation for each chromosomes;
Step 5: For next generation - selecting the chromosomes for mating;
Step 6: Crossover between the chromosomes with random pairs;
Step 7: Changing the chromosome with new chromosome i.e. mutation;
Step 8: Stop the loop while the chromosome finds the best resource;

End

Figure 5 : GA based Resource Allocation Structure

b) Fusion of Swift Scheduler and Genetic Algorithm (Ss-Ga) Based Resource Allocation

The main objective of this resource allocation model is to allocate the proper jobs to the fittest resources in order to meet the Quality of Service (QoS) parameters like minimum job completion time, cost efficient, economy, and resource utilization. Our proposed grid resource allocation model combines the swift scheduling mechanism and genetic algorithm to make the resource allocation process as more proficient when compare with other methods. In our methodology, we take the input data set which contains jobs and resources to be processed. A job pool consists of the number of jobs having its id, length of the job and job priority while the resource pool has resource id, its capacity and cost to execute the particular job.

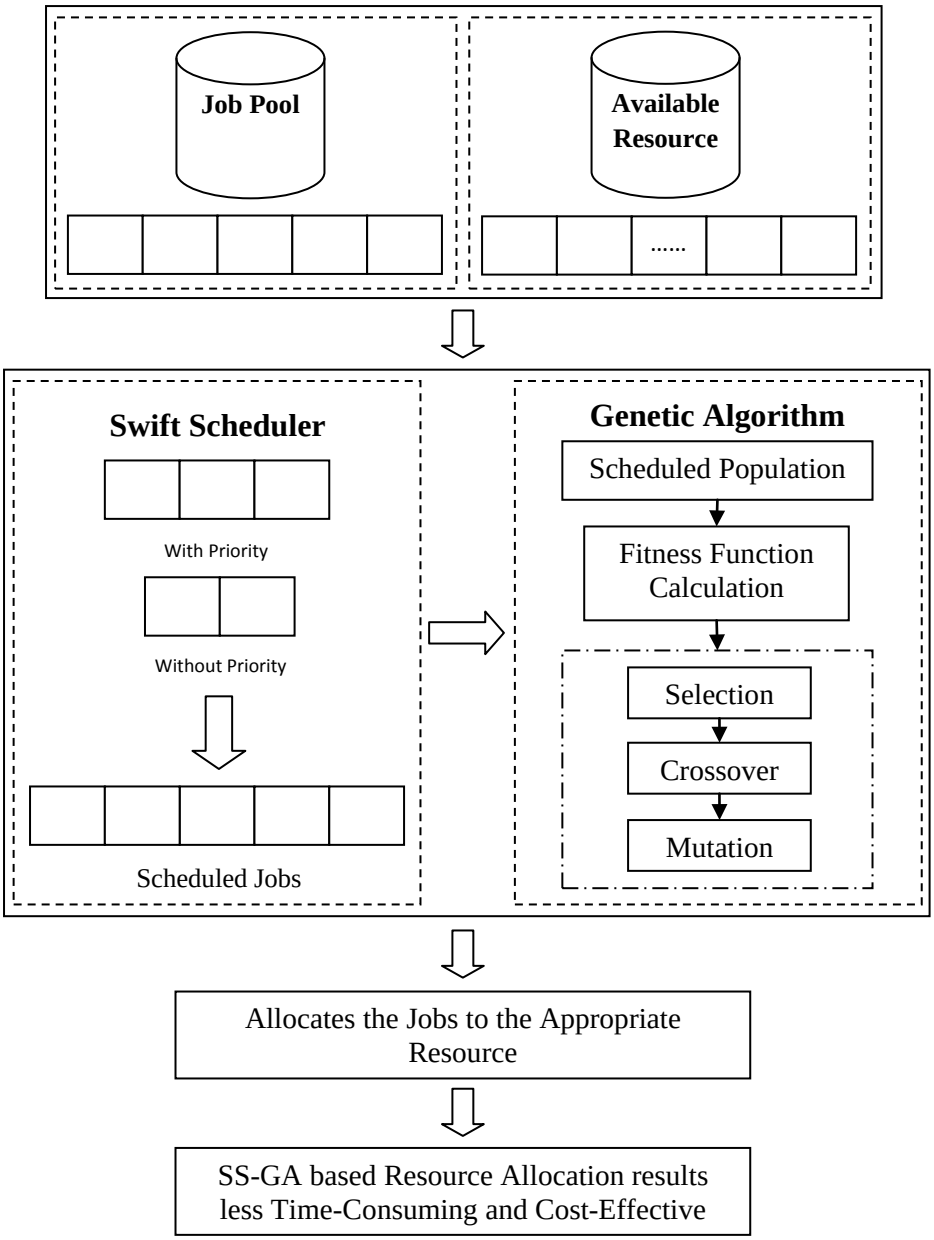

Figure 6 : Proposed SS-GA algorithm for Resource Allocation

i. *Basic Assumptions*

Let us consider that there are n jobs to be processed by m resources or machines where n>m. The following assumptions can be made to achieve the resource allocation as the finest one.

- Every single job is autonomous of each other and there is a priority among them.
- Every resource and jobs is simultaneously available at the early stage of time.
- A resource can only process one job at a time and this procedure cannot be interrupted before completed.
- For every resource and job has its unique id to differentiate each other and avoid conflicts while processing jobs in the resources.
- The migration of jobs is not permitted.
- In job and resources, we need to specify the job length, job priority, resource capacity and resource cost to allocate effectively.

In the proposed SS-GA algorithm, it fuses the swift scheduler with the population based Genetic Algorithm. So it makes full use of advantages of the priority based job scheduling offered by swift scheduler, as well as the powerful accurateness finder ability provided by the Genetic Algorithm. Fig.4. shows the proposed Fusion of Swift Scheduler and Genetic Algorithm (SS-GA) Based Resource Allocation.

ii. *Basic Structure*

Fig. 7 illustrates the algorithmic structure of proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation. The

proposed algorithm accepts the dynamic nature of grid resource availability and users' task.

Begin

Step 1: Initialize the jobs in the job pool;
Step 2: Initialize the available resources in resource pool;
Step 3: Schedule the job with high priority in ascending order;
Step 4: If the job with no priority, schedules the job in shortest job first order;
Step 5: Initialize the scheduled Population size P(s) based on the number of jobs;
Step 6: Perform a Fitness function to minimize the following objective function;
Step 6.1: Job Completion Time (J_C);
Step 6.2: Job Completion Cost (C_F);
Step 7: Selecting half of the individuals among the population P(s) having best fitness values;
Step 8: Apply One-Point Crossover for the selected individuals with a crossover rate C_r;
Step 9: Mutation to be carried out at the rate of M_r to obtain the new better chromosomes;
Step 10: Terminate the above process after 'n' iterations to achieve best results in case of objective function in resource allocation;

End

Figure 7 : Proposed SS-GA Algorithm Structure

iii. *Flow Chart*

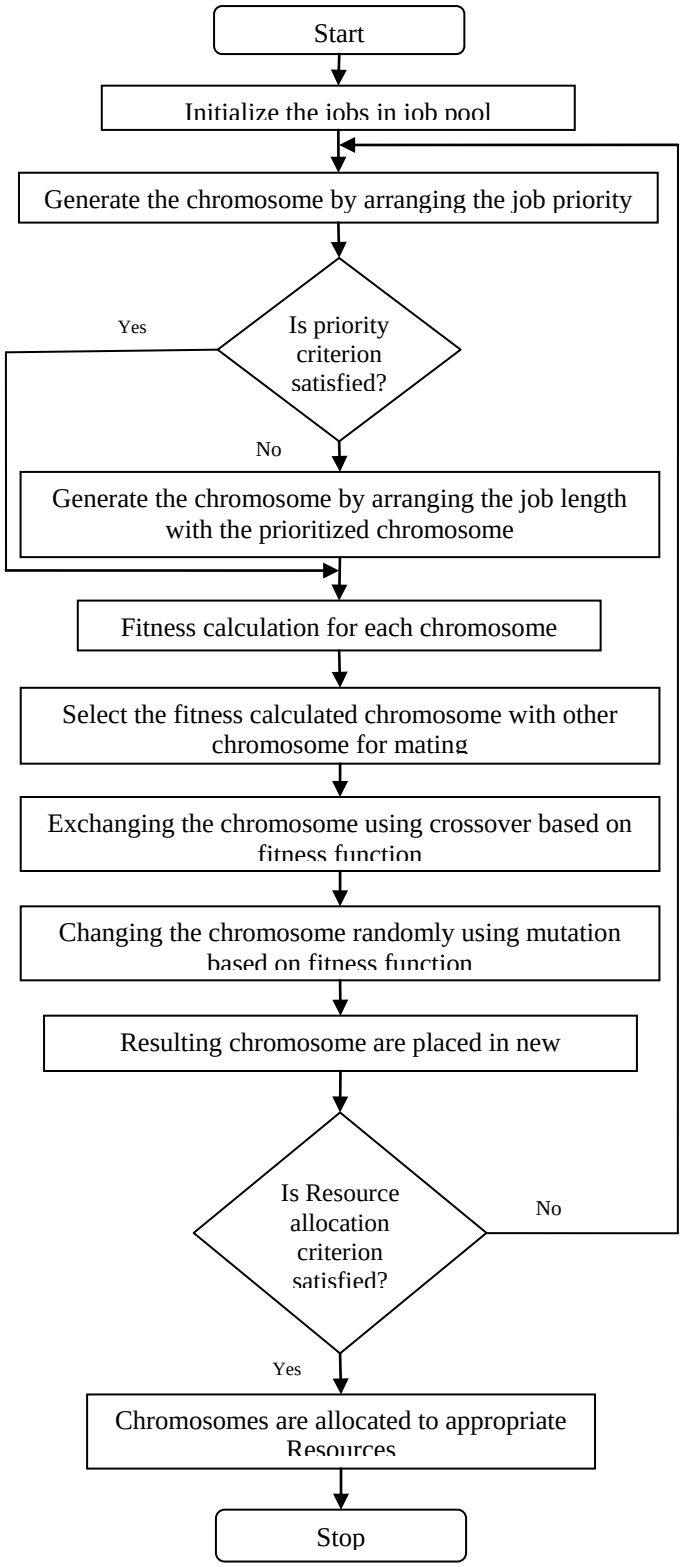

Figure 8 : Flowchart for proposed SS-GA algorithm

Nomenclature

J_l	Job length in million instructions
J_P	Job priority (0 to 5)
R_{id}	Unique id of each job
	Job cost
R_{Cap}	Capacity of a resource in MIPS
R_{Cos}	Resource cost per minute of execution
R_{id}	Resource id
	Crossover rate
	Mutation rate
	Makespan or Overall completion time
	Total job completion cost
Ch_l	Position of the chromosome

Mathematical Steps

BEGIN

Step 1:

Initialize Job length (J_l), Job Id (R_{id}), Job priority (J_P), Resource Id (R_{id}), Resource Capacity (R_{Cap}), Resource Cost (R_{Cos}), Chromosome length (Ch_l), Population size;

Step 2:

Swift schedules the jobs in the job pool according to the user's importance i.e. priority;

Step 2.1: Priority of the jobs are randomly assigned between 0 to 5, 0 holds the high priority of the jobs and some jobs doesn't hold priority;

Step 2.2: Separate the jobs with (J_L^P) and without (J_L) priority in the job pool;

Step 2.3: With prioity jobs (J_L^P):

If P =! Null;

For each J_L^P

P[j] < P[Min]; Min = j; $\forall J_L^P$

If P[i] == P[j];

Consider the length of the jobs

For each J_L^P

L[j] < L[Min]; Min = j; $\forall J_L^P$

Arrange the jobs according to the priority of the job;

Ptmp = P[i]; P[i] = P[j]; P[j] = Ptmp;

Step 2.4: Without priority jobs (J_L):

If P == Null

For each J_L

L[i] < L[Min]; Min = j; $\forall J_L$

Arrange the jobs according to the length of the job;
Ltmp = L[i]; L[i] = L[j]; L[j] = Ltmp;

Generally, $$\mathbf{Job} = \begin{cases} \mathbf{J_l^P}, & \textbf{order the job by its priority} \\ \mathbf{J_l}, & \textbf{otherwise order using length} \end{cases}$$

Step 2.5: Finally the chromosomes are arranged in the job pool by prioritized jobs followed by the without prioritized jobs;

Step 3:

Initialize the chromosomes scheduled by the swift scheduler;

Step 3.1: Evaluation of fitness function

Step 3.1.1: Time fitness

For each job's completion time: $$Jc = \frac{J_L}{R_{Cap}}$$

Overall completion time: $$O_{J_C} = \sum_{i=1}^{n} Jc$$

Step 3.1.2: Cost fitness

For each Job's cost: $$C_F = R_{Cos}.J_C$$

Overall cost: $$T_{C_F} = \sum_{i=1}^{n} C_F$$

Step 3.2: Best chromosome selection

$$\mathbf{Selection} = \frac{\mathbf{Pos_{c1}} + \mathbf{Pos_{cL}}}{\mathbf{2}}$$

Here, c1 & cL are first chromosome and last chromosome respectively;

For each selection & total chromosome length

$$\mathbf{Pos_{cN}[i,j]} = \mathbf{Pos_c[Selection[i]]};$$

Here, cN represents the intial best chromosomes;

Step 3.3: Chromosome Crossover

$$Div = \frac{Chrom_L}{4}, Pnt_1 = Div * 2, Pnt_2 = Div * 3;$$

For each $Chrom_L$

$$Chrom_1[j] = IChrom[i,j];$$
$$Chrom_2[j] = IChrom[i+1,j];$$

Applying crossover,

$$L_V = Pnt_2 - Pnt_1;$$

For each L_V

If $\{(L_V + i)\ \%\ 3\} \neq 0$

Crossover the chromosomes;

Step 3.4: Chromosome Mutation

$$Mut_{pnts} = 2;$$

For each Mut_{pnts}

$$Tmp = \{Rand_{Mut_{Ind}}[j] - 1\}\ \%\ 3;$$

If Tmp == 1

$$BChrom\left[i, \{Rand_{Mut_{Ind}}[j] - 1\}\right] = Rand_{Chrom}(1,10);$$

If Tmp == 2

$$BChrom\left[i, \{Rand_{Mut_{Ind}}[j] - 1\}\right] = Rand_{Chrom}(0,4);$$

Step 3.5: Termination

If cretiria satisfied

Return the results of time and cost;

Else Resturn to main;

Step 4:

Display the results of total best time and cost of the allocated job;
And find the resource utilization percentage and economy rate;

Finally, compare all the results with the existing resource allocation algorithm.

END

iv. *Swift Scheduler Process*

The fusion of swift scheduler and genetic algorithm (SS-GA) based resource allocation defines the resource allocation process in the fast resource scheduling manner. The word swift describes the capability of the work process to be done in high speed. The resource allocation algorithm uses the advantage of the swift algorithm by allocating the jobs in available resource according to user requirement. So the job with priority gives more importance for the resource allocation. The job assigned with priority denotes the urgency of job and '0' is considered as the highest priority among the jobs in job pool. The swift algorithm uses search algorithm to find the optimized available resource in the grid system. The search algorithm finds optimized available grid resource for the high-priority job; it makes an effective resource allocation. Here genetic algorithm used along with swift algorithm to find the optimized available resource in proposed grid system. Swift scheduler separates to the prioritized jobs, and non prioritized job in the job pool. And it arranges/schedules the prioritized job (i.e. priority numbered '0' is considered as high priority job) and the non-prioritized jobs are schedules the jobs according to its length in ascending order (i.e. Shortest Job First (SJF) order) as shown in the fig.4. The jobs are arranged/scheduled using the following formula.

$$Job = \begin{cases} J_l^P, & order\ the\ job\ by\ its\ priority \\ J_l, & otherwise\ order\ using\ length \end{cases} \quad (1)$$

Where, J_l^P denotes the job with priority, J_l denotes the non-priority job, P denotes the priority of the job and / denotes the job's length. The resource allocation takes the full advantage of swift scheduler for fast execution of high prioritized job first. Swift scheduler increases the scalability of memory and computational time. So the swift algorithm ameliorates the resource allocation process.

v. *Genetic Algorithm Process*

After performing the swift scheduling process, job tasks have scheduled according to its priority, and the procedure can be merged with genetic algorithm to achieve the objective function of less job completion time, cost-effective and its economy, and resource utilization. A Genetic Algorithm (GA) has four major steps: fitness, selection, crossover, and mutation. Genetic algorithm is capable of solve optimization problems by carrying out the genetic process. A possible solution to a definite problem may be represented as a chromosome containing a sequence of genes. Initially, the population size is a set of chromosomes is generated, and it undergoes many genetic processes. By using selection, crossover and mutation operations, GA is capable of progress towards the population to generate a best possible solution of resource allocation in grid environment.

a. *Initial Chromosome Representation*

The GA operates on a population of chromosomes, which is encoded according to the problem. The chromosome represents a complete solution to the problem. The chromosome is a collection of n number of jobs to be initialized randomly with resources from 1 to R_L where L = {1,2,3,……N}. Therefore, the number of chromosome sequence can be generated based on population size P(s).

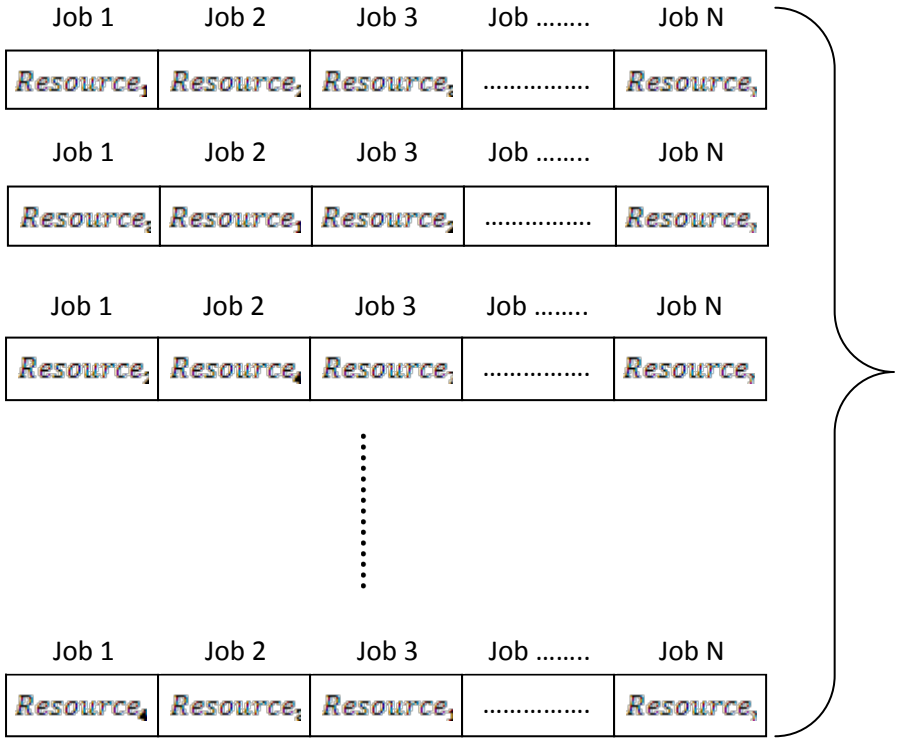

Figure 9 : Chromosome Representation of Jobs and Resources based on Population size

b. *Evaluation of Fitness Function*

In genetic algorithm, fitness function is the most important step for optimize the solutions in grid resource allocation. Fitness is initially resulted from the objective function and used in consecutive genetic operations. The objective function may be minimized or maximized based on the problem occurs in various domains. The purpose of a fitness function is to afford a significant, quantifiable, and equivalent value given a set of genes present in the chromosome. Here the objective function is cost efficient and minimum job completion time.

Fitness can be defined as the value assigned to an individual based on how far or close an individual is from the solution; greater the fitness value superior the solution it contains. It mainly considers two parameters those have to be optimized as follows

Job Completion Time (J_C)

It can be defined as the time taken by a job to be executed successfully in the given allocated resource,

$$J_C = {}^{J_L}/_{R_{Cap}} \quad (2)$$

In this equation, J_L is the length of the job and R_{Cap} is the resource capacity to finish the certain job.

From this, the overall job completion time (O_{J_C}) can be calculated as it is the sum of execution of all the n jobs running on their corresponding allocated resources as,

$$O_{J_C} = \sum_{i=1}^{n} J_C \quad (3)$$

Where i = {1,2,3,….n} denotes the number of jobs to be assigned for appropriate resource.

Job Completion Cost (C_F)

The cost of executing a job in specific resource is known by multiplying job execution time with its resource cost of the allocated resource as,

$$C_F = R_{Cos} \cdot J_C \quad (4)$$

Therefore, the total cost occurred to complete the grid resource allocation can be derived as,

$$T_{C_F} = \sum_{i=1}^{n} C_F \quad (5)$$

The fitness function F_X can take these above parameters to evaluate the initial population of jobs and resources arranged in Chromosome representation. It guarantees that the resource allocation is optimized in case of time and cost by refining the population.

c. *Selection Process*

The selection of best individuals among the initial population size of chromosomes can be done after the fitness operation is performed. It is a genetic operator used in genetic algorithms for selecting potentially useful solutions for recombination. It plays

the major role in well-organized allocation of resources because it only picks the best fitness chromosomes to undergo the genetic operations namely crossover and mutation. It selects the chromosomes having probability of best fitness values. If there are two or more chromosomes having the same best fitness, one of them is chosen randomly. Finally, there are n/2 chromosomes are selected for a genetic operation where n is the population size to produce new chromosomes.

d. *Crossover Operation*

Crossover is a genetic operator that combines or mates' two chromosomes (parents) derived from selection process to generate a new child chromosome (offspring). The two chromosomes (strings) take part in the crossover operation is known as parent and the ensuing chromosomes are known as child strings. This process takes place by fixing the crossover rate C_r as 60% to 70% for making this process a more effective. There are several types of crossover is carried out in genetic algorithm like one-point crossover, two-point crossover, arithmetic crossover, uniform crossover and heuristic crossover. Here the crossover function used is one-point crossover, because one-point crossover is suitable for task ordering problems.

One-Point Crossover

This crossover function selects a particular point randomly in the parent chromosomes to be induced, and then it interchanges the two-parent chromosomes at this point to create two new offspring. A point which selected randomly plays the significant role in one-point crossover. This process carried entirely depends on the crossover rate C_r. Let us consider that the two-parent chromosomes have been selected for undergone the crossover process as shown below.

Parent 1: **110010 | 00111010**

Parent 2: **010110 | 10111001**

The crossover point of both parent chromosomes is denoted by varying the colors in the chromosome. The offspring (child) chromosomes can be produced by interchanging their strings according to the crossover point as,

Child 1: **110010 | 10111001**

Child 2: **010110 | 00111010**

Figure 10 : One-Point Crossover

e. *Mutation*

If the chromosomes arrived after many steps of crossover, there is a chance of strings (chromosomes) having repeated or same one. In order to prevent the particular environment, the mutation is done to gather the best individuals among the parent chromosomes. The mutation has to be performed with the help of mutation rate M_r. This could make creating the completely new genes among the chromosomes in population by altering (changing) gene values. Flip Bit is one of the most common types of mutation used as it changes the bit values of genes from 0 to 1 or vice versa. This mechanism can be illustrated as follows.

Parent 1: **11000110101110010111**

Parent 2: **01001001110110011010**

The child can be produced by invert the gene values present in the above parent chromosomes. The pink color denotes the changed values which are done by Flip bit mutation.

Child 1: **11010110101010010111**

Child 2: **01001011110110011010**

Figure 11 : Flip Bit Mutation

f. *Termination*

After the swift scheduling and operators of genetic algorithm are executed over a certain number of iterations, the jobs stored in the job pool are ready to join their appropriate resource for executes the respective tasks in grid environment. The corresponding efficient resources in terms of flowtime (overall job completion time) and cost can already obtain by the fusion of SS-GA algorithmic methodology to carry out the jobs to finish simultaneously to meet the quality of service.

IV. Experimental Results and Discussion

In this section, the experimental results are discussed in simulated grid network using dotNet framework platform. The proposed algorithm is compared and evaluated with existing algorithm using certain parameters like resource utilization, computational time, processing time, makespan, economy, time, cost etc. And these results are evaluated using separate line graphs. The results are taken in dynamic nature of grid environment. The dataset are simulated which contains job table and resource table separately. The job table contains its id,

length and cost, similarly the resource table also contains its id, capacity and its cost.

a) Resource Utilization

The total resources consumed by the jobs against the expected amount of resources for a particular work. The utilization is normally calculated in percentage of time. Resource utilization is generally, the resource capability of the job execution in a period of time. Resource utilization will differ according to the resource processing speed. The idle resource should have very low utilization rate. The proposed algorithm boosts up all the resource in active stage in dynamic manner, the resources are utilized by the incoming tasks. The utilization formula is defined in equation 6.

$$Utilization = \left(\frac{\sum_{j=1}^{n} J_L(j)}{\sum_{j=1}^{n} R_{Cap}(j)}\right) \times 100 \qquad (6)$$

b) Economy

Economy is the word in grid resource allocation defines vigilant management and proficient or restrained use of resources in grid. "Maximum effect for the minimum effort" is the motto for economy. Economy purports the proficiency of larger resource allocation process with lower price. It will track the changes on the price level with the efficient use of resource price and can analyze and improve the performance of macroeconomic. The economy formula is defined in equation 7.

$$Economy = \left(\sum_{j=1}^{n} \frac{(1 - R_{Cos}(j))}{J_{Cos}}\right) \times 100 \qquad (7)$$

c) Processing Time

The processing time of the tasks are calculated by the following equation 8. It checks the first job with allocated resource and took its length of the job that which execute in the resources. And the upcoming jobs are checked with the all resources when the i^{th} resource id and j^{th} resource id are equal, this has to be done similar manner to all jobs allocated in the resources and so on.

$$Processing\ time = \sum_{i=1}^{n} \sum_{j=1\ \&\&\ i\neq 1\ \&\&\ R_{id}(i)==R_{id}(j)}^{i-1} J_L(i) \qquad (8)$$

The resource in the grid environment may connect and leave at anytime. So the results are taken in a dynamic way by varying job count and resource count in the simulated grid environment using dotNet framework. The proposed SS-GA algorithm with existing algorithms using utilization, economy, total resource cost, processing time and computational time parameters are evaluated.

The following results are compared between First Come First Serve (FCFS) based Resource Allocation, Shortest Job First (SJF) based Resource Allocation, Swift Scheduler (SS) based Resource Allocation, Ant Colony Optimization based Resource Allocation, AWPF Constraint checked Genetic Algorithm (GA) based Resource Allocation, AWOPF Constraint checked Genetic Algorithm (GA) based Resource Allocation with the proposed fusion of Swift Scheduler and Genetic Algorithm (SS-GA) based Resource Allocation.

These algorithms are compared using parameters like resource utilization, resource economy, total resource cost, processing time and computational time. The results are taken with different available resource pool size and in all cases of available resource the dynamic job inputs are compared in following table with proposed algorithm.

Table.1a – Table.1e with constant resource 10 with varying job size are analyzed with proposed algorithm

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	101.13731	101.13731	101.13731	96.8543	104.0636	100.3295	94.96753
Economy	35.97241	35.97241	35.97241	39.2	44.2418	44.12	38.86897
Total Resource Cost	2321	2321	2321	2204	2048	2124	2116
Processing Time	7412	7412	7412	1502	1630	1534	2163
Computational Time	63.4956	41.0576	15.3729	7.2747	217.9281	487.5417	671.9201

Table 1.a : Resource - 10 Job - 40

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	109.9085	125.1736	109.9085	106.0294	105.042	105.9474	102.7066
Economy	40.42171	42.21624	40.42171	39.88335	39.2986	39.4193	39.38986
Total Resource Cost	2656	2679	2656	2680	2700	2754	2652
Processing Time	11529	12214	11529	2210	2262	2320	2789
Computational Time	20.331	35.7789	23.1065	2.2418	285.6306	554.5056	723.9278

Table 1.b : Resource - 10 Job - 50

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	88.96484	125.8287	125.8287	122.4462	91.9287	91.3758	118.7744
Economy	39.90874	45.17375	45.17375	44.82274	39.7398	39.7032	44.41909
Total Resource Cost	3424	3154	3154	3144	3242	3291	3137
Processing Time	19543	18990	18990	3214	3103	3221	3974
Computational Time	26.8791	19.9335	22.5993	3.7538	329.6535	614.6301	757.6617

Table 1.c : Resource - 10 Job - 60

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	106.4024	106.4024	100.2874	103.869	97.8095	98.6705	97.30483
Economy	42.00766	42.00766	41.08812	41.63985	39.3844	39.376	40.5977
Total Resource Cost	3784	3784	3844	3808	3840	3886	3776
Processing Time	27525	27525	28899	4252	4090	4225	5186
Computational Time	43.6689	14.5231	25.3487	3.0638	433.0316	718.1304	830.0515

Table 1.d : Resource - 10 Job - 70

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	99.72919	109.10143	99.72919	104.4964	105.5111	104.6957	97.89385
Economy	38.76685	39.75405	38.76685	39.74287	40.6181	40.6526	38.6948
Total Resource Cost	4389	4669	4389	4312	4304	4365	4297
Processing Time	36040	36703	36040	5333	5208	5332	6375
Computational Time	16.1187	16.3831	21.1927	3.2143	452.622	808.8489	879.7745

Table 1.e : Resource - 10 Job - 80

The above five tables are compared with constant 10 resources with dynamic job inputs. The result with different job inputs explains that the resource utilization is comes best compared to other algorithms, the resources are not fully occupied and it is used effectively. The proposed algorithm maintains the resources and avoiding the resources in idle stage. The economy rate using the resource cost shows proficient resource usage. The most important and expecting parameter should be the cost, the performance of the proposed algorithm bring the effective cost while comparing other existing algorithm. The processing time and computational time are totally depends implemented code.

Table.2a – Table.2e with constant resource 15 with varying job size are analyzed with proposed algorithm

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	115.0763	115.0763	115.0763	99.17764	108.6022	104.8658	98.36868
Economy	43.76489	43.76489	43.76489	41.5409	43.6906	43.8856	41.40852
Total Resource Cost	2224	2224	2224	2208	2133	2198	2203
Processing Time	5399	5399	5399	1152	1162	1208	1242
Computational Time	26.5083	40.0634	24.3662	3.3151	230.8437	498.984	671.4895

Table 2.a : Resource - 15 Job - 40

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	93.17889	93.17889	93.17889	100	102.8329	106.8017	95.01312
Economy	37.70749	37.70749	37.70749	38.89637	36.957	37.3555	38.04397
Total Resource Cost	2777	2777	2777	2724	2685	2710	2682
Processing Time	10018	10018	10018	1664	1620	1633	2056
Computational Time	32.1723	28.8015	28.069	4.2309	272.0314	543.2246	738.77

Table 2.b : Resource - 15 Job - 50

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	99.10011	104.3839	99.10011	102.9206	102.8136	104.2056	97.45575
Economy	39.70669	40.53162	39.70669	40.31164	40.1455	40.6211	39.43172
Total Resource Cost	3289	3264	3289	3256	3292	3327	3254
Processing Time	16148	15478	16148	2203	2290	2385	2720
Computational Time	27.5003	31.5784	22.5031	4.0919	342.8629	632.962	787.1048

Table 2.c : Resource - 15 Job - 60

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	123.5437	115.8134	115.8134	99.12366	93.3148	93.0657	97.32314
Economy	42.60374	41.73266	41.73266	39.38866	39.8867	39.7923	39.08774
Total Resource Cost	3824	3879	3879	3827	3819	3884	3811
Processing Time	22756	23503	23503	2975	2911	2949	3667
Computational Time	29.8284	29.1116	45.4337	5.0326	385.6059	658.5975	809.2128

Table 2.d : Resource - 15 Job – 70

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	95.70403	111.8088	95.70403	100.2521	100.3407	101.2744	99.58263
Economy	38.21313	42.58612	38.21313	40.93111	38.6827	38.8788	38.62347
Total Resource Cost	4592	4267	4592	4390	4329	4372	4338
Processing Time	33765	32349	33765	3900	3842	3950	4436
Computational Time	35.7035	34.5346	33.2747	5.9805	467.8766	868.2343	876.4405

Table 2.e : Resource - 15 Job - 80

The above five tables are compared with constant 15 resources with dynamic job inputs. When the number of resources goes high, the rate of resource utilization maintains its proficiency with the use of proposed algorithm. The resource utilization and economy shows the good rate while using the proposed algorithm. The fusion of SS-GA algorithm gives the computational time is pretty high because the fused algorithm takes both the work of swift algorithm and genetic algorithm.

Table.3a – Table.3e with constant resource 20 with varying job size are analyzed with proposed algorithm

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	117.2917	117.2917	117.2917	100.5357	87.6387	86.9029	93.05785
Economy	39.09224	39.09224	39.09224	36.74963	37.3456	36.8762	35.43192
Total Resource Cost	2180	2180	2180	2160	2181	2239	2155
Processing Time	3532	3532	3532	864	871	896	1256
Computational Time	40.6763	66.9054	35.3175	4.2761	241.6037	514.3924	705.319

Table 3.a : Resource - 20 Job – 40

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	111.026	111.026	111.026	98.10555	103.9589	99.7241	93.06804
Economy	40.80768	40.80768	40.80768	38.88889	40.2273	39.5456	37.99643
Total Resource Cost	2753	2753	2753	2739	2682	2767	2679
Processing Time	7349	7349	7349	1338	1354	1321	1808
Computational Time	40.7577	32.6365	54.1946	6.7784	309.8983	595.1578	732.8609

Table 3.b : Resource - 20 Job - 50

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	100.4372	100.4372	100.4372	97.24868	107.554	112.0098	94.25641
Economy	42.71643	42.71643	42.71643	42.19803	35.8771	36.5934	35.67963
Total Resource Cost	3315	3315	3315	3345	3235	3261	3237
Processing Time	13611	13611	13611	1864	1849	1906	2451
Computational Time	43.4866	34.0002	41.0565	6.0564	383.4802	661.8143	798.3901

Table 3.c : Resource - 20 Job - 60

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	118.5099	118.5099	118.5099	93.65225	104.7856	108.6823	88.59878
Economy	41.86527	41.86527	41.86527	38.24277	39.753	40.184	37.25771
Total Resource Cost	3759	3759	3759	3887	3757	3772	3749
Processing Time	19330	19330	19330	2384	2441	2550	3063
Computational Time	48.9577	60.6802	44.1474	8.5887	415.6853	696.6445	850.4797

Table 3.d : Resource - 20 Job – 70

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	90.8028	90.8028	90.8028	110.7414	98.0504	100.3276	95.5254
Economy	37.80522	37.80522	37.80522	41.00999	39.0302	39.7539	39.28857
Total Resource Cost	4483	4483	4483	4352	4338	4357	4304
Processing Time	28390	28390	28390	2914	3079	3153	3593
Computational Time	38.908	44.2368	40.7869	8.2141	502.8877	908.7781	856.5241

Table 3.e : Resource - 20 Job - 80

The above five tables are compared with constant 20 resources with dynamic job inputs. The economy results of SS-GA algorithm bring good rate, when increase in the number of available grid resources in the grid network. The total resource cost of proposed algorithm with varying resources results the effective cost. The performances of the resource allocation in the grid environment are boosted up by SS-GA algorithm. The processing time and computational time of the ACO based resource allocation algorithm is very low because it allocates the jobs by checking with the resources in first time itself. But the proposed algorithm takes the advantage of swift algorithm; it checks the job with priority brings higher order for allocation process. And so it takes more time for resource allocation computation. The available resources are dynamically checked by the proposed algorithm.

Table.4a – Table.4e with constant resource 25 with varying job size are analyzed with proposed algorithm

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	114.3678	114.3678	114.3678	101.8771	102.0942	104.5534	98.84106
Economy	42.94165	42.94165	42.94165	41.22076	35.8982	35.8967	40.73676
Total Resource Cost	2232	2232	2232	2186	2191	2234	2184
Processing Time	2297	2297	2297	836	842	879	1139
Computational Time	52.7427	45.809	75.3122	8.6043	284.6685	568.168	710.2946

Table 4.a : Resource - 25 Job – 40

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	115.3724	115.3724	87.60529	102.2472	106.1026	108.8235	95.41284
Economy	41.32471	41.32471	36.86441	39.51829	43.0998	43.1747	38.38091
Total Resource Cost	2731	2731	2831	2712	2680	2710	2663
Processing Time	5594	5594	5212	1111	1259	1257	1666
Computational Time	65.6896	47.8802	67.6473	7.7583	313.8459	593.0732	763.5657

Table 4.b : Resource - 25 Job - 50

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	98.71899	99.54338	98.71899	102.9516	102.1378	97.3184	97.31663
Economy	38.54863	39.19822	38.54863	39.73645	37.8291	36.848	37.1611
Total Resource Cost	3311	3276	3311	3247	3282	3378	3238
Processing Time	10016	9418	10016	1555	1613	1627	1881
Computational Time	51.9269	56.5841	49.946	11.9545	319.5781	585.6896	775.5075

Table 4.c : Resource - 25 Job - 60

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	103.6847	103.6847	87.55402	102.5304	103.6585	101.4519	97.70473
Economy	39.68381	39.68381	36.80933	39.50814	39.2811	38.5519	38.061
Total Resource Cost	3777	3777	3957	3788	3733	3819	3726
Processing Time	15466	15466	14706	2007	2279	2345	2427
Computational Time	78.7174	52.712	52.2184	11.8828	385.6945	651.1487	854.7021

Table 4.d : Resource - 25 Job - 70

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	91.76289	100.2523	100.2523	101.4468	99.2599	100.1643	98.6198
Economy	41.39925	42.88093	42.88093	41.0965	41.428	41.4092	41.09644
Total Resource Cost	4499	4389	4389	4375	4356	4399	4343
Processing Time	22542	23287	23287	2602	2672	2764	3185
Computational Time	37.1361	40.0114	50.1337	15.8017	444.0259	787.7263	904.6483

Table 4.e : Resource - 25 Job - 80

The above five tables are compared with constant 25 resources with dynamic job inputs. In the table.4a shows the slight change in economy rate of proposed algorithm with AWPF and AWOPF and resource cost also ACO based resource allocation algorithm's very nearest to the proposed one. When there is increase in number of job in resource allocation process, the resource utilization and economy rate are being fine. And the small scheduling based algorithm i.e. FCFS, SFJ, SS are maintaining good results but particularly in some cases because it only gives big concentration in scheduling the jobs to resources and it won't consider the cost of the resources while resource allocation.

Table.5a – Table.5e with constant resource 30 with varying job size are analyzed with proposed algorithm

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	88.854	103.4735	103.4735	94.64883	105.5172	99.6795	90.56
Economy	35.81062	38.39311	38.39311	36.9297	41.4513	39.8808	36.15495
Total Resource Cost	2237	2147	2147	2198	2130	2219	2125
Processing Time	1180	1304	1304	724	801	880	914
Computational Time	60.8625	51.836	57.4498	14.2908	216.1684	481.3565	729.9444

Table 5.a : Resource - 30 Job - 40

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	105.7991	102.8886	105.7991	97.26918	102.6243	101.7568	96.76585
Economy	42.20752	41.78053	42.20752	40.88386	40.305	39.7933	40.79846
Total Resource Cost	2727	2737	2727	2769	2740	2796	2723
Processing Time	4040	3586	4040	1057	1081	1100	1372
Computational Time	105.4399	82.6424	56.9463	13.9463	286.4863	602.7403	776.9802

Table 5.b : Resource - 30 Job – 50

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	90.24918	90.24918	82.231	88.05497	96.973	99.2366	85.87629
Economy	34.49635	34.49635	32.72226	34.04297	38.0134	38.1965	33.56988
Total Resource Cost	3423	3423	3413	3346	3245	3283	3340
Processing Time	6801	6801	7059	1277	1459	1557	1713
Computational Time	56.6984	60.839	53.8097	12.8454	353.1478	649.9953	807.4937

Table 5.c : Resource - 30 Job - 60

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	123.7695	123.7695	123.7695	102.6892	96.4486	96.8692	97.63258
Economy	42.99388	42.99388	42.99388	40.31069	37.2644	37.7858	39.49474
Total Resource Cost	3733	3733	3733	3804	3761	3810	3756
Processing Time	12566	12566	13079	1799	1966	1934	2348
Computational Time	52.7387	45.2918	58.1812	20.3609	378.1919	647.6226	860.2208

Table 5.d : Resource - 30 Job - 70

	FCFS	SJF	SS	ACO	AWPF	AWOPF	SS-GA
Utilization	118.8993	113.0306	113.0306	91.67334	97.1264	95.8333	90.08655
Economy	40.81604	40.1052	40.1052	36.75007	39.8438	39.4922	36.43731
Total Resource Cost	4363	4313	4313	4349	4312	4385	4301
Processing Time	17925	18260	18260	2199	2282	2383	3013
Computational Time	76.1423	84.2506	68.7347	19.757	479.4874	843.1664	918.3843

Table 5.e : Resource - 30 Job – 80

The above five tables are compared with constant 30 resources with dynamic job inputs. The proficient resource allocation in grid computing and the raise in the performance show best result with the proposed fusion of SS-GA based resource allocation. And total resource cost while resource allocation process.

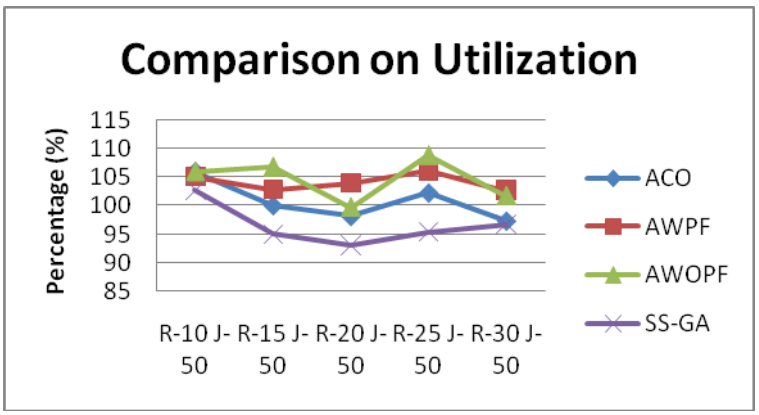

Figure 12 : Comparison on Utilization with constant Job 50

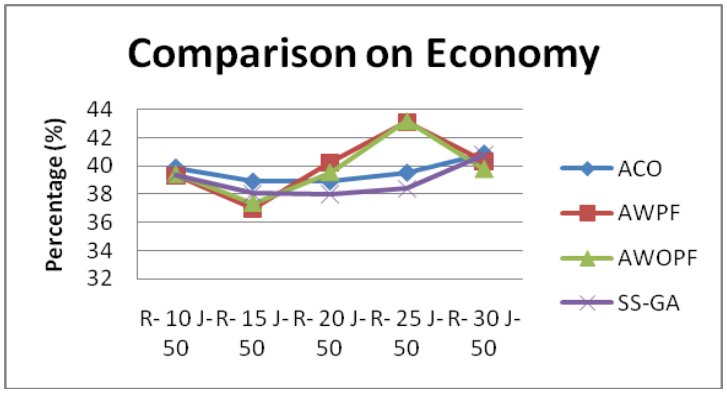

Figure 13 : Comparison on Economy with constant Job 50

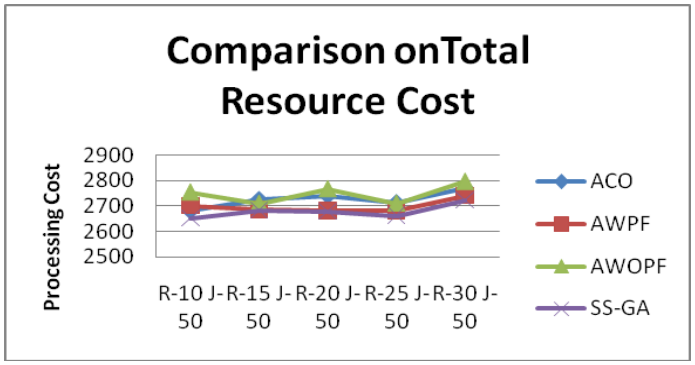

Figure 14 : Comparison on Total Resource Cost with constant Job 50

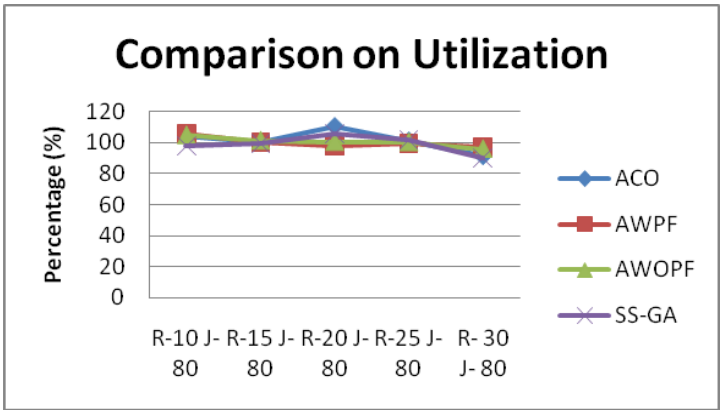

Figure 15 : Comparison on Utilization with constant Job 80

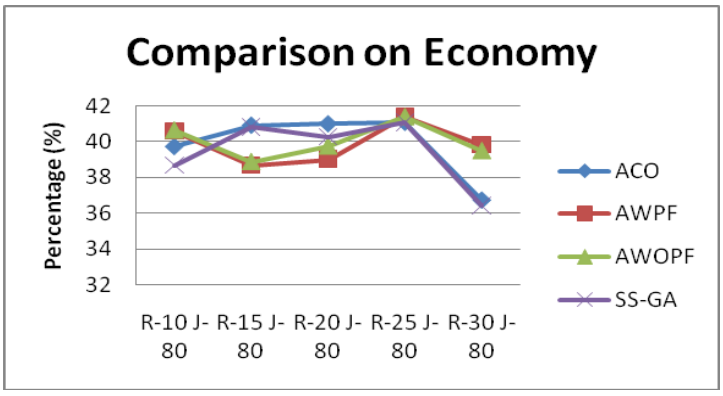

Figure 16 : Comparison on Economy with constant Job 80

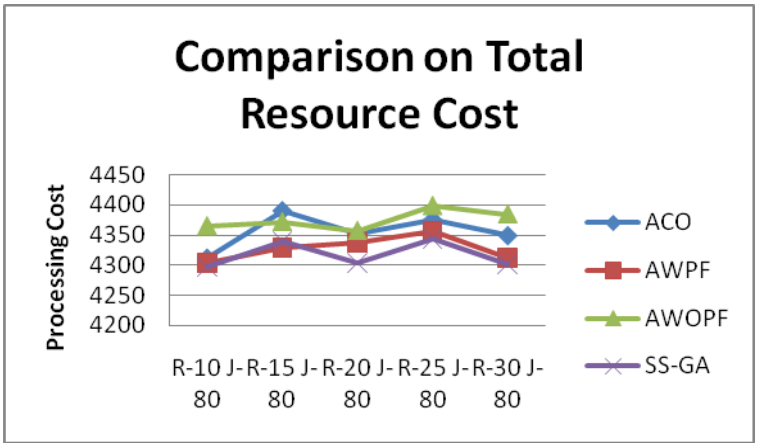

Figure 17 : Comparison on Total Resource Cost with constant Job 80

The above line graph shows the proficiency of the proposed fusion Swift Scheduler and Genetic Algorithm (SS-GA) based resource allocation.

V. Conclusion

In this paper, a mechanism was proposed to allocate the jobs efficiently to the corresponding resources during the process. The proposed method was implemented in the Microsoft Visual Studio 9.0 environment with dotNet framework. This paper has taken GA-SS based algorithm to achieve the quality of service in the resource allocation. In the first phase, the jobs in the job pool are to be scheduled according to its priority. The scheduling of these jobs was done by Swift scheduler. In the second phase, the scheduled jobs can obtain their corresponding resource to allocate by Genetic algorithm. The genetic algorithm performs major operations like fitness, crossover and mutation in the scheduled jobs to achieve less flow time and cost. The experimental result shows that the proposed GA-SS based scheme was much better than the other traditional methods like First Come First Serve (FCFS), Shortest Job First (SJF), Swift Scheduler (SS) and Ant-Colony Optimization (ACO) algorithms in terms of makespan, cost, resource utilization and economy. And also, the experimental results were verified that the proposed methodology can be adaptable to give good results while varying the number of jobs and resources in the input data set.

References Références Referencias

1. Georgiana Marin, "Grid Computing Technology", Database Systems Journal, Vol. 2, No. 3, 2011.
2. Rajeev Wankar, “Grid Computing With Globus: An Overview and Research Challenges”, International Journal of Computer Science and Applications, Vol. 5, No. 3, pp 56-69, 2008.
3. Ajith Abraham, Fatos Xhafa, Javier Carretero, "Genetic Algorithm Based Schedulers For Grid Computing Systems", International Journal of Innovative Computing, Information and Control, Vol. 3, No. 5, pp 1-19, 2007.
4. Mathiyalagan, Suriya, Sivanandam, "Modified Ant Colony Algorithm for Grid Scheduling", International Journal on Computer Science and Engineering, Vol. 2, No. 2, pp 132-139, 2010.
5. Akash Malhotra, Kunal Gupta, "A New Framework for Job Scheduling and Resource Management in Grid Environment", International Journal of Emerging Technology and Advanced Engineering, Vol. 2, 2012.
6. Adil Yousif, Abdul Hanan Abdullah, Muhammad Shafie, Abd Latiff, Mohammed Bakri Bashir, "A Taxonomy of Grid Resource Selection Mechanisms", International Journal of Grid and Distributed Computing, Vol. 4, No. 3, 2011.
7. Haruna Ahmed Abba, Nordin B. Zakaria, Nazleeni Haron, "Grid Resource Allocation: A Review", Research Journal of Information Technology, Vol. 4, pp 38-55, 2012.
8. Harsh Bansal, Babita Pandey, Kewal Krishan, "Intelligent Methods for Resource Allocation in Grid Computing", International Journal of Computer Applications, Vol. 47, No.6, 2012.
9. Mayank Kumar, Maheshwari, Abhay Bansal, "Process Resource Allocation in Grid Computing using Priority Scheduler", International Journal of Computer Applications, Vol. 46, No.11, 2012.
10. Karthick Kumar, "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", International Journal of Computer Science Issues, Vol. 8, No. 1, 2011.
11. Murugesan, Chellappan, "An Economic Allocation of Resources for Divisible Workloads in Grid Computing Paradigm", European Journal of Scientific Research, Vol. 65, No. 3, pp. 434-443, 2011.
12. Manavalasundaram, Duraiswamy, "Association Based Grid Resource Allocation Algorithm", European Journal of Scientific Research, Vol. 78 No. 2, pp. 248-258, 2012.

13. Navjeet Kaur, Harpal Kaur, Satinder Pal Ahuja, "Inter-Intra Fairness Scheduler for Resource Management in Grid Environment", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, 2012.
14. Rajkumar Buyya, David Abramson, Srikumar Venugopal, "The Grid Economy", Proceedings of the IEEE, Vol. 93, pp 698 - 714, 2005.
15. Zne-Jung Lee, Chou-Yuan Lee, "A hybrid search algorithm with heuristics for resource allocation problem", Information Sciences - Elsevier, Vol. 173, pp. 155–167, 2005.
16. Devaki, Valarmathi, "Job Scheduling using Genetic Algorithm with QoS Satisfaction in Grid", European Journal of Scientific Research, Vol. 74, No. 2, pp. 272-285, 2012.
17. Prabhu, Naveen Kumar, "Multi-Objective Optimization Based on Genetic Algorithm in Grid Scheduling", International Journal of Advanced In Technology, Vol. 1, pp. 54-58, 2011.
18. K. Somasundaram, S. Radhakrishnan, "Task Resource Allocation in Grid using Swift Scheduler", International Journal of Computers, Communications & Control, Vol. 4, No. 2, pp. 158-166, 2009.
18. Zafril Rizal M Azmi, Kamalrulnizam Abu Bakar, Abdul Hanan Abdullah, Mohd Shahir Shamsir, Wan Nurulsafawati Wan Manan, "Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Environment", International Journal of Grid and Distributed Computing, Vol. 4, No. 3, 2011.

This page is intentionally left blank