ANT COLONY OPTIMIZATION ALGORITHM FOR LOAD BALANCING IN GRID COMPUTING

MONOGRAPH SERIES

SWARM INTELLIGENCE

ANT COLONY OPTIMIZATION ALGORITHM FOR LOAD BALANCING IN GRID COMPUTING

KU RUHANA KU MAHAMUD
ANIZA MOHAMED DIN

UUM PRESS
Penerbit UUM
Universiti Utara Malaysia

UUM Press
Universiti Utara Malaysia
06010 UUM Sintok
Kedah Malaysia.

Tel: 04-9284958
Fax: 04-9284142
E-mail: penerbit@uum.edu.my
Website: http://uumpress.uum.edu.my

First Published 2012

UUM is a member of Malaysian Scholarly Publishing Council (MAPIM).

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data

Ku Ruhana Ku Mahamud
Ant colony optimization algorithm for load balancing in grid computing /
Ku Ruhana Ku Mahamud, Aniza Mohamed Din.
Bibliography: p. 65
ISBN 978-967-0474-09-0
1. Computational grids (Computer systems). 2. Ant algorithms.
I. Aniza Mohamed Din. II. Title.
005.1

Printed in Malaysia by
UUM Press
Universiti Utara Malaysia
06010 UUM Sintok
Kedah Malaysia.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ACO	Ant Colony Optimization
ACS	Ant Colony System
AHS	Adaptive Hierarchical Scheduling
AS	Ant System
BACO	Balanced Ant Colony Optimization
BWAS	Best Worst Ant System
CPU	Central Processing Unit
EACO	Enhanced Ant Colony Optimization
EAS	Elitist Ant System
FCFS	First Come First Serve
GA	Genetic Algorithm
GJAP	Tabu+Granularity-based Job Allocation Policy
LBTAS	Local Best Tour Ant System
MDS	Metacomputing Directory Service
MMAS	Max-Min Ant System
MOEA	Multi-Objective Evolutionary Algorithm
NP	Nondeterministic Polynomial
NWS	Network Weather System
P2P	Peer-to-Peer
PE	Processing Elements
PPSO	Parallel Particle Swarm Optimization
PSO	Particle Swarm Optimization
PV	Pheromone Value
PVT	Pheromone Value Table
QAP	Quality of Service Access Point
QRC	Quality Research Collection
RN	Recent Neighbour
SA	Simulated Annealing
TS	Tabu Search
TSP	Travelling Salesman Problem
VO	Virtual Organization

PREFACE

Grid computing is developed through a combination of various resources from different geographic locations. This makes grid computing different from conventional distributed computing and cluster computing. One of the most important problems in grid computing is load balancing where all submitted jobs need to be equally distributed among resources. A good load balancing algorithm must be capable of balancing the entire resources by distributing workload evenly across two or more computers, CPUs, network links, hard disk or other resources in order to get optimal resource utilization. Stagnation problem in grid computing will be minimized when all resources are well utilized. Ant Colony Optimization (ACO) is one of the most recent algorithms for load balancing in grid computing. It has been used in solving the scheduling problem between the submitted jobs and available resources in grid computing. ACO algorithm is used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems.

The study on load balancing of resources in grid environment is presented in this monograph. An enhancement of existing ant colony based algorithm has proposed that it can schedule jobs to resources with the aim of balancing the load on all resources. Dynamic scheduling of jobs can be processed by the resources is implemented where the scheduling depends on the changing rate of the evaporation value. Findings from this research will contribute to another grid resource management algorithm that can significantly improve the performance of the available ACO algorithms. The new algorithm enhances the classical approach of ACO algorithm by dynamically scheduling submitted jobs to suitable resources, thus offering the chance to improve the efficiency of managing resources in grid computing.

The structure of the monograph is as follows. An overview of grid computing and ACO algorithm concept are introduced in Chapter 1. The discussion on ant-based approach for managing resources in grid computing and current approaches to control stagnation problem are presented in Chapter 2 while Chapter 3 highlights the methodology and framework for resource management in grid environment. Chapter

4 presents the proposed ant based resource management algorithm. The grid resource management scenario and details of the proposed algorithm are discussed, followed by the description of the design and implementation of the proposed algorithm. Experimental results and analysis of the implementation of the proposed algorithm in grid computing environment are presented in Chapter 5 while contribution of the research and future research directions are highlighted in Chapter 6.

We would like to express our gratitude to Ministry of Higher Education for the financial support under the Fundamental Research Grant Scheme and to Universiti Utara Malaysia for facilitating the management of this research.

Ku Ruhana Ku Mahamud
Aniza Mohamed Din

1

RESOURCE MANAGEMENT IN GRID COMPUTING

INTRODUCTION

Grid computing is based on large-scale resources sharing in a widely connected network such as the Internet (Yan et al., 2009). Grid computing emerged from metacomputing with the introduction of middleware design as a wide-area infrastructure to support data-intensive applications and diverse online processing (Moallem, 2009). At the same time, systems such as Globus Toolkit (Foster & Kesselman, 1997), Storage Resource Broker (Baru et al., 1998), Legion (Grimshaw et al., 1999) and Condor (Frey et al., 2002) were developed to support scientific applications.

Research by Foster and Kesselman (2004) defined that cluster and grid computing are several ways for establishing a distributed system. A distributed system consists of multiple computers that communicate through a computer network. Several personal computers or workstations in cluster computing are combined through local networks in order to develop distributed applications. In cluster computing, applications are inflexible in variation because they are limited to a fixed area. From this disadvantage, grid computing has been proposed as a solution to this problem. Grid computing is developed through a combination of various resources from different geographic locations. This makes grid computing different from conventional distributed computing and cluster computing. However, computational grid has different constraints and requirements compared to the traditional high performance computing systems.

In grid computing system, resource management is the main component that needs to be considered in order to manage all submitted jobs and available resources. There are various issues in grid resource management such as resource discovery, resource scheduling, resource monitoring, resources inventories, resource provisioning, load balancing, fault isolation, autonomic capabilities and service level management system (Sharma & Bawa, 2008). However, grid scheduling and grid load balancing are the main issues that are often discussed by many researchers (Moallem, 2009; Chang et al. 2007; Liu & Wang, 2008)

Grid scheduling is an essential function provided by the grid infrastructure (Kousalya & Balasubramanie, 2008). The scheduling problem in grid computing is classified as Nondeterministic Polynomial (NP) – complete problem (Kousalya & Balasubramanie, 2009) which means that there is no exact algorithm that can solve the problem in a polynomial time. Scalability and adaptability are two main aspects that have to be considered in implementing any scheduling algorithm.

Resources in grid environment are geographically distributed in a large-scale way and resource performance changes from time to time. On the other hand, jobs submitted by the users require resources with different Quality of Service (QoS) requirements. An effective scheduling technique must be defined in order to manage the grid computing environment. Scheduling aims to maximize throughput, minimize computational time (response time) and avoid an overload on certain resources. In order to achieve these aims, the conditions of jobs and resources such as job characteristics and resource capacity must be considered. At the same time, a good scheduling algorithm influences the balancing of workload on each resource.

Scheduling algorithm can be classified as static or dynamic (Moallem, 2009). In the static scheduling algorithm, all information about jobs, resources and communication network are known in advance and jobs are assigned to suitable resources before execution begins. Once started, they keep running on the same resource without interruption. However, static scheduling has one major disadvantage, i.e. all information about jobs and resources remain constant during the process. In contrast, dynamic scheduling attempts to use the runtime

state information to make a scheduling decision more informative. Reevaluation is allowed on the already-taken assignment decisions during job execution in dynamic scheduling algorithm (Chtepen, 2005). However, research by Liu and Wang (2008) stated that static or dynamic scheduling algorithm is categorized by the evaporation rate factor. In static scheduling algorithm, the evaporation rate value remains constant in all situations, while in dynamic scheduling algorithm, evaporation rate value is adaptively changed according to the grid condition. Static scheduling algorithm is easier to implement and has minimal runtime overhead compared to dynamic scheduling algorithm. However, dynamic scheduling may result in better performance. In this research, dynamic scheduling is implemented where the scheduling depends on the changing rate of the evaporation value.

One of the most important problems that need to be handled in grid computing is load balancing. In order to solve this problem, all submitted jobs need to be equally distributed among resources in grid computing system. A good load balancing algorithm must be capable of balancing the entire resources by distributing workload evenly across two or more computers, central processing units (CPUs), network links, hard disk or other resources in order to get optimal resource utilization.

Ant colony optimization (ACO) is one of the most recent algorithms for load balancing in grid computing. It has been used in solving the scheduling problem between the submitted jobs and available resources in grid computing (Fidanova & Durchova, 2006). ACO also has been applied in solving the grid load balancing problem. Research by Chang et al. (2007) and Yan et al. (2005) used the ACO algorithm to solve the load balancing problem. ACO is inspired by a colony of ants that work together in foraging behavior. This behavior encourages ants to find the shortest path between their nest and food source. It is one of the examples of the application of swarm intelligence (Dorigo & Socha, 2006). Swarm intelligence is the field of artificial intelligence that studies the intelligent behavior of groups such as the behavior of natural systems of social insects like ants, bees, termites and wasps. The other example of swarm intelligence is the particle swarm intelligence which studies swarm behavior in fish schooling and bird flocking (Zhan et al., 2009).

There are a number of experiments that have been done by many researchers to study the behavior of real ants like foraging and nest construction. Gross, Aron, Deneubourg, and Pasteels in 1989 (as cited in Dorigo & Stützle, 2004) conducted the double bridge experiment to investigate the foraging behavior of ants. Ants move in a continuous path from the nest to the food source as shown in Figure 1.1(a). In Figure 1.1(b), when an obstacle appears in the way, ants will choose whether to turn left or right with equal probability because they have no clue about which is the best choice or the shorter path as there is no pheromone at that time on both paths. At this time, about 50% of the ants will choose each path. While ants are moving from nest to food source on both paths, they deposit certain amount of pheromone. Ants that have chosen the shorter path will reach the opposite direction faster assuming that all ants walk at approximately the same speed. This means that more ants travelled on the shorter path than those that travelled on the longer path. Therefore, the pheromone will be accumulated more quickly on the shorter path as shown in Figure 1.1(c). The probability of ants choosing the shorter path will be increased with time. After a transitory phase, almost all ants will choose the shorter path due to the large amount of pheromone accumulated on that path as shown in Figure 1.1(d).

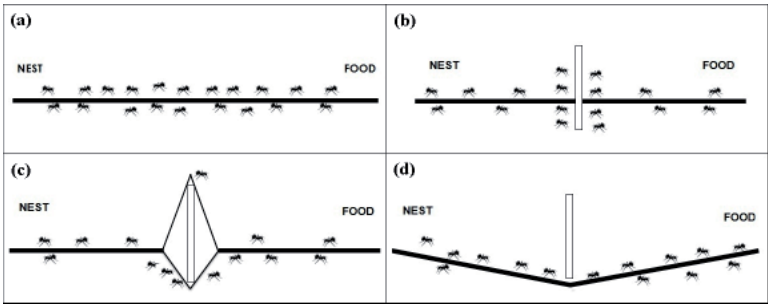

Figure 1.1. Ant behavior in foraging process (Perretto & Lopes, 2005).

Ant Colony System (ACS), Max-Min Ant System (MMAS), Rank-Based Ant System and Elitist Ant System (EAS) (Dorigo & Stützle, 2004) are variants of ACO algorithms. ACO also has been applied to solve many problems in scheduling such as Job Shop Problem, Open

Shop Problem, Permutation Flow Shop Problem, Single Machine Total Tardiness Problem, Single Machine Total Weighted Tardiness Problem, Resource Constraints Project Scheduling Problem, Group Shop Problem and Single Machine Total Tardiness Problem with Sequence Dependent Setup Times (Dorigo & Stützle, 2004).

ACO algorithm is used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems. However, more research work is needed to enhance the performance of ACO algorithms to solve the scheduling and load balancing problems in order to get maximum throughput, minimize response time, avoid overload, minimize stagnation problem and at the same time, balance the entire resources. Stagnation in grid computing may occur if the computational time of the processed job is high. Stagnation also may occur when all jobs are assigned to the same resources which lead to the resources having high workload. The stagnation problem in grid computing will be minimized when all resources are well utilized.

RESEARCH BACKGROUND

In grid computing system, there exists more than one resource to process the submitted jobs. Users will experience delay in response time when the number of jobs increase (Lorpunmanee et al., 2007). This is because the number of available resources is insufficient to cater for all the jobs and there is inefficient assignment of jobs to resources (Fidanova & Durchova, 2006; Wenming et al., 2009). Jobs must be queued to be processed by the available resources. This will lead to stagnation in grid environment. Available resource management algorithm tries to schedule the submitted jobs to available resources as evenly as possible. However, improvements of resource management algorithm to reduce the stagnation problem and to minimize the computational time of each job are still needed.

Scheduling the jobs to the resources in grid computing is also complicated due to the distributed and heterogeneous nature of the resources (Li, 2006). The matching process between jobs and resources is the most important problem that must be handled in grid computing. The resource matching problem involves assigning

jobs to resources in order to satisfy job requirements and resource policies (Tangmunarunkit et al., 2003; Moallem & Ludwig, 2009). The submitted jobs must be matched between the available resources in terms of their job characteristics and resources capacity. Current algorithms (Xu et al., 2003, Chang et al., 2007) have also considered the jobs characteristics and resources capacity in scheduling of jobs but did not take into considerations of Million Instruction per Second (MIPS) and CPU time needed for each job.

Current algorithms in managing the resources have not always consider the load balancing problem (Xu et al., 2003; Ali et al., 2010). As a result, this leads to the increase in computational time because the available resources are not utilized well. The load balancing problem cannot be completely solved with the present resource management algorithms that are based on ACO approach. This is because the pheromone updates technique and resource selection techniques in the present ACO used a fix value for the pheromone evaporation rate (Lorpunmanee et al., 2007; Wenming et al., 2009).

The main objective of the study is to develop an enhanced ant based resource scheduling algorithm which can minimize job computational time, match jobs with suitable resources and balance workload of entire resources in grid environment. Specific objectives of the research are:

(i) To construct a graph model to represent the resource selection strategy that can be used to assign submitted job to resource(s).
(ii) To formulate a formula to calculate an initial pheromone value that can match jobs and resources according to job characteristics and resources capacity.
(iii) To develop pheromone update techniques that can update current status of each resource during scheduling process.
(iv) To simulate the proposed model that can be used to evaluate the proposed algorithm.

Grid computing is emerging as a new computing paradigm to solve the challenging applications in engineering, science and economics. Grid architecture involves the efficient management of distributed, heterogeneous and dynamically available resources. Therefore managing resources is crucial in grid environment.

The outcome of this research contributes to another grid resource management algorithm that can significantly improve the performance of available ACO algorithms. The new algorithm enhances the classical approach of ACO algorithm by dynamically scheduling submitted jobs to suitable resources, thus offering the chance to improve the efficiency of managing resources in grid computing.

The concentration of this research is on improving the way ants search for the best resources and at the same time trying to balance all the workload on available resources. ACO is selected as the based algorithm to be enhanced in solving the dynamic scheduling of resources to jobs. The pheromone update technique in ACS is also adopted and adapted in the proposed algorithm. Single colony of ants is used for searching the best resources to process jobs.

Grid computing has been proposed to provide services through a combination of various resources from different geographic locations. Good resource management approach will enable grid system to be fully utilized. However, there are various issues in grid computing resource management specifically resource discovery, resource scheduling, resource monitoring, resource inventory, resource provisioning, load balancing, fault isolation, autonomic capabilities and service level management system. As such, resource management algorithms are essential for efficient management of the resources.

2

ANT COLONY OPTIMIZATION IN GRID RESOURCE MANAGEMENT

This chapter presents the reviews of research that have been done in the areas of grid computing and ACO. Discussions on grid resource management, grid scheduling and grid load balancing are also included. Previous work on ACO and ant based approach for grid resource scheduling and load balancing will be presented.

GRID RESOURCE MANAGEMENT

Resource management is a central component of a grid computing system. Resource management is the process of managing submitted jobs and available grid resources accordingly. This process includes the resources that are allocated, assigned, authorized, assured and authenticated to process the request jobs (Sharma & Bawa, 2008).

In grid environment, jobs that are submitted by users always have different quality of service requirements or accept best-effort service levels provided by grid system (Krauter et al., 2002). From that point, resource management system is required to handle all submitted jobs in terms of maximizing the quality of service requirements with the aim of distributing workload evenly across two or more computers in order to get optimal resource utilization. In this research, processing time and load balancing aspects are the main quality of service components that have been considered.

Grid Resource Discovery and Matching

Resource discovery and matching involves the process of searching the available resources and scheduling of jobs to suitable resources (Fattahi & Charkari, 2009). Matching is defined as a process of evaluation of the degree of similarity of two objects (Bai et al., 2004). During the scheduling process, jobs needed to be matched with suitable resources that can fulfill their requirements. Grid resource matching is the process of matching between the jobs and resources while taking into account job requirements and available resource capacities and optimizing one or more objective functions (Naik et al., 2006).

There are many types of algorithms that have been used in resource matching in grid computing system. The research by Bai et al. (2004) proposed the framework to solve the resource matching problem. The framework contains a resource specifications component, a request specification components and matchmaking algorithms. A request specification includes a matchmaking function and two additional constraints, a cardinality threshold and matching degree threshold. The cardinality threshold defines how many resources are expected to be returned by the matchmaking service while the matchmaking degree threshold specifies the least matching degree of one resources returned by the service. The matchmaking process executes a matchmaking algorithm for each request that is sent by the requester. Request and grid resource instances that are stored in the knowledge base of the matchmaking service are the input of this algorithm. On the other hand, the output of this algorithm is the number of grid resources ranked according to their matching degree. However, the proposed framework only considers the resource characteristics without taking into considerations the characteristics of each submitted jobs.

The study done by Naik et al. (2006) used the concepts of Qualifying Resource Collection (QRC) in finding the minimal set of resources that need to be assigned to the jobs to satisfy their requirements. The problem was represented by a graph model. QRC specified the resource capacity needed and in which interval is required by the jobs. For a certain number of jobs, there are many QRCs, and it represents all possible assignment of the resources to the job. The relationship between jobs and a QRC indicates that the

requirements and consumptions of the job are met by the resources in the corresponding QRC. The requirement that is needed by each job described the provision a of set of resource dependencies on one or more types of resources. Each dependency put the attribute constraints on the attribute values of the resources of a specific type. A job can also specify optional temporal and location constraints. A job may specify preferences which provide a selection of criteria when multiple resource sets satisfy dependencies related with a job. A job can identify its preference either by providing a method of ordering qualifying resources or by simply identifying specific resource instances by attribute value or by name. The attributes of each job are very important to ensure that job will be processed with the suitable resources that fulfill its requirement. However, the proposed graph model only considers the matching problem between jobs and resources without considering the processing time of each resource.

Semantic based grid resource discovery and its integration with the grid resource broker was proposed by Somasundaram et al. (2006). The research proposed five layered architecture that implements a knowledge layer on top of Gridbus broker which are fabric layer, core middleware layer, high level middleware layer, knowledge layer and application layer. Semantic grid architecture was proposed where knowledge layer was introduced at the top of Gridbus broker architecture that can enable broker to discover resources semantically. On the other hand, the semantic component in the knowledge layer enables semantic description of grid resources with the help of ontology template. Protégé-OWL editor will create the ontology template for different types of computing resources in the grid environment. The Protégé-OWL libraries are used to dynamically create knowledge base of grid resource and Globus Toolkit's MDS is used to gather grid resource information. The research also used Algernon inference engine in interacting with the knowledge base to discover suitable resources. However, the proposed ontology template depends on MDS component and did not support middleware other than Globus.

The study by Li (2006) proposed a bio-inspired adaptive job scheduling mechanism in grid computing. Various software ant agents were designed with simple functionalities. This research proposed the system architecture with five main components of ant

agents namely the queen, scout, tester, worker and cleaner. In this architecture, there is no direct communication among these agents. The only indirect communication is via the pheromone values stored in a grid resource table. To develop this architecture, the queen will produced another agent. The grid resources may become available or unavailable without any notifications. The scout is responsible to find the new grid resources that are providing computational services and adds it to the grid resource table. A tester executes a small sample programme on a grid resource and test for the computational time of the sample programme. According to the job completion time, tester will update the pheromone value of this resource. A worker chooses an available grid resource and runs a computational job on this resource. Resources with higher pheromone value will have a higher probability to be selected. The cleaner will remove the resources that have low pheromone value from the grid resource table. However, the proposed algorithm did not consider the load balancing problem of each resource.

Research by Zhu et al. (2009) proposed the resource discovery method based on the adjacency list and the ant colony algorithm. There are three layers in the proposed resource discovery model which are Peer-To-Peer (P2P) layer, Virtual Organization (VO) layer and resource layer. P2P layer composed of many supernodes where each node represents a super management domain. P2P layer is modified to interacte information between supernodes. When supernodes search resources, the users firstly query the resources in the supernodes of local VO. If there is no query result, Ant colony algorithm is used to search the other supernodes. Mobile agent technology is used in every layer including P2P layer, VO layer, and resource layer. The proposed algorithm used resource adjacency list a centering resource discovery method in the supernode of every VO while Ant Colony algorithm a distributed resource discovery method is used in the middle of supernodes of P2P layer. However, the performance of the proposed method is not used to compare with any other algorithm.

Many researchers proposed the matching algorithms in solving the matching problem between submitted jobs and available resources. However, the improvement of existing algorithms is still needed in order to get a better matching algorithm that can reduce the stagnation problem in grid computing.

Grid Scheduling

Many algorithms have been proposed to solve the grid scheduling problem. The grid scheduling experiment that applies Simulated Annealing (SA) was performed by Yarkhan and Dongarra (2002). The experiments were implemented over a non-homogeneous set of grid resources located at geographically disparate locations. Dynamic machine status and connectivity information were obtained from the Globus Metacomputing Directory Service (MDS) and the Network Weather System (NWS). In many experiments, the SA and Ad-Hoc Scheduler were compared in order to see which scheduler produces better estimated schedules when given the same information. In order to get consistent information, the list of available machines and their characteristics were obtained from MDS and NWS. Experimental results showed that SA is better than Ad-Hoc scheduler in terms of the estimated execution times. This is because SA can avoid some of the local optima that are not anticipated in the Ad-Hoc technique search. In the study, SA algorithm was not verified in a larger experimental environment.

In a study conducted by Carretero and Xhafa (2006), Genetic Algorithms (GA) for job scheduling in grid computing wash done in order to optimize the makespan and total flow time. The researchers aimed to obtain an efficient scheduler that could allocate a large number of jobs to a large number of grid resources. In order to determine which work is better for the problem, several variations of GA operators were examined. A grid simulator package was developed to generate large size instances of the problem which were then used to study the performance of GA implementations. Experimental results showed that the proposed GA algorithm performed well in static scheduling benchmark. However, this study did not consider the use of the simulator over a period of time and the statistical significance during the experiments.

The study by Abraham et al. (2006) proposed a new grid scheduling algorithm based on the particle swarm optimization (PSO) approach. The velocity and position of the particles in the conventional PSO were enhanced from the real vector to fuzzy matrices in order to dynamically generate an optimal schedule. The researchers aimed to complete the tasks within a minimum period of time and also to

utilize the resources in an efficient way. The performance of the proposed algorithm was also compared to the GA and SA algorithms. Experimental results showed that the PSO algorithm was better than GA and SA in terms of speed of convergence and the ability to obtain faster and feasible schedules. However, the experiments only focused on makespan values instead of considering the utilization of each resource.

Grid scheduling algorithm based on the particle swarm was proposed by Chen et al. (2006). The study expressed each possible grid scheduling technique as a task resource assignment graph and mapped the grid scheduling problem into a graph optimal selection problem in order to find an optimal solution quickly and accurately. The proposed scheduling algorithm assumed that the longest path of the task resource assignment graph as a fitness value and encoded every task resource assignment as a particle. The performance of the proposed algorithm was compared to the GA algorithm. Experimental results showed that the proposed PSO algorithm was better than GA algorithm in terms of the makespan and completion time of each job. However, the proposed algorithm did not consider the utilization of each resource in order to balance all the resources.

Wang et al. (2006) proposed a new decentralized grid job scheduling based on hill climbing algorithm. Decentralized job scheduling was implemented by job migration between neighboring grid nodes. Hill climbing algorithm was used to determine the migration route which the job needs to migrate many times in order to optimize node selection of new submitted job. A set of experiments were done in order to simulate a decentralized job scheduling including node adjacencies, local scheduling of grid nodes and grid workload. The performance of the proposed algorithm was compared with k-distributed and auction methods. Experiment results showed that hill climbing scheduling algorithm could enhance the processor utilization and reduce bounded slowdown. However, the proposed algorithm only considered the conditions of resources and not the conditions of submitted jobs such as their makespan and completion time.

Multi-objective evolutionary algorithm for scheduling jobs on computational grid was proposed by Grosan et al. (2007). The proposed algorithm introduced a Multi-Objective Evolutionary Algorithm (MOEA) by using the Pareto dominance as the way to solve the

scheduling problem in grid computing. The researchers aimed to minimize the makespan which is the time when the last task is finished and also to minimize the flowtime of the grid system that minimizes the sum of completion times of all the tasks. The Pareto dominance theory was used in order to compare two solutions, i.e. dominance and non-dominance. Mutation and crossover were used as operators while the binary tournament selection was used in the implementation. The performance of MOEA algorithm was compared to GA, SA, and PSO in terms of their makespan and flowtime aspects. Experimental results showed that MOEA produced excellent results when compared to the other algorithms. However, the proposed algorithm only considered the number of jobs and resources and not the characteristics of jobs and the capacity of resources.

A bio-inspired adaptive job scheduling mechanism on a computational grid was proposed by Li (2006). The researcher aimed to solve the problem of scheduling a set of parallel jobs with different arrival times to run on a computational grid. The proposed bio-inspired scheduling algorithm was inspired by the behavior of the ant colony to effectively utilize the dynamic distributed resources in the grid computing environment in order to achieve an optimal job completion time. The bio-inspired mechanism is similar to the collective behavior of social insects in terms of their strong adaptability and robustness to the dynamic nature of the grid computing environment. The bio-inspired mechanism also designed a tester program that can produce easy-to-verify intermediate values and partial results in order to verify the trustworthiness of the distributed computation grid. The performance of the proposed bio-inspired scheduling mechanism was compared with the random mechanism and heuristic mechanism in terms of their job completion time. Experimental results showed that the proposed ant inspired scheduling algorithm was better than the other algorithms from the adaptability and robustness aspects. However, the bio-inspired scheduling algorithm did not consider the load balancing and the current conditions of each resource during scheduling process.

A grid scheduling algorithm based on ant algorithm which is a Monte Carlo method was proposed by Fidanova and Durchova (2006). The researchers aimed to find a good solution in a reasonable time and also to achieve high throughput computing in grid environment. The proposed algorithm was designed for distributed systems shared

asynchronously by both remote and local users. The heuristic algorithm based on ACO method was developed and its basic strategies for grid scheduling were formulated. The performance of the proposed ant algorithm was compared to the online-mode algorithm in terms of their execution time. Experimental results showed that the proposed ant algorithm was better than online-mode algorithm from the execution time and balancing aspects. However, the proposed algorithm did not consider the requirement of each submitted job and the capacity of available resources.

The study by Lorpunmanee et al. (2007) proposed an ACO for dynamic job scheduling in grid computing. The researchers aimed to develop an effective grid scheduling algorithm that could minimize the total tardiness time of each submitted job. The proposed algorithm was designed to adapt the dynamic grid environment and at the same time improve the overall performance of the system. An optimal resource allocation technique for each job within the dynamic grid environment was developed and tested by using Gridsim toolkit. The performance of the proposed scheduling algorithm was compared with First Come First Serve algorithm, Minimal Tardiness Earliest Due Date algorithm and Minimal Tardiness Earliest Release Date algorithm. Experimental results showed that the proposed ACO algorithm performed better than the other algorithms. However, the proposed algorithm did not consider the balancing of each resource during the scheduling process.

Dynamic grid scheduling algorithm based on self adaptive Tabu Search (TS) was proposed by Kong et al. (2010). The proposed algorithm is suitable for the grid dynamic characteristics in order to reduce the makespan of the submitted jobs. The scheduling process was separated into partial scheduling and last partial information was exploited to decide the next partial scheduling parameters set. During searching process, tabu list kept a local optimal solution and marked it in order to get a simple searching way for these solutions in the future search process. The performance of the proposed tabu search algorithm was compared to several typical algorithms, i.e. Min – min algorithm, Max - min algorithm and Sufferage algorithm. Experimental results showed that the TS algorithm was better than the other algorithms in terms of the makespan value. However, the proposed TS algorithm did not consider the utilization of the resources.

In order to solve the grid scheduling problem, many researchers proposed scheduling algorithms which considered the processing time of each job. Based on previous research discussed above, ACO has proven to be the most promising algorithm that has been successfully used in grid computing to solve scheduling problems which eventually reduces the stagnation problem. However, the load balancing problem is another aspect that should be considered in managing resources in grid computing. Load balancing is the technique to distribute workload between several computers, workstation, CPUs, network links, and other resources in order to get optimal resources utilization, throughput and response (Moallem, 2009).

Grid Load Balancing

Grid load balancing is one of the most difficult problems that must be handled in managing resources. In grid computing environment, load balancing algorithm should be 'fair' in distributing jobs across the resources (Zhu & Hu, 2004). The objectives of the load balancing algorithm are to spread the job equally on each resource, minimize the total task execution time of each job and maximize the utilization of each resource. In order to achieve these objectives, the difference between the heaviest-loaded node and the lightest node should be minimized. The problem of balancing resources is also defined as NP-complete problem (Ibarra & Kim, 1977).

There are many types of algorithms that have been used in resource balancing in grid computing system. The study by Cao et al. (2005) used a combination of intelligent agents and multi-agent approaches that work in grid load balancing area. In static grid load balancing, the iterative heuristic algorithm is better than the First Come First Serve (FCFS) algorithm. The study highlighted that a peer-to-peer service advertisement and discovery technique were more effective in dynamic grid load balancing environment. Instead of using a centralized control, distributed agent could reduce the network overhead significantly and allow the system to operate well in distributed environment which helped the user to achieve good resource utilization and minimize the processing time of each job.

The research by Subrata et al. (2007) addressed the use of Genetic Algorithm (GA) and Tabu Search (TS) to solve the grid load balancing problem in the dynamic environment. In the study, GA and

TS performed better than the Best-Fit, Random, Min-min, Max-min and Sufferage algorithms in terms of time taken to schedule submitted jobs and job completion time. GA and TS could balance the extra overhead by considering the ever decreasing costs of storage and processing power. However, these algorithms required extra storage and processing requirement at the scheduling nodes.

Balanced Ant Colony Optimization (BACO) algorithm proposed by Chang et al. (2007) chooses optimal resources to process jobs based on resource status and size of submitted job. The researchers aimed to balance the entire resources and at the same time minimize the makespan of the jobs in grid computing system. The performance of the BACO algorithm was compared with the improved Ant Colony Optimization algorithm, Fastest Processor to Largest Task First algorithm, and random algorithms. Experimental results showed that the BACO algorithm performed better than the other algorithms in terms of standard deviation and makespan. However, BACO algorithm did not consider the capacity of each resource during the scheduling process.

Rose et al. (2008) proposed the allocation strategies for utilization of space shared resources in bag of tasks grids. In this research, an adaptor automatically fits grid requests to the resource in order to decrease turn-around time of application. The jobs from the user are received by the grid broker. The request adaptor receives the grid broker requirements and tries to provide workers by the submission of space shared requests crafted by heuristics. Each processor can run several tasks during the requested time but only one at a given moment. In order to choose the suitable parameters for requests, the request adaptor should obtain some information about the space shared resource state, space shared resource scheduler administrative policies and the grid application. The performance of the proposed allocation strategies was compared with Transparent Allocation Strategy (Netto et al., 2005). Experimental results showed that the proposed strategy enable users to natively submit tasks without having to submit their requirement. However, the proposed strategy only can process one job at a time and does not consider the jobs requirement.

A hybrid load balancing strategy of sequential tasks that uses a combination of static and dynamic load balancing strategies which combines a FCFS algorithm with a special designed GA was proposed

by Li et al. (2009). The FCFS algorithm can make decisions instantly which can reduce the system's response time, resulting in a shorter makespan. GA was used to control the overall performance over a list of tasks and target the balance of the resources in grid computing area. A sliding-window technique was used to trigger the switch between the two algorithms and to make a rapid task assignment as well. From the experiment conducted, it was found that hybrid GA provided better performance than dynamic GA and FCFS in different conditions such as makespan and the current work load. Besides, the proposed strategy did not consider the requirement of each submitted job and the capacity of available resources.

The study by Sadhasivam and Meenakshi (2009) proposed a load balanced, efficient scheduling with parallel job submission in computational grids using Parallel Particle Swarm Optimization (PPSO). The researchers aimed to maximize the speed of completion of processes, minimize the communication overhead, enhance resource utilization, and parallel efficiency. PPSO approach is used to group the jobs and to submit them in parallel to available grid resources. In order to improve the job submission time and to ensure security, the trust based parallel job submission was also proposed. PPSO groups the jobs based on resource utilization and trust level of the users/resources. All information about jobs and resources such as the total number of jobs, processing requirement of each job, trust of each job, total number of available resources, processing capabilities of each resource and the granularity time was determined in order to optimize the utilization of the resources and also to minimize the job execution time. The groups of jobs were then submitted in parallel to the resources in the grid environment. The performance of PPSO in terms of its simulation time and resource utilization was compared with job grouping scheduling framework using PSO. Experimental results showed that PPSO was better than classical PSO from both aspects.

The use of artificial life technique for distributed grid job scheduling was proposed by Moallem and Ludwig (2009). The research proposed two distributed artificial life inspired load balancing algorithms based on ACO algorithm and PSO algorithm. The researchers aimed to minimize the processing time of the submitted jobs and also to balance the entire resources. The performance of the proposed algorithm was

compared with the random approach. Experimental results showed that the proposed algorithm worked well in distributing the submitted jobs to resources. However, the ant algorithm did not schedule jobs well to a small number of resources.

Adaptive hierarchical scheduling policy for enterprise grid computing systems was proposed by Abawajy (2009). This research proposed an approach by combinding both time sharing and space sharing policy. An adaptive hierarchical scheduling (AHS) policy was introduced with special attention to input/output and service demands of parallel jobs in homogeneous and heterogeneous systems with background workload. In order to assign resource to parallel jobs, AHS integrates affinity scheduling, job assignment, and self scheduling approach into a framework. The performance of the proposed algorithm was compared with static space-time sharing policy. Experimental results showed that the proposed algorithm performs better than the static space-time sharing policy in term of arrival time and utilization. However, AHS policy did not consider the processing time of each job.

A decentralized Recent Neighbor (RN) load balancing algorithm for computational grid was proposed by Balasangameshwara and Raju (2010). The researchers aimed to solve the grid load balancing problem by assigning loads in grid system without neglecting the communication overhead in collecting the load information. RN algorithm performs intra-cluster and inter-cluster load balancing in dynamic grid environment. Experimental results showed that RN was better in terms of its response time and resource utilization when compared to the other algorithms. However, the proposed algorithm only considered the processing power of resources without considering the other aspects such as the bandwidth and current load of resources that can affect the performance of the algorithm.

Many researchers proposed the load balancing algorithms which consider the utilization of each resource in order to solve the grid load balancing problem. Based on the previous research discussed above, ACO has proven to be the most promising algorithm that has been successfully used in solving the load balancing problem. However, the improvement of existing algorithms is still needed in order to get a better load balancing algorithm than can reduce the stagnation problem in grid computing.

ANT COLONY OPTIMIZATION

ACO is a biologically inspired algorithm that can provide the user with an opportunity to solve optimization problem and design the meta-heuristics algorithm (Dorigo & Stützle, 2004). This algorithm is a new evolutionary approach where several ants work together to search for a good solution. Every ant builds up a solution step by step by its own decision points until a complete solution is found. Ants put an amount of pheromone on the edges of the path to mark their solution (paths). The strength of pheromone is used to build the solution. The next ant will be attracted by the pheromone, so it will search for the solution.

The main part of the ACO is the use of a combination of priori information (heuristics) and posteriori information (pheromone) (Dorigo & Stützle, 2004). Priori information is about the quality of candidate solutions (called greedy strategy) while posteriori information is about the goodness of the previously obtained solution (called positive feedback or autocatalytic process). ACO not only uses heuristics to create a solution but also uses the accumulated experiences about obtaining good solutions in the previous process.

Ant System (AS) is the first member of among several of the well known ACO algorithms to be introduced and the prototype of a number of ant algorithms. It was initially proposed by Colorni et al. (1991) and Dorigo (1992) and it aimed to search for an optimal path in a graph based on the behavior of ants seeking a path between their colony and a source of food. AS is also the first ACO algorithm which was applied to the Traveling Salesman Problem (TSP) (Dorigo et al., 1996). Three different versions of ant system were proposed, i.e. ant-density, ant-quantity and ant-cycle. In ant-density and ant-quantity, the ants update the pheromone directly after a move from a city to an adjacent city. But in ant-cycle, the pheromone update was only done after all the ants had constructed the tours. The two main phases of the AS algorithm constitute the ants' solution construction and the pheromone update. The performance of AS when compared to other algorithms tends to decrease dramatically as the size of the test-instances increases.

ACS proposed by Dorigo and Gambardella (1997a, 1997b) is the first extension of AS to improve its performance. ACS differs in three main aspects from the ant system. Firstly, ACS uses a more aggressive action choice rule as compared to AS. Secondly, the pheromone is added only to arcs belonging to the global-best solution. Thirdly, each time an ant uses an arc (i,j) to move from city i to city j, it removes some pheromone from the arc. In ACS, ants choose the next city using the *pseudo-random-proportional action choice rule*. This probabilistic rule is a trade-off between exploration and exploitation. Exploration gives the chance to add new edge to the solution. Exploitation uses the accumulated information from previous iterations preferring the choice of an edge with maximum combination of pheromone trails and heuristic values. An ant with probability q exploits the available information about previous good solutions or with probability $(1- q)$ explores new areas of the solution space focusing on shorter edges with pheromone rate. In ACS, only the global best ant is allowed to add pheromone after each iteration. The trail update only applies to the arcs of the global-best tour, not to all the arcs like in AS. Only the global best solution receives feedback. Additionally, with regards to the global updating rule, in ACS, the ants use a local update rule that they apply immediately after having crossed an arc during the tour construction. The effect of the local updating rule is to make an already chosen arc less desirable for a following ant. In this way, the exploration of yet to be visited arcs is increased.

Bullnheimer et al. (1996) proposed ranked AS (AS_{rank}) as an extension of $AS_{elitist}$ proposed by Dorigo et al. (1996). In AS_{rank}, the global best tour is always used to update the pheromone trails, similar to the elitist strategy of AS. Additionally, a number of the best ants of the current iteration are allowed to add pheromone. To this end, the ants are sorted by tour length and the quantity of pheromone an ant may deposit is weighted according to the rank r of the ant. Only the $(w-1)$ best ants of each iteration are allowed to deposit pheromone. The global best solution which gives the strongest feedback is given weight, w. The rth best ant of the current iteration contributes to pheromone updating with a weight given by $\max\{0, w-r\}$. Among the AS-based algorithms, both AS_{rank} and $AS_{elitist}$ performed significantly better than AS, with AS_{rank} giving slightly better results than $AS_{elitist.}$

Max-Min AS (MMAS) was proposed by Stützle and Hoos (2000) as another improvement over AS-based algorithm and it has showed a

higher performance than other ACO algorithms for TSP. The solutions in MMAS are constructed in the same way as in AS. Additionally, MMAS uses the pseudo-random-proportional action choice rule of ACS. Using that action choice rule, very good solution could be found faster, but the final solution quality achieved was worse. The main modifications in MMAS with respect to AS are the following aspects: 1) to exploit the best solution found, after each iteration, only one ant is allowed to add pheromone; 2) to avoid search stagnation, the allowed range of the pheromone trail strengths is limited to the interval $[T_{min,} T_{max}]$; 3) the pheromone trails are initialized to the upper trail limit which causes a higher exploration at the start of the algorithm; 4) the pheromone trails are updated after all ants have constructed a solution; and 5) the ant which is allowed to add pheromone may be the global best solution or iteration best solution. Therefore, if the same arcs are often used in the best solutions, it will receive a larger amount of pheromone. The lower and upper limits on the possible pheromone strengths on any arc are imposed in MMAS to avoid search stagnation. This trail limit has the effect of indirectly limiting the probability T_{ij} of selecting a city j when an ant is in city i to an interval $[T_{min,} T_{max}]$ with $0< T_{min} \leq T_{ij} \leq T_{max} \leq 1$. The pheromone trails in MMAS are initialized to their upper pheromone trail limits.

Best Worst AS (BWAS) was proposed by Cordon et al. (2000) as another extension of the basic idea of AS by including some concepts from evolutionary computation algorithms. BWAS is similar to AS (same transition rule) in constructing ants' solutions. BWAS enhances the ants' solution by using local optimizer to bring each solution to its local optimum. There are basically three core activities performed offline by daemon actions. First, daemon actions perform positive and negative pheromone updates by reinforcing the edges of *global best solution* through the addition of an amount of pheromone proportional to the quality of this solution. Furthermore, daemon actions penalize the edges belonging to the *worst solution* obtained from the current iteration and do not share in the global best solution by evaporating extra amount of pheromone. Second, like MMAS, a mechanism to avoid stagnation is incorporated in BWAS. Stagnation usually takes place when there are big differences between the pheromone trails of edges of the best solutions (very high) and the pheromone trails of other edges (very small). In such situations, BWAS considers restarting the search process by reinitializing all pheromone trails to an initial value. Third, to encourage the exploration of new areas of the solution space,

a mutation operation is applied on the pheromone trails by performing small changes in early iterations (no chance of stagnation) and strong changes in latter iteration when there is a higher chance of stagnation. The mutation range depends on the average of pheromone trails on the edges of the global best solution. Cordon et al. (2002) conducted an analysis of the above three components, considering each component separately as well as different combination of them on the TSP and quality of service access point. Their analysis studies showed that BWAS with the above three components gave much better solutions than BWAS with one or two of these components.

Kaegi and White (2003) proposed Local Best Tour Ant System (LBTAS) as a new version of AS which considered the use of local information to guide the ants' search process. The basic change was that each ant updates the pheromone trails according to its own best tour from the beginning of the algorithm. This modification avoids the use of global information by observing all ants' tours and selects the best global one as in ACS and MMAS. In LBTAS, each ant works individually on a copy of AS and indirectly cooperates with other ants. The early results of applying LBTAS on some versions of TSP were promising when compared with original AS and $AS_{elitist}$. Kaegi and White (2003) were of the opinion that adding local search procedure to LBTAS improves its chance to outperform other versions of AS.

ANT BASED GRID SCHEDULING ALGORITHM

In a grid computing system, when a job is submitted, it needs to be processed by the available resources. Best resources in terms of processing speed, memory and availability status are more likely to be selected for the submitted jobs during the scheduling process (Lorpunmanee et al., 2007). The best resources are categorized as optimal resources. In a research by Li (2006), ACO has been used as an effective algorithm in solving the scheduling problem in grid computing. The process undertaken by ACO considers the value of pheromone, a chemical substance used for indirect communications between the ants for resource selection.

Simple grid simulation architecture for resource management and dynamic grid scheduling was proposed in Xu et al. (2003). This study also validated the scalability of ant algorithm. The ant algorithm for

grid task scheduling was integrated into the simulation architecture. The initial pheromone value was calculated based on the number of processing elements (PEs), MIPS, size of job, and transfer time. A set of experiments was conducted to see the performance of the proposed algorithm. Good results were obtained in terms of resource average utilization and response time.

The study to improve ant algorithm for dynamic job scheduling in grid computing which is based on the basic idea of ACS was proposed by Yan et al. (2005). The pheromone update function in this research was performed by adding encouragement, punishment coefficient and load balancing factor. The initial pheromone value of each resource was based on its status where job was assigned to the resource with the maximum pheromone value. The strength of pheromone of each resource was updated after completion of the job. The encouragement and punishment and local balancing factor coefficient were defined by users and were used to update pheromone values of resources. If a resource completed a job successfully, more pheromone was added by the encouragement coefficient in order to be selected for the next job execution. If a resource failed to complete a job, it was punished by adding less pheromone value. The load of each resource was taken into account and the balancing factor was also applied to change the pheromone value of each resource.

The study by Li (2006) proposed a bio-inspired adaptive job scheduling mechanism in static grid computing. The purpose of this research was to minimize the execution time of the computational jobs by effectively taking advantage of the large amount of distributed resource. Various software ant agents were designed with simple functionalities. The pheromone update function was done based on the execution of each resource. In the research, comparison was made between the bio inspired adaptive scheduling with the random mechanism and heuristic mechanism. Experimental results showed that a bio-inspired adaptive job scheduling had good adaptability and robustness in a dynamic computational grid.

Ant algorithm to solve static grid scheduling problem was proposed by Fidanova and Durchova (2006). The researchers aimed to find a good scheduling algorithm that can minimize the processing time of the jobs. In this proposed algorithm, the scheduler allocated submitted

jobs to available resources based on the prediction of the computing power of the resource. There were two types of mapping heuristics which are online mode and batch mode. The proposed algorithm was based on batch mode as the scheduler considers a meta-task for matching and scheduling at each mapping event. The pheromone value was calculated based on the evaporation rate value and the function free factor which is the time when the machine will be free. The performance of the proposed algorithm was compared with the online mode algorithm in terms of their execution time. Experimental results showed that the proposed algorithm performed better than online mode algorithm from the execution time aspects. However, there was no pheromone update function in this proposed algorithm. The job was only submitted if the resource is fully free to be used. This affected the execution time of the algorithm.

For dynamic job scheduling in grid environment, an ACO based algorithm was proposed by Lorpunmanee et al. (2007) which aimed to minimize the total job tardiness time. The process to update the pheromone value on each resource was based on local update and global update rules as in ACS. The performance of the proposed algorithm was compared with existing algorithms which are First Come First Serve, Minimal Tardiness Earliest Due Date and Minimal Tardiness Earliest Release Date algorithms. Experimental results showed that the proposed algorithms performed better than the other algorithms because it considered the load of each resource during the resource selection process.

The dynamic grid scheduling algorithm based on adaptive ant colony algorithm was proposed by Liu and Wang (2008). The aim of the research was to minimize the searching time and avoid the stagnation problem in grid computing system. In the algorithm, the evaporation rate value was adaptively changed and a minimum value for evaporation rate was assigned. The evaporation rate used by the algorithm was under control and was never reduced to 0. The local and global pheromone updates were used in order to control the pheromone value of each resource. The performance of the proposed algorithm was compared with the basic ant colony algorithm. Experimental results showed that an adaptive ant colony algorithm performed better than the basic ant colony algorithm in terms of the searching time.

An improved ant algorithm for static grid scheduling problem was proposed by Bagherzadeh and MadadyarAdeh (2009). The researchers aimed to minimize the processing time of jobs and to balance the entire resources in grid computing system. The proposed ant algorithm was based on batch mode, where jobs and resources were collected and mapped at prescheduled time. The pheromone update mechanism was done at the end of the iteration instead of the selection process. This allows faster convergence of the proposed algorithm to the optimal solution. The performance of the proposed algorithm was compared to the existing algorithms which are Opportunistic Load Balancing, Minimum Execution Time, Minimum Completion Time, Switching Algorithm, K-Percent Best, MinMin, MaxMin, MaxStd, Dupplex, and previous ACO. Experimental results showed that the proposed ant algorithm performed better than the other algorithms in terms of makespan and utilization.

The research by Wenming et al. (2009) proposed the trust based ACO for dynamic grid resource scheduling. The researchers aimed to minimize the completion time of jobs and utilization of resources. The local pheromone update and global pheromone update were used in the algorithm in order to achieve the load balance system by incorporating resource oriented trust mechanism. Local pheromone update would reduce the pheromone value on the path, thus freeing the ant to explore the new path that is to be used. The global pheromone update was done by updating the pheromone value on the shortest path after the task had been finished. From those points, the optimal solution could be obtained and the system load balancing could be achieved. The performance of the trust based ant colony algorithm was compared with MinMin algorithm. Experimental results showed that the trust based ant algorithm performed better than MinMin algorithm in terms of completion time.

From the above research, ACS is found to be the most popular variant of ACO that has been successfully used in grid computing to solve the scheduling problems which eventually reduced the stagnation problem. However, more work is needed to enhance the performance of the algorithm in this application domain.

ANT BASED GRID LOAD BALANCING ALGORITHM

A study by Salehi and Deldari (2006) proposed a new dynamic algorithm that is based on an echo intelligent system, autonomous and cooperative ants. Ant level load balancing was proposed to improve the performance of the mechanism. In this proposed algorithm, the ants can procreate and also can commit suicide depending on the existing condition. Ants were created on demand to achieve load balancing during their adaptive lives. The ants may bear offspring when they detected the system is drastically unbalanced and commit suicide when they detect equilibrium in the environment. The ants will care for every node visited during their steps and record node specifications for future decision making. Theoretical and simulation results indicated that this new algorithm surpasses its predecessor. However, the pheromone values were not updated in this proposed algorithm which enabled the assignment of jobs to the same resource. Therefore, stagnation occurred in the grid computing system.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by Chang et al. (2007). The research aimed to minimize the computation time of job executing in Taiwan‘s UniGrid environment which also focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chose optimal resources to process the submitted jobs. The local and global pheromone update techniques were used to balance the system load. Local pheromone update function updated the status of the selected resource after a job had been assigned and the job scheduler depends on the latest information of the selected resource for the next job submission. The global pheromone update function updated the status of each resource for all jobs after the completion of the jobs. By using these two update techniques, the job scheduler will obtained the latest information of all resources for the next job submission. From the experimental results, BACO was capable of balancing the entire system load regardless of the size of the jobs in the static scheduling benchmark.

An enhanced ant algorithm for dynamic task scheduling in grid computing was proposed by Sathish and Reddy (2008) which gave better throughput with a controlled cost. The proposed scheduling algorithm increased the performance in terms of low processing time

and low processing cost when applied to a grid application with a large number of jobs such as the parameter sweeps application. This algorithm worked effectively in minimizing the processing time and processing cost of the jobs. The simulation results of various scheduling algorithm such as the modified ant algorithm and the cost controlled algorithm were also compared. The results showed that this enhanced algorithm worked better than the ant algorithm. By considering the processing cost, this enhanced ant algorithm was more suitable for a wide use. However, this algorithm did not consider the size of the jobs which leads to appropriate assignment of jobs to resources.

Load balancing in non-dedicated grids using ACO was proposed by Chen (2008). The proposed static algorithm was based on ACO algorithm in solving the load balancing problem in grid computing system. In the algorithm, the efficiency of the resources was maintained by immigrating jobs from overloaded resources to under loaded resources. The inherent features of a non dedicated grid computing system, such as dynamics and heterogeneity, were embedded in the model while the pheromone update function was done according to the Gauss Function and the evaporation rate. The performance of the proposed algorithm was compared with the First In First Out algorithm, the Tabu algorithm, and the Tabu + Granularity-based Job Allocation Policy algorithm. Experimental results showed that the proposed algorithm performed better than the other algorithms in terms of makespan and resource usage. However, the proposed algorithm did not consider the requirement of each submitted jobs and the capacity of resources.

In Moallem and Ludwig (2009), two distributed artificial life-inspired algorithms were introduced and they are ACO and PSO in solving the static grid load balancing problem. Distributed load balancing are categorized as a robust algorithm that can adapt to any topology changes in a network. In the proposed algorithm, an ant acted as a broker to find the best node in terms of the pheromone value stored in the pheromone table. The node with the lightest load was selected as the best node. The position of each node in the flock could be determined by its load in PSO. The particle compared the load of nodes with its neighbours and moved towards the best neighbour by sending assigned jobs to it. The proposed algorithm performed better than ACO for job scheduling where jobs were submitted from different

sources and different time intervals. PSO showed better results than ACO in terms of the makespan. However, PSO used more bandwidth and communication compared to ACO. The main drawback of the Ant Colony was that jobs are not scheduled efficiently and therefore load among the resources were not balanced. This problem can be fixed by increasing the number of ants that can explore the entire grid system to find resources with the lightest load.

ACO algorithm for dynamic load balancing in distributed systems through the use of multiple ant colonies was proposed by Ali et al. (2010). In this algorithm, information on resources was dynamically updated at each ant movement. Load balancing system was based on multiple ant colonies information. Multiple ant colonies were adopted in order that each node sent a colored colony throughout the network. Coloured ant colonies were used to prevent ants of the same nest from following the same route and also forcing them to be distributed all over the nodes in the system. Each ant acted like a mobile agent which carried newly updated load balancing information to the next node. This proposed algorithm was compared to the work-stealing approach for load balancing in grid computing. Experimental results showed that multiple ant colonies worked better than work-stealing algorithm in terms of their efficiency. However, the multiple ant colonies did not consider resources capacity and jobs characteristics. This can make matching the jobs with the best resources a difficult task for the scheduling algorithm.

Based on the previous research discussed above, it is found that many researchers have used an ACO approach in solving the grid load balancing problem. The pheromone update function is applied in order to manage the pheromone value of each resource during selection process. However, more work is needed to enhance the performance of the algorithm in this application domain.

SUMMARY

Resource scheduling is the process of managing submitted jobs to available resources in grid computing system. The scheduling algorithm must consider the characteristics of each job and the capacity of resources in scheduling the submitted jobs to the available

resources. In current situations, many effective algorithms are applied in order to solve the grid resource scheduling problem. The most promising technique that has been used is the ACO technique. ACO technique can solve the scheduling problem, stagnation problem and minimize the computational time in grid computing environment. The local pheromone update and the global pheromone update are used to update the pheromone value of each resource.

3

METHODOLOGY AND PROPOSED FRAMEWORK

This chapter presents the methodology and proposed framework that have been used in this research. The activities of the research methodology will be discussed in detail followed by the description of the proposed framework.

RESEARCH METHODOLOGY

The experimental research methodology that has been used by a large number of ACO researchers like Dorigo et al. (1991a, 1991b, 1996) and Gambardella (1997a, 1997b), Stützle and Hoos (2000) and Kawamura et al. (2000), has been chosen to be used in this research. There are five steps in this methodology (refer Figure 3.1) which are analyzing the research problem, developing the proposed framework, constructing the simulation environment, conducting a set of experiments and evaluating the results. This methodology is adapted because it suits the proposed algorithm, provides a good output, and is easy to use in solving the grid resource management problem.

Analyzing the Research Problem

The first step in this research is to analyze available ACO algorithms applied in grid computing and to determine existing problems. This include cases where the ACO algorithms only considered the computational time of each processed job without considering the matching problem between job requirement and resource capacity

which leads to stagnation problem in grid computing. These problems were captured when studying the mechanisms used by existing ACO algorithms to control the activities of ants namely the resource selection mechanism and the pheromone updating mechanism. These mechanisms influence the ant's decision making process and are used to organize the attraction of ants toward the previously obtained solution. Thus, stagnation in the grid computing will be minimized when good resource selection and pheromone updating mechanisms can reduce the attraction of ants to a single solution.

The research problem is determined at the end of this step, i.e. after the stagnation in grid computing was clearly defined and the available approach that attempts to reduce this problem was identified. The initial pheromone value and the global pheromone update were modified to get a better scheduling result and which, at the same time, could balance the entire resources in grid computing system.

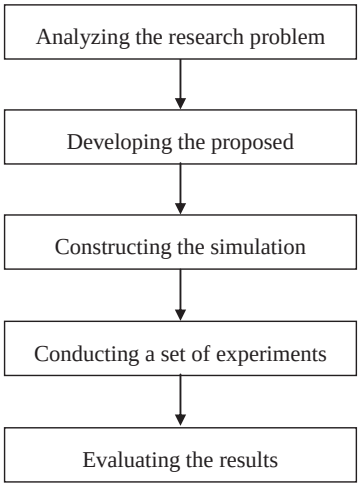

Figure 3.1. Steps of experimental research methodology.

Developing the Proposed Framework

The proposed framework is developed in this step. Details of the framework will be discussed in Chapter 4.

Constructing the Simulation Environment

A computer simulation was developed for the proposed algorithm and the environment to be applied. Full implementation of the proposed algorithm, ant algorithm by Moallem (2009), Particle Swarm (Moallem and Ludwig, 2009), Space Share and Time Share were carried out using Java programming under Gridsim toolkit. The computational time of each job and the load of each resource were measured during the simulation process.

Conducting the Experiment

A set of experiments wase conducted. These experiments were used to test the performance of the proposed algorithm. Tests were carried out on job processing time and utilization of resources in grid computing. The objectives of the experiments are to evaluate the performance of the proposed algorithm in term of processing time and utilization of each resource and comparing the results with existing algorithms. A set of experiments was also conducted to determine how different values of evaporation rate will affect the performance of the proposed algorithm. In all the experiments, the number of machine per resource is 1 and the number of PEs per machine is randomly chosen from 1 to 5. The PE ratings and bandwidth for each machine will vary.

Four experiments have been conducted in order to evaluate the performance of the proposed algorithm in term of processing time and utilization. Three experiments were conducted in order to evaluate the performance of the proposed algorithm in the load balancing aspect. Various numbers of resources and jobs have been used in all the experiments.

Evaluating the Results

Performance of the proposed algorithm during different experiments was reported and compared with the performance of other available algorithms. This process is important to show the relative strength and weakness of the proposed algorithm. The result of this proposed algorithm was compared with ant algorithm by Moallem (2009) and other algorithms such as Particle Swarm (Moallem & Ludwig,

2009), Space Shared (Rose et al., 2008), and Time Shared (Abawajy, 2009). The results of these algorithms were taken from the literature wherever possible; otherwise they were taken from the implementation developed in this research.

The comparisons presented in this report are designed so that all algorithms run exactly the same parameters. This gives a fair judgment of the results of these algorithms. Thus, the comparison of performances gives an indication on how successful the organization of ant's population is in the proposed algorithm and how effective the mechanisms are being incorporated.

THE PROPOSED FRAMEWORK

The main aim of the research is to develop enhancement of the ant colony algorithm that can balance the load among resources in the grid system. The proposed algorithm hereafter will be known as enhanced ant colony optimization (EACO). There are three mechanisms in the framework that are used to organize the work of ant colony to form the proposed algorithm. The mechanisms are initial pheromone value mechanism, resource selection mechanism and pheromone updating mechanism The initial pheromone value mechanism will solve the matching problem between submitted jobs and the available resources while the resource selection mechanism will solve the scheduling problem where by the best resource will be selected to process jobs. Pheromone updating mechanism will solve the load balancing problem of each resource. Figure 3.2 shows the basic components of this framework.

Initial Pheromone Value Mechanism

An initial pheromone value is calculated after each job enters the grid information system. The effect of the initial pheromone value is to match all submitted jobs to suitable resources. This value is calculated by considering the jobs characteristics and the capacity of resources. The initial pheromone value of each resource for each job is calculated based on the estimated transmission time and the execution time of a given job when assigned to this resource.

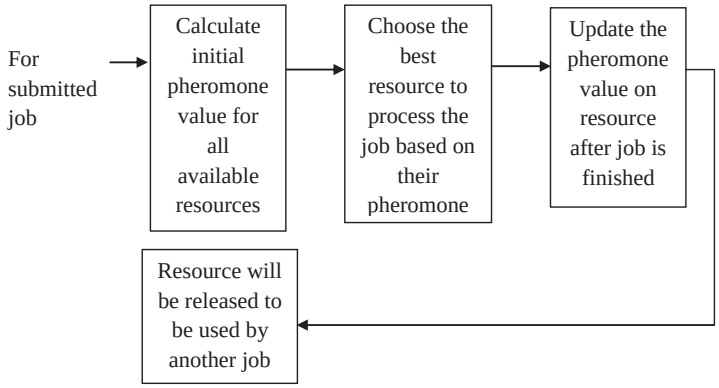

Figure 3.2. The proposed EACO framework.

Resource Selection Mechanism

When an ant wants to move to another resource, it must make a probabilistic decision to select a suitable resource. The probabilistic decision is based on heuristic information (computational time) and pheromone information. Pheromone represents information about previous experiences of the ant while heuristic represents a priori information about the goodness of a solution. In this situation, ants use the exploration control mechanism when they explore new resource in the grid environment. On the other hand, ants that want to use the previously selected resources will use the exploitation control mechanism.

Pheromone Updating Mechanism

The proposed pheromone updating mechanism encourages a balanced form of exploitation of previous experiences and exploration of new nodes. The global pheromone updating mechanism has been used in encouraging the exploration of new areas of the search space by reducing the importance of the visited nodes. That was used in this proposed algorithm. In global pheromone updating mechanism, the best ant will deposit an amount of pheromone on its own nodes. The best ant refers to the ant that obtains the best solution in the current iteration of the algorithm execution or the ant that obtains the best solution since the start of the algorithm execution. The global

pheromone update concludes that each ant reduces the amount of pheromone on nodes that it has visited to give more chance to other nodes to be chosen by the next ants.

SUMMARY

The proposed framework which consists of three mechanisms has been used to organize the work of ant colony in the resource scheduling algorithm with the aim to balance the load of all resources. The initial pheromone value is calculated by considering the jobs characteristics and the capacity of resources. The initial pheromone mechanism will solve the matching problem between the submitted jobs and the available resources.

The resource selection mechanism will be performed when ants try to search for a new solution. However, this should be done under certain control to avoid exploring a very wide area of search space that might be far from the optimal solution. On the other hand, an exploitation of the search history is necessary to search for previous good solution. However, very strong exploitation is not required to prevent the stagnation problem of certain resources. The resource selection mechanism will solve the scheduling problem in grid computing.

The global pheromone update mechanism will be performed after all jobs are completely processed. The best ant will deposit an amount of pheromone on its own nodes during the global pheromone update mechanism. The pheromone updating mechanism will solve the load balancing problem of each resource.

4

ENHANCEMENT OF ANT COLONY OPTIMIZATION ALGORITHM

This chapter presents the proposed enhanced ant based grid resource scheduling algorithm called EACO. The proposed algorithm takes into consideration the capacity of resources and the characteristics of jobs in determining the best resource to process a job. The grid resource scheduling scenario is discussed in detail, followed by the description of the proposed algorithm. The enhancement of the ant colony algorithm design and implementation are also presented.

GRID RESOURCE SCHEDULING SCENARIO

In a distributed system, there might be issues in which a job waits for a service at the queue of one resource, while at the same time another resource which is capable of processing a job is idle. The purpose of a resource scheduling algorithm is to prevent these problems from occuring as much as possible (Livny & Melman, 1981).

Information collection, decision making and data migration are three phases in grid resource scheduling process. In the collection phase, the grid broker collects all information of each resource and detects whether there is a load imbalance among resources. Optimal job distribution is calculated during the decision making process while an exact amount of jobs is transferred to other suitable resources during the job migration process.

Grid scheduling are divided into three classes of architecture and they are centralized, decentralized and hierarchical. In centralize approach, all jobs are submitted to a single scheduler that is responsible to schedule all jobs to the available resources. This method is optimal to use since all information are available at one place but this method is not very scalable in grid computing because bottleneck problems occur when the scheduler tries to keep all information on the state of the resources. Thus, scalability is a problem in centralized method that influences a single point of failure to the system.

There is no central scheduler in decentralized scheduling method and scheduling is implemented by the resource requestors and owners independently. This scheduling method is scalable in grid computing system and is suitable for peer-to-peer architectures and dynamic environments. In this method, individual scheduler must cooperate with each other in making scheduling decisions. The proposed EACO algorithm is based on decentralized scheduling method.

In hierarchical scheduling method, the schedulers are organized in a hierarchy with resources with high level entities being scheduled at higher levels and resources with low level entities being scheduled at lower levels. This method is a combination of centralized and decentralized scheduling methods.

ENHANCED ANT COLONY OPTIMIZATION

EACO is developed by integrating the idea of how ants cluster the objects. Figure 4.1 shows behaviour of ants in clustering the objects. Ant will move randomly until it encounters an object. The ant will disregard this object if it is carrying another object, or will pick it up and will continue on its way. It can be seen that each ant seems to cooperate in piling up dead corpses in the nest. This proposed algorithm uses the inspiration of how ants are able to cluster objects and tries to use it in the inverse version to spread jobs in the grid system. The ants try to distribute as many jobs as possible rather than piling them.

The proposed algorithm is inspired by a colony of ants that works together to find the shortest path between their nest and food source. Every ant will deposit a chemical substance called pheromone on the ground after they move from the nest to food sources and vice versa.

Therefore, they will choose the shortest or optimal path based on the pheromone value. The path with high pheromone value is shorter than the path with low pheromone value. This behavior is the basis for a cooperative communication.

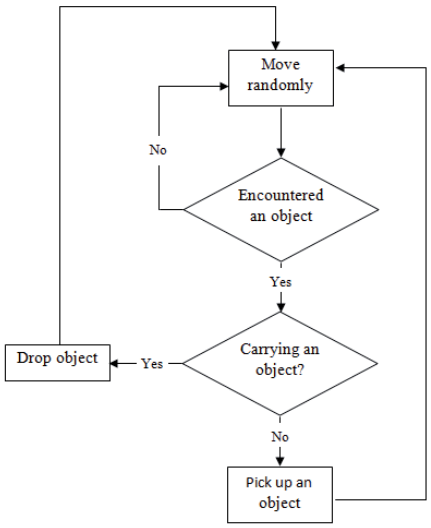

Figure 4.1. Ant behaviour of clustering the objects.

EACO Graph Model

The EACO graph model was developed in order to manage the resources in grid computing system (refer Figure 4.2). This directed graph model consists of a set of jobs with requirement associated with it and resources with their capacity. Jobs are submitted by different users from different geographic locations. Each job has its own characteristics and requirements that need to be satisfied by available resources in grid system such as their size and CPU time needed for each job. There are many types of resources such as application, database, printer, and server. Resources are distributed in different geographic area and owned by different owners with their own rules. Each resource type has its static and dynamic attribute. For example, the static attributes for the resource type server are host name, CPU speed and CPU architecture. On the other hand, the dynamic attributes for resource type server are current CPU load, memory usage and availability status.

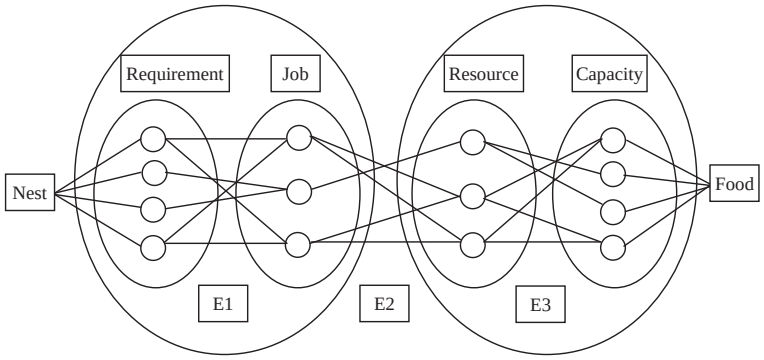

Figure 4.2. EACO Graph Model.

As can be seen from the graph model, there are four types of vertices that are related to each other. Due to that special kind of relation, the graph can be seen as three sub- graphs, each of which is a bipartite graph related to each other:

- Sub-graph E1, a bipartite graph that connects Job and Requirement, E1 = {Job, Requirement}.
- Sub-graph E2, a bipartite graph that connects Resource and Capacity, E2 = {Resource, Capacity}.
- Sub-graph E3, a bipartite graph that connects Job and Resource, E3 = {Job, Resource}.

By using the computational model as described above, the main aim of grid resource management process can be accomplished by choosing appropriate path from job to resource. The task in this model is perfectly matched with the task done in the Simple AS algorithm (Colorni et al. 1991; Dorigo, 1992), where ants must go back and forth between food and nest, passing more preferred path, and then deciding which path should be chosen.

By using the simple AS algorithm, the grid system element that is represented by sequence of vertices in the path between nest and food will be selected based on the amount of pheromone deposited by the ant that moves from nest to food. Therefore, before the path selection is performed, there should be a process to construct the pheromone trail.

The construction of pheromone trail is done by moving the ants from nest to food. The pheromone value of each job to be processed by each resource will be calculated. Resource with high pheromone value will be selected to process the submitted jobs. If the job is finished, the ant will update the pheromone value and the resource will be released to be used by another job.

Proposed EACO Algorithm

The proposed algorithm, EACO, is inspired by a colony of ants that work together in foraging behavior. The EACO takes into consideration the job requirements and resources capacity in determining the best resource to process a job. The EACO algorithm selects the resources based on the pheromone value on each resource which is recorded on a pheromone value table (PVT). The proposed EACO algorithm consists of 5 steps namely obtain job requirements, create an ant for a job, calculate the initial pheromone value and store in PVT for all resources, assign resource with highest pheromone value in PVT to the job, and perform global pheromone update after complete processing the job. The pseudo code of EACO algorithm is shown in Figure 4.3.

```
Get number of resources from num_resource
Get number of jobs from num_gridlet
Compute evaporation rate value;
    evap_rate = calculateEvapRate(num_resource, num_gridlet);
For each num_resource:
    Bandwidth = getBandwidth();
    MIPS = getMIPS();
    load = getLoad();
    createResource(id, bandwidth, MIPS, load);
For each num_gridlet:
    job_size = getJobSize();
    cpu_time = getCPUTime();
    createGridlet();
    createAnt();
Calculate PV for each resource and job combination, and store into pv_table

While process_iteration < num_gridlet

    For (R = 0; R < num_resource; R++)
        For (J = 0; J < num_gridlet; J++)
            getHighestPV();
        process (R, J);
        global_update(evap_rate, R, J);
    process_iteration++;
```

Figure 4.3. Pseudo code of the EACO algorithm.

Jobs are submitted by different users to the grid environment. Submitted jobs are independent of each other and contain different requirements. For each job, the scheduler will record details the size of the job and CPU time needed by the job.

An ant which is to represent a job in the grid system is created for every job that is submitted to the system. The task for the ant is to move from one resource to another with the aim to evaluate the best resource to be assigned to the job. The calculation of the initial pheromone value is presented in the third step.

The initial pheromone value is calculated by considering the job requirements and resource capacity. The ant that represents a job will move from one resource to another to calculate the pheromone value. Pheromone value on a resource indicates the capacity of each resource in grid system. The initial pheromone value of each resource for each job is calculated based on the estimated transmission time and execution time of a given job when assigned to this resource. The estimated transmission time can be determined by $\frac{S_j}{bandwidth_r}$ where S_j is the size of a given job *j* and is the bandwidth available between the grid resource broker and the resource. The initial pheromone value is defined by:

$$PV_{rj} = \left[\frac{S_j}{bandwidth_r} + \frac{C_j}{MIPS_r * (1 - load_r)} \right]^{-1} \tag{4.1}$$

where PV_j is the pheromone value for job *j* assigned to resource *r*, C_j is the CPU time needed of job *j*, $MIPS_r$ is the processor speed of resource *r* and 1-*load* is the current load of resource *r*. The load, processor speed and bandwidth can be obtained from grid information server. Pheromone value will be stored on the PVT as a reference to the other ants.

The ant decides which resource to choose in its next step by looking at the PVT. Assume that there are *n* jobs and *m* resources in PVT as shown below:

$$PV = \begin{array}{c} \\ r_1 \\ r_2 \\ .. \\ r_m \end{array} \begin{array}{c} \begin{array}{cccc} j_1 & j_2 & .. & j_n \end{array} \\ \begin{bmatrix} PV_{11} & PV_{12} & .. & PV_{1n} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ PV_{m1} & PV_{m2} & .. & PV_{mn} \end{bmatrix} \end{array}$$

The largest entry from PVT will be selected in each iteration. The job will be assigned and processed by the resource represented by the largest entry in PVT.

In the final step of the algorithm, the global pheromone update is performed to recalculate the entire PVM when a job is completely processed. The global pheromone update is adapted from the ACS algorithm that has been proposed by Dorigo and Gambardella (1997a, 1997b). After all ants have constructed a solution, the pheromone value is updated according to the following formula:

$$PV_{rj}(t+1) = (1-\rho).\tau_{rj} + 1/(\rho \Delta \tau_{rj}^{bs}) \tag{4.2}$$

where $\Delta \tau_{rj}^{bs} = 1/L^{best}$ and ρ is the evaporation rate value that adaptively change with grid condition. Many researchers used the static evaporation rate value which is 0.5 while a dynamic evaporation rate changer was proposed in this research by considering the size of the submitted jobs and the available resources. Increasing the ratio of the resources to the jobs will increase the evaporation rate of the EACO algorithm while decreasing the ratio of the resources to the jobs will decrease the evaporation rate of the EACO algorithm. The evaporation rate value is defined by

$$\rho = \left[(R/J) * 0.45^{n} * (J/R)\right] + 0.05 \tag{4.3}$$

where R is the number of resource, J is the number of job and n is defined by

$$n = \sqrt[3]{(J/R) - 1} \tag{4.4}$$

The dynamic value of the evaporation rate will ensure that the ant will move faster as the number of job increases. The ant which is allowed to add pheromone may be the *iteration-best solution* or *global best solution.* If a specific resource is often used in the best solution, it will

receive a larger amount of pheromone and stagnation will occur. The effect of the global pheromone update is to make an already chosen resource less desirable for the following ant (Dorigo et al., 1991b). So, the exploration of the not yet visited resource is increased. Once the job is finished, the resource will be released to be used by the other jobs.

ENHANCED ACO DESIGN AND IMPLEMENTATION

The proposed algorithm was implemented in the grid simulation toolkit, called Gridsim toolkit. Gridsim was selected because it is one of the current and complete frameworks for simulating the grid environment. Gridsim toolkit is a Java based toolkit that supports simulation and modelling of heterogeneous Grid resources, users and application models. Additionally, Gridsim toolkit also provides the service to create application jobs, thr mapping of jobs to resource and their management. In Gridsim toolkit, researchers can integrate the scheduling algorithm to be used in managing resources in grid computing system. In addition, Gridsim toolkit supports modelling of heterogeneous types of resources and resources can be modelled as space shared or time shared mode. Resources can be mapped in any time zone and at the same time weekend and holidays can be located depending on resource's local time to model non-Grid workload. Resource can be booked for advance reservation and resource capability can be defined.

Application that runs in Gridsim toolkit can be simulated with different parallel application. Application can also be CPU or I/O intensive and can be heterogeneous. There is no limit to the number of jobs that can be submitted to a resource and multiple users can submit jobs for execution simultaneously in the same resource. Static and dynamic schedulers are supported by Gridsim toolkit and network speed between resources can be determined. Statistics of all operations can also be recorded and can be analyzed using the Gridsim statistics analysis methods. In the study, EACO was implemented in the Gridsim toolkit in order to solve the dynamic grid resource management problem.

There are several steps suggested by Gridsim team (Buyya & Murshed, 2002) in order to simulate a grid scheduling algorithm using Gridsim toolkit. The steps are as follows:

i. Create resources with different capabilities and configurations, for example single or multiprocessor, time shared or space shared resource manager, connections links and speed, etc.
ii. Create users with different requirements and characteristics. Each user can submit jobs (Gridlets) at different intervals with different characteristics.
iii. Create a user entity that creates and interacts with the grid resource broker entity to coordinate an experiment. It can also directly interact with the resource entity and grid information service entity in order to get the grid information and submitting or receiving processed jobs. On the other hand, the implementation of a separate resource broker entity is encouraged.
iv. Implement a grid resource broker entity that performs application scheduling on resources. To do this, based on time, for example, access the grid information service, and then inquire about the resource capabilities including time. Depending on the processing requirements, a schedule is developed for assigning Gridlets to resources and coordinating the execution.

The AllocPolicy class in the Gridsim toolkit must be overridden in order to implement the EACO algorithm. In the design specifications of Gridsim toolkit, each resource is attached to the allocation policy. The enhancement is made to adapt the idea of the proposed scheduling algorithm. Figure 4.4 depicts the UML class diagram of the design. As can be seen in the figure, the EACO algorithm is inheriting the AllocPolicy class. There is also a class called MyGridSimulator which extends the Gridsim class in the Gridsim toolkit that creates all resources and submits the jobs to the grid. The characteristics of each resource in the grid system are provided in the Resource Characteristics class.

Each resource was created and initialized with a specific scheduling algorithm. Jobs are sent to the grid computing system and they are delivered to their resources according to the scheduling algorithm defined for the system. Resources and jobs can be created by using different parameters according to the simulation needs.

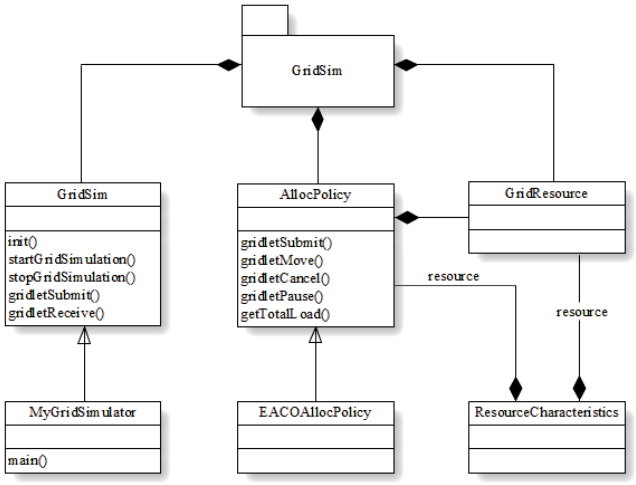

Figure 4.4. The UML class diagram of the design.

By extending the AllocPolicy class, the AntColonyAllocPolicy has been created which tries to select the best resources to process the job. It can be implemented by coordinating jobs in one resource together within one class. Figure 4.5 shows the class diagram of AntColony AllocPolicy which is inherited from the AllocPolicy class in the Gridsim toolkit.

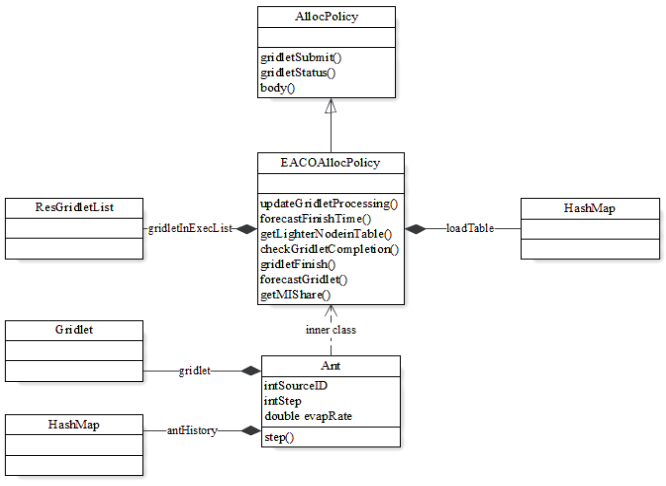

Figure 4.5. The UML class diagram of the EACO algorithm.

In this proposed algorithm, Round Robin scheduling policy is adopted to manage jobs which are assigned to one resource. Whenever a job is submitted to a resource, it uses a Round Robin policy to execute it inside the resource. In a Round Robin policy, jobs in the executing list get an equal time stamp to execute in a resource. In order to do this, there is a list containing executing jobs which is (gridletInExecList) in the AntColonyAllocPolicy class. There is also PVT which is filled by visiting ants and their pheromone value. Class Ant is an inner class of AntColonyAllocPolicy which controls the movement of the ant from one resource to the other resource. All ants have a small memory to carry a history of the visited resource and also the gridlet that it is scheduling.

Figure 4.6 depicts a sequence diagram of simulation process which shows how the EACO scheduling works. A step by step explanation of the simulation process is as follows:

1. MyGridSimulator is responsible for simulating the jobs which are submitted to the grid system. When MyGridSimulator sends a job to a grid system, the job will be sent to its scheduling policy.
2. AntColonyAllocPolicy will create a new Ant object in response to receiving a job and sends it out to explore the grid system to find the best resource to process the job.
3. The ant moves in the grid system by calculating the pheromone value of the visited resources and stores it to the pvList. The ant decides which resource will process a job by reading the pvList information.
4. The ants decides which resource to choose either by looking at the previous resource that was used to process jobs or by choosing another resource by referring to the pheromone value associated with it. This is to prevent the ant from getting caught in local minimum. The ant needs to send a request to GridInformationService which contains the information about the resources in the grid in order to choose a random resource.
5. When the ant finds a suitable resource, them the job will be processed using the function processGridletSubmission().
6. When a job is completed, the global pheromone update is performed to recalculate the pheromone value of the resource using the function updatePV (global_update).

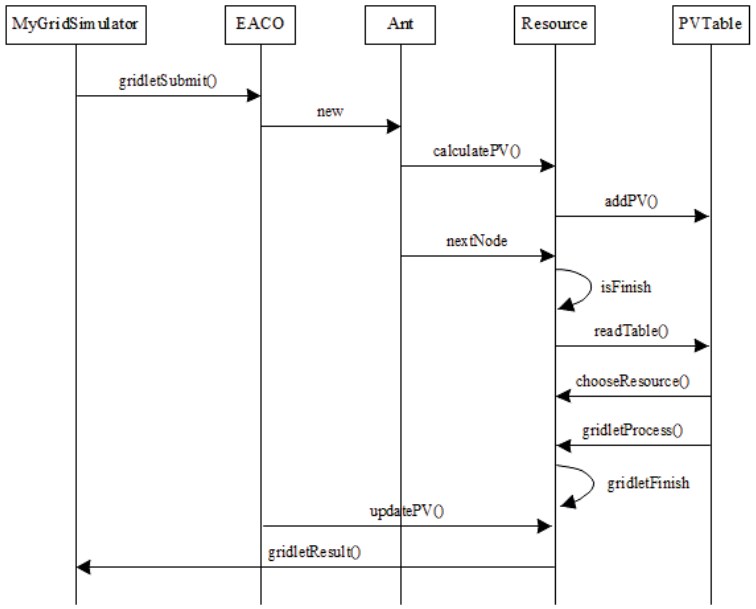

Figure 4.6. UML sequence diagram of the EACO algorithm.

SUMMARY

One of the main difficulties in selecting the resource is when the requirement of each job is uniform. The proposed EACO takes into consideration the capacity of the resources and characteristics of the jobs in determining the best resource to process a job. EACO selects the the resources based on the pheromone value on each resource which is recorded in a matrix form. The initial pheromone value of each resource for each job is calculated based on the estimated transmission time and execution time of a given job if it is assigned to the resource. Resources with high pheromone value are selected to process the submitted jobs. The global pheromone update is done after the jobs have completely being processed with the aim to reduce the pheromone value of the resources.

5

ENHANCED ANT COLONY OPTIMIZATION ALGORITHM

This chapter presents the experimental results of EACO and several existing grid scheduling algorithms in terms of their processing time and their utilization of each resource. Details of the system model that focuses on how the environment setting is chosen and how the grid system is constructed will be presented, followed by the application model of the grid system. The characteristics of the jobs that are submitted to the grid system, and the performance evaluation criteria that are used to evaluate the performance are also explained. Lastly, experimental results and analysis are discussed.

SYSTEM MODEL

In the grid computing environment, there are a set of resources that are connected via different communication networks with different speeds. Each resource may have one or multiple numbers of machines and each machine may have single or multiple processing elements. The speed of a processor or computational power is defined by the number of cycles per unit time. As the processors in each machine can be heterogeneous, so, they may have different processing power.

In the experiments that were conducted, each resource is assumed to consist of one machine and each machine may have one or several processors. The processors in the same or different machines can consist of different processing power. A machine in the grid system may also

have a local user that uses the machine for other computations. From that point, onward, at any one time, a machine may have background workload associated with it. It will affect the computational time of jobs assigned to it. In order to solve this problem, the Gridsim toolkit provides the users with the ability to define the background workload according to the historical and statistical information for each machine. Each resource has an associated background load that is taken from the average load that the resource has experienced at similar times (such as weekends or working days).

APPLICATION MODEL

In order to develop an application model, it is assumed that the applications which are being run or the jobs which are submitted to the grid system consist of a set of independent jobs with no particular order of execution. Jobs that are submitted consist of different computational time, therefore each job will require a different data transmission time and computation time for completing.

The length of each job is presented in MIPS and each job has different input and output size requirements. Jobs in the grid computing system can be classified into one of the two categories namely computational intensive task or data intensive task. This research focuses on computationally intensive tasks as it is more common in today's real life applications and the waste of computational power of resources is more costly than their memory (Moallem, 2009).

PERFORMANCE EVALUATION CRITERIA

The performance evaluation criteria that are used to evaluate the performance of the proposed algorithm are the processing time and utilization of each resource. Minimizing the variations in workloads on all machines is one of the aims of a load balancing algorithm. Standard deviation in workload distribution is often used to determine the performance and stability of the algorithm. A good load balancing scheme is indicated by a small standard deviation value. Standard deviation was calculated using equation 5.1 where x is the resource utilization and n is the size of resource

$$SD = \sqrt{\frac{\sum(x-\overline{x})^2}{(n-1)}} \quad (5.1)$$

Processing Time

Processing time of each algorithm is one of the most common measures in order to evaluate the performance of a scheduling algorithm. The processing time is the total application execution time which is measured from the time the job is sent to the grid system until the job comes out of the grid. As jobs and topologies are generated randomly, every simulation will roughly yield different results. In order to get better results, the average processing time of 10 runs is recorded. Equation 5.2 shows how the processing time of each submitted job is calculated.

$$Processing\ Time = Finish\ Time - Submission\ Time \quad (5.2)$$

Utilization

The utilization of each resource in the grid system is dependent on the time to process all jobs which are assigned to the machine by the grid scheduler and the total time to process all the jobs in the system. The utilization of each resource can be calculated using the equation 5.3.

$$utilization(\%) = \frac{total\ \ busy\ \ time}{total\ \ time\ \ processing\ \ all\ \ jobs} * 100 \quad (5.3)$$

where *total busy time* is the time each resource consumes to process all assigned jobs and *total time processing all the jobs* is the total time taken for all the jobs to be processed by all resources

EXPERIMENTAL DESIGN

A set of experiments was conducted in order to evaluate the performance of EACO. The performance of EACO algorithm was compared with ant based algorithm that was proposed by Moallem (2009), PSO algorithm (Moallem & Ludwig (2009), Space Shared algorithm (Rose et al., 2008) and Time Shared algorithm (Abawajy,

2009) in terms of processing time and resource utilization. In this study, the ant based algorithm by Moallem (2009) is referred to as AntZ algorithm. PSO and AntZ have been implemented in Gridsim. The results from these algorithms were compared with the ones in Moallem (2009) in the process of validating. The Space Shared algorithm and the Time Shared algorithm were already provided by the Gridsim toolkit.

The characteristics of the resources are shown in Table 5.1. Each resource has one machine and each machine has a random number of PEs ranging between 1 and 5. Each PE has a different processing power.

Table 5.1

Resource Characteristics

Number of machines per resource	*1*
Number of PEs per machine	*1-5*
PE ratings	*10 or 50 MIPS*
Bandwidth	*1000 or 5000 B/S*

Jobs which are submitted to the grid system are supposed to be independent of each other. Table 5.2 shows the characteristics of the submitted jobs in order to compare the processing time of each algorithms.

Table 5.2

Jobs Characteristics

Length	0 – 50000 MI
File Size	100 + (10% to 40%)
Output Size	250 + (10% to 50%)

EXPERIMENTAL RESULTS

Results of the experiments were compared with the AntZ algorithm, the Space Shared algorithm and the Time Shared algorithm in terms of processing time and the utilization of each resource.

Processing Time

Figure 5.1 depicts a comparison between the processing times of the EACO algorithms with the AntZ algorithm, the PSO algorithm, the Space Shared algorithm and the Time Shared algorithm with the parameter specifications described in Table 5.3. Experimental results showed that EACO outperformed the others. This is expected as the EACO algorithm keeps track of the state of all resources at each point in time which makes it able to make more optimal decisions at any time. The Particle Swarm algorithm has the smallest processing time after the EACO algorithm followed by the AntZ algorithm, the Time Shared algorithm, and the Space Shared algorithm.

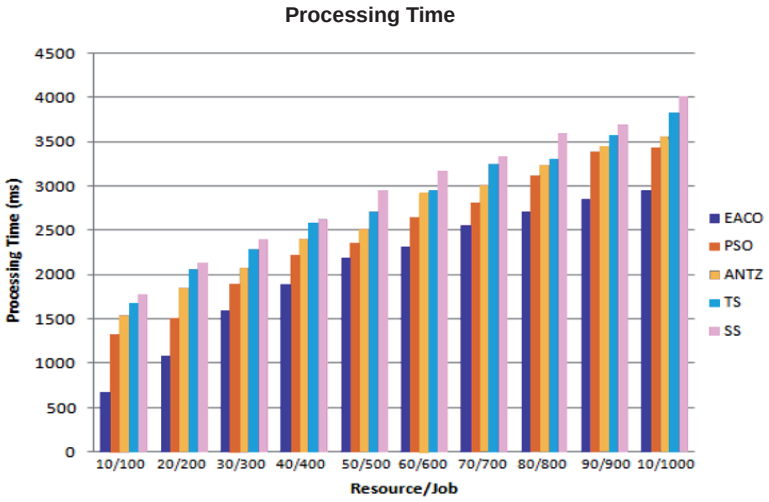

Figure 5.1. Processing time for different number of jobs and resources.

Table 5.3

Experimental Setting for Different Number of Jobs and Resources

Experiment	No. of Jobs	No. of Resources
1	100	10
2	200	20
3	300	30

(continued)

Experiment	No. of Jobs	No. of Resources
4	400	40
5	500	50
6	600	60
7	700	70
8	800	80
9	900	90
10	1000	100

The effects of submitting the same number of jobs into the grid system that has the same number of resources was investigated in the next experiment. The specifications and parameter settings that were used in the experiment are listed in Table 5.4. Figure 5.4 depicts the performance of EACO and the other algorithms in comparing the processing time for the same number of jobs and resources. Again it can be seen that the EACO algorithm performed better than the other algorithms.

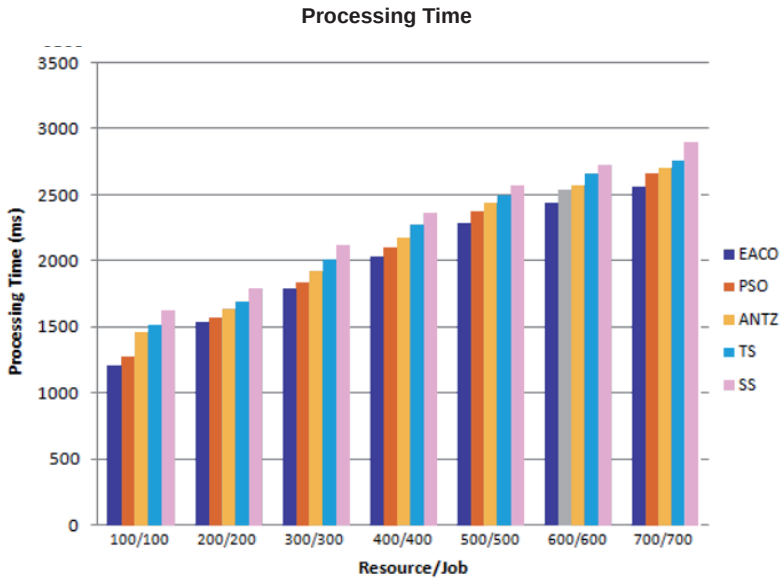

Figure 5.2. Processing time for equal number of jobs and resources.

Table 5.4

Experimental Setting for the Same Number of Jobs and Resources

Experiment	No. of Jobs	No. of Resources
1	100	100
2	200	200
3	300	300
4	400	400
5	500	500
6	600	600
7	700	700

Figure 5.3 shows how increasing the number of jobs, while having the same number of resources that are available in grid system will affect the performance of the grid in terms of processing time. In this experiment, 10 resources are available in the grid system with varying number of jobs. The specifications and parameter settings are listed in Table 5.5. As can be seen, all the algorithms show a linear growth in response to the increasing number of jobs. Experimental results showed that the EACO algorithm had smoother growth compared to the other algorithms in terms of processing time.

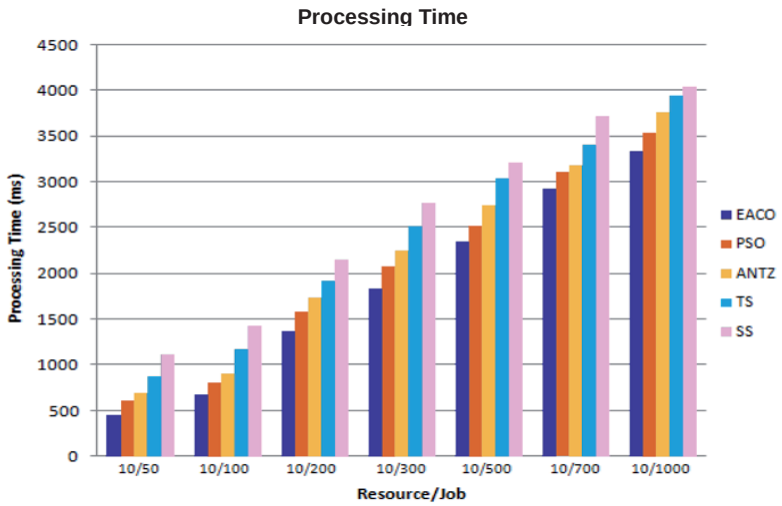

Figure 5.3. Processing time for 10 resources with different number of jobs.

Table 5.5

Experimental Settings for 10 Resources with Different Number of Jobs

Experiment	No. of Jobs	No. of Resources
1	50	10
2	100	10
3	200	10
4	300	10
5	500	10
6	700	10
7	1000	10

In the next experiment, the effect of increasing the number of resources on the performance of the algorithms was investigated. In this experiment, a fixed number of jobs were sent to the grid system while the number of resources that were available to process a job was increased. The specifications and parameter settings are listed in Table 5.6. As can be seen in Figure 5.4, increasing the number of resources has a decreasing exponential effect on the processing time. The EACO algorithm performed better than the other algorithms in response to the increasing number of resources.

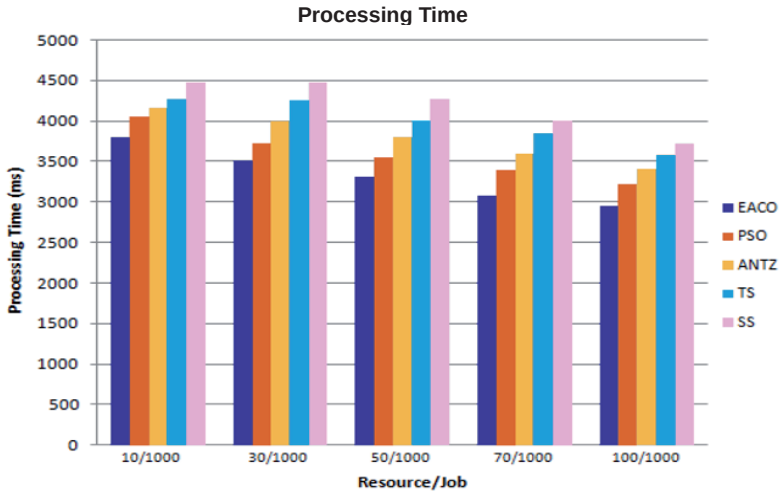

Figure 5.4. Processing time for the same number of jobs with different number of resources.

Table 5.6

Experimental Setting for the Same Number of Jobs with Different Number of Resources

Experiment	No. of Jobs	No. of Resources
1	1000	10
2	1000	30
3	1000	50
4	1000	70
5	1000	100

Utilization

Experiments were also conducted to see whether jobs were distributed evenly among the resources. For this purpose, the utilization of the resources was measured. Figure 5.5 shows the utilization of 10 resources in processing 100 jobs. The mean utilization for the 10 resources is 10%. The utilization of 7 resources is within 1 standard deviation away from the mean. This shows that the EACO algorithm successfully scheduled the jobs among the resources which led to a balanced load network.

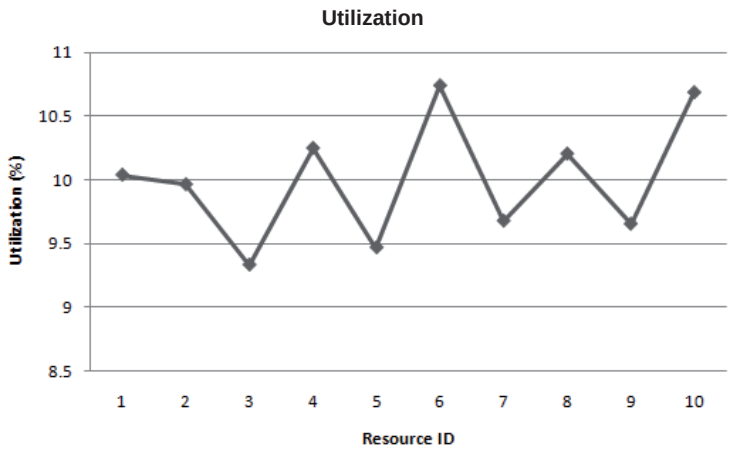

Figure 5.5. Utilization of 10 resources in processing 100 jobs for EACO algorithm.

A set of experiments was also conducted in order to compare the utilization of each resource between the EACO algorithm and the AntZ algorithm (refer Figure 5.6). Experimental results showed that the EACO algorithm performed better than the AntZ algorithm. The standard deviation for the EACO algorithm is 0.48109736 while the standard deviation for the AntZ algorithm is 3.89609944. This is expected as the EACO algorithm is keeping track of the state of all the resources at each point in time which makes it able to make more optimal decisions at each point in time.

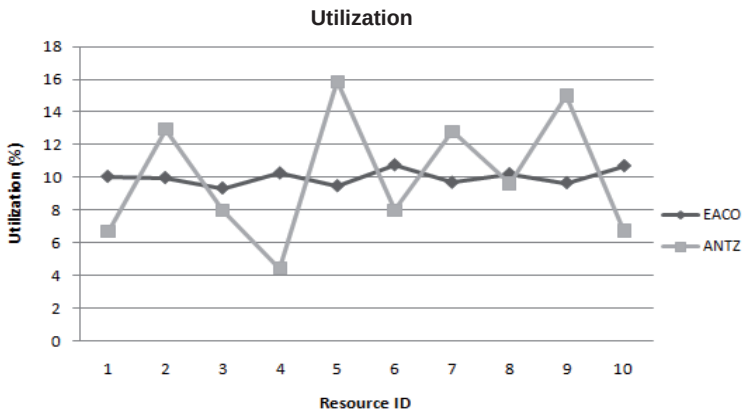

Figure 5.6. Utilization of 10 resources in processing 100 jobs for EACO and AntZ algorithms.

A set of experiments was conducted to see the utilization of 10 resources in processing 500 jobs (refer Figure 5.7). The mean utilization for the 10 resources is 10%. The utilization of 8 resources is within 1 standard deviation away from the mean. This showed that the EACO algorithm successfully balanced the load among the resources.

Experiments was also conducted to compare the utilization of each resource between the EACO algorithm and the AntZ algorithm in order to process 500 jobs with 10 resources (refer Figure 5.8). The standard deviation for the EACO algorithm is 0.24781421 while the standard deviation for the AntZ algorithm is 3.6131422.These results also proved that the EACO algorithm performed better than the AntZ algorithm. Utilization of each resource was investigated for 1000 jobs (refer Figure 5.9). The results showed that the mean utilization for

the 10 resources is 10. Seven (7) resources have utilization within 1 standard deviation away from the mean. This showed that the loads/ jobs were better scheduled.

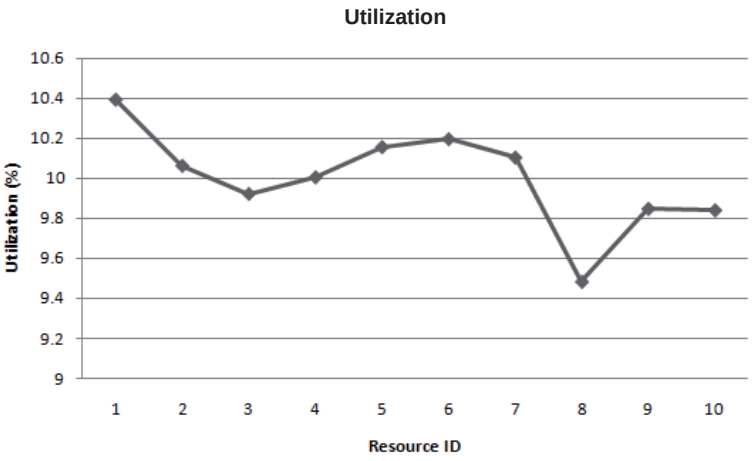

Figure 5.7. Utilization of 10 resources in processing 500 jobs for EACO algorithm.

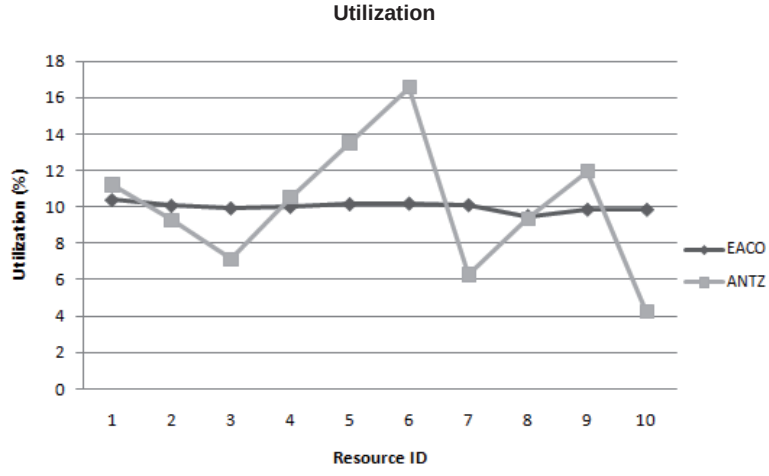

Figure 5.8. Utilization of 10 resources in processing 500 jobs for EACO and AntZ algorithms.

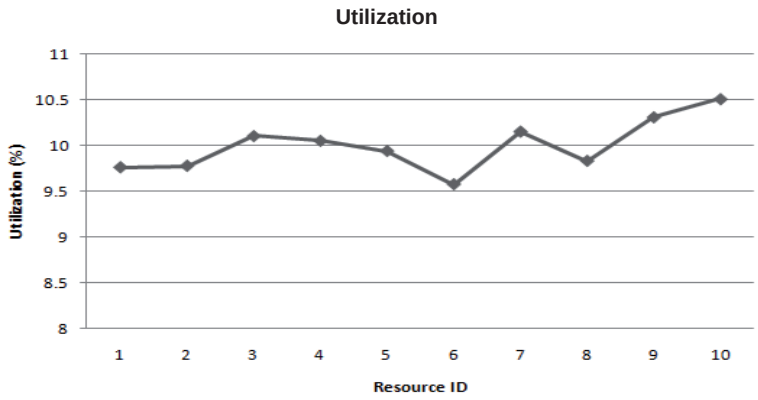

Figure 5.9. Utilization of 10 resources in processing 10 jobs for EACO algorithm.

Comparison on utilization of each resource between the EACO algorithm and the AntZ algorithm in processing 1000 jobs with 10 resources was also conducted. Experimental results showed that the EACO algorithm was better than the AntZ algorithm as the standard deviation for the EACO algorithm is 0.28084636 and the standard deviation for the AntZ algorithm is 2.77428296 (refer Figure 5.10). These results showed that the EACO algorithm successfully balanced the load among the resources in a large scale manner.

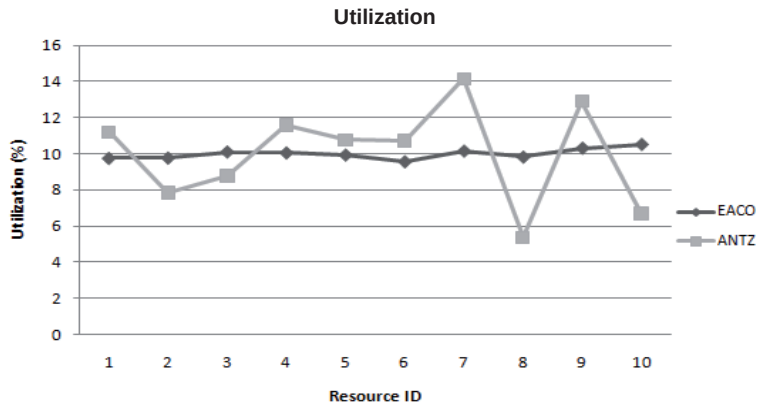

Figure 5.10. Utilization of 10 resources in processing 1000 jobs For EACO and AntZ algorithms.

SUMMARY

In this chapter, the experimental results of applying EACO with preferred values for different control parameters is compared with an existing grid resource management algorithms in terms of processing time and utilization of each resource. All algorithms ran exactly the same scheduling parameters such as the number of jobs, the number of resources, the number of machines per resource, the number of PEs per machine, the PE ratings, the bandwidth, the size of jobs, and the CPU time needed by each job.

Experimental results showed that EACO performed better than the other algorithms in terms of processing time. In all conditions, EACO showed a lower processing time compared to the other algorithms. This is expected as the EACO algorithm is keeping track of the state of all resources at each point in time which makes it able to make more optimal decisions at each time. These results proved that EACO is promising in solving the matching and scheduling problems in grid computing.

The performance of the proposed algorithm was also investigated in the load balancing aspect. Experimental results showed that the EACO algorithm performed better than the AntZ algorithm, the Space Shared algorithm and the Time Shared algorithm in terms of utilization of each resource. The EACO algorithm has successfully scheduled the jobs among resource in all conditions which leads to a balanced load network.

6

CONCLUSION AND FUTURE WORK

EACO offers the opportunity to enhance the results of the ACO algorithms reported in the literature. The results of EACO showed that this approach can be superior to the best known ACO algorithms like ACS and MMAS. The enhancement of the ant based resource scheduling algorithm in grid computing was able to minimize job computational time, match jobs with suitable resources, and balance the resources in grid environment.

The research has considered:

1. A directed graph model that consists of four vertices which are job, requirement, resource and capacity. The graph model reflects the behavior of ants that must go back and forth between the food and nest in foraging for food.
2. A formula to calculate an initial pheromone value that can match jobs and resources according to the jobs characteristics and the resources capacity. The initial pheromone value considered the size and CPU time needed by jobs and also considered the bandwidth, the MIPS and the current load of resources. This value will be stored in PVT as a reference to the following ants.
3. A resource selection strategy that can be used to assign submitted jobs to resources. Ants can decide which resource to choose either by looking at the previous resource that was used to process jobs or choose a resource randomly by the probability of mutation factor in this strategy.

4. A pheromone update technique that can be used to update current status of each resource during scheduling process. The calculation pheromone is performed after a resource has finished processing a job and the pheromone update is performed globally. The effect of the global pheromone update is to make an already chosen resource less desirable for the next ant.
5. A simulation model that can be used to simulate the grid environment and evaluate the proposed enhanced ACO algorithm in terms of processing time and the utilization of each resource.

CONTRIBUTION OF THE RESEARCH

The main contribution of the research is to show how the ant tries to match submitted jobs with available resources. To achieve this, the initial pheromone value is calculated by considering the characteristics of each job and the capacity of each resource such as the size and the CPU, the time needed by each job and also the bandwidth, the MIPS and the current load of each resource. By considering these aspects, jobs can be scheduled well to the suitable resources. It can reduce the processing time of each resource and also balance the entire resources.

The pheromone updating mechanism is used to support the idea of diversification as the pheromone updating is the means to store new experiences of ants. The global pheromone update in the EACO algorithm plays an important part by reducing the pheromone value on the resource that completely processed a job. The effect of this aspect is to make an already chosen resource less desirable for the next ant.

A range for the best value of each evaporation rate which is the control parameter is defined. Evaporation rate is different for different sets of jobs to be processed. A dynamic evaporation rate that can automatically change the evaporation rate based on the number of jobs submitted and the number of available resources was proposed. By considering this aspect, the EACO algorithm successfully scheduled jobs among the resources which lead to a balanced load network.

The contributions mentioned above were able to minimize job computational time, match jobs with suitable resources and balance entire resources in grid environment.

FUTURE WORK

In grid environments, computational performance changes from time to time, network connections may become unreliable, resources may join or leave the system at any time and resources may become unavailable without any notifications will require a dynamic scheduling algorithm in managing jobs and resources. Future work can enhance the proposed EACO to consider the changes such as resource failure. In this case the dynamic algorithm will stop and migrates jobs to other available resources in a dynamic evaporation rate environment.

Another potential future work is to apply the proposed algorithm on multiple ant colonies in solving the grid resource management problem. Using multiple ant colonies might improve the performance of the scheduling algorithm as ant populations will be divided into appropriate number of colonies to find the appropriate ways for these colonies to organize their activities with high level cooperation.

REFERENCES

Abawajy, J. (2009). Adaptive hierarchical scheduling policy for enterprise grid computing systems. *Journal of Network and Computer Applications, 32*(3), 770-779.

Abraham, A., Liu, H., Zhang, W., & Chang, T. (2006). Scheduling jobs on computational grids using fuzzy particle swarm algorithm. *Proceedings of the 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems,* 500-507.

Ali, A., Belal, M., A., & Al-Zoubi, M., B. (2010). Load balancing of distributed systems based on multiple ant colonies optimization. *American Journal of Applied Sciences, 7*(3), 433-438.

Bagherzadeh, J., & MadadyarAdeh, M. (2009). An improved ant algorithm for grid scheduling problem. *Proceedings of the 14th International CSI Computer Conference*, 323-328.

Bai, X., Yu, H., Ji, Y., & Marinescu, D. (2004). Resource matching and a matchmaking service for an intelligent grid. *International Journal of Computational Intelligence, 1*(3), 197-205.

Balasangameshwara, J., & Raju, N. (2010). A decentralized recent neighbour load balancing algorithm for computational grid. *International Journal of ACM Jordan, 1*(3), 128-133.

Baru, C., Moore, R., Rajasekar, A., & Wan, M. (1998). The SDSC storage resource broker. *Proceedings of the 1998 Conference of the Centre for Advanced Studies on Collaborative Research*, 1-12.

Bullnheimer, B., Hartl, R.F., & Strauss, C. (1996). A new ranked-based version of the ant system: A computational study. *Central European Journal of Operations Research and Economics, 7(1)*, 25-38.

Cao, J., Spooner, D., Jarvis, S., & Nudd, G. (2005). Grid load balancing using intelligent agents. *Future Generation Computer Systems, 21*(1), 135-149.

Carretero, J., & Xhafa, F. (2006). Use of genetic algorithms for scheduling jobs in large scale grid applications. *ŪKIO TECHNOLOGINIS IR EKONOMINIS VYSTYMAS, 12*(1), 11-17.

Chang, R., Chang, J., & Lin, P. (2007). Balanced job assignment based on ant algorithm for computing grids. *Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference*, 291-295.

Chen, T., Zhang, B., Hao, X., & Dai, Y. (2006). Task scheduling in grid based on particle swarm optimization. *Proceedings of the 5th International Symposium on Parallel and Distributed Computing* (ISPDC'06), 238-245.

Chen, Y. (2008). Load balancing in non-dedicated grids using ant colony optimization. *Proceedings of the 4th International Conference on Semantics, Knowledge and Grid,* 279-286.

Chtepen, M., Dhoedt, B., & Vanrolleghem, P. (2005). Dynamic scheduling in grid systems. *6th FirW PHD Symposium*, 110 -111.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. *Proceedings of European Conference on Artificial Life*. Paris, France, Amsterdam: Elsevier Publishing, 134-142.

Cordon, O., Fernandez, I., Herrera, F., & Moreno, L. (2000). A new ACO model integrating evolutionary computation concepts: The best-worst ant system. *Proceedings of ANTS2000 – From Ant Colonies to Artificial Ants: A Series of International Workshops on Ant Algorithms,* 22-29.

Cordon, O., Herrera, F., & Stutzle, T. (2002). A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware and Soft Computing, 9*(2/3), 141-175.

Dorigo M., V. Maniezzo & A. Colorni (1991a). Positive feedback as a search strategy. *Technical Report No. 91-016*, Politecnico di Milano, Italy.

Dorigo M., V. Maniezzo & A. Colorni (1991b). The ant system: An autocatalytic optimizing process. *Technical Report No. 91-016 Revised*, Politecnico di Milano, Italy.

Dorigo, M. (1992). *Optimization, learning and natural algorithms.* (Unpublished PhD thesis). Politecnico di Milano, Italy.

Dorigo, M., & Gambardella, L. (1997a). Ant colonies for the travelling salesman problem. *BioSystems*, *43*(2), 73-81.

Dorigo, M., & Gambardella, L. (1997b). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

Dorigo, M., & Socha, K. (2006). An introduction to ant colony optimization. *Handbook of Approximation Algorithms and Metaheuristics*, 26.21–26.14.

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization.* Cambridge, Massachusetts, London, England: MIT Press.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, *26*(1), 29–41.

Fattahi, S. M., & Charkari, N. M. (2009). Distributed ACS Algorithm for Resource Discovery in Grid. International Conference on IT to Celebrate S. Charmonman's 72nd Birthday, 37.1-37.7.

Fidanova, S., & Durchova, M. (2006). Ant algorithm for grid scheduling problem. *Large-Scale Scientific Computing*, 405-412.

Foster, I., & Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications, 11*(2), 115-128.

Foster, I., & Kesselman, C. (2004). *The grid: Blueprint for a new computing infrastructure.* Morgan Kaufmann.

Frey, J., Tannenbaum, T., Livny, M., Foster, I., & Tuecke, S. (2002). Condor-G: A computation management agent for multi-institutional grids. *Journal of Cluster Computing, 5*(3), 237-246.

Grimshaw, A., Ferrari, A., Knabe, F., & Humphrey, M. (1999). Wide area computing: Resource sharing on a large scale. *Computer, 32*(5), 29-37.

Grosan, C., Abraham, A., & Helvik, B. (2007). Multi-objective evolutionary algorithms for scheduling jobs on computational grids. *Proceedings of the International Conference on Applied Computing*, 459-463.

Ibarra, O., & Kim, C. (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the Association for Computing Machinery (JACM), 24*(2), 280-289.

Kaegi, S., & White, T. (2003). Using local information to guide ant based search. *Proceedings of the 16th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, 692-701.

Kawamura, H., Yamamoto, M., Suzuki, K., & Ohuchi, A. (2000). Multiple ant colonies algorithm based on colony level interactions. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 83*(2), 371-379.

Kong, X., Shen, H., Chen, X., Wang, C., & Song, C. (2010). Dynamic grid scheduling algorithm based on self-adaptive Tabu Search. *Proceedings of the International Conference on Computer Design and Applications (ICCDA), 2*, V2-271-V2 274.

Kousalya, K., & Balasubramanie, P. (2008). A solution to grid scheduling problem using an improved ant algorithm. *Journal of the Advances in Computational Sciences and Technology, 1*(2), 113-126.

Kousalya, K., & Balasubramanie, P. (2009). To improve ant algorithm's grid scheduling using local search. *International Journal of Computational Cognition, 7*(4), 47-57.

Krauter, K., Buyya, R., & Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. *Journal of Software: Practice and Experience, 32*(2), 135-164.

Li, Y. (2006). A bio-inspired adaptive job scheduling mechanism on a computational grid. *International Journal of Computer Science and Network Security, 6*(3B), 1-7.

Li, Y., Yang, Y., & Zhu, R. (2009). A hybrid load balancing strategy of sequential tasks for computational grids. *Proceedings of the International Conference on Networking and Digital Society,* 112-117.

Liu, A., & Wang, Z. (2008). Grid task scheduling based on adaptive ant colony algorithm. *Proceedings of the International Conference on Management of e-Commerce and e-Government,* 415-418.

Livny, M., & Melman, M. (1982). Load balancing in homogeneous broadcast distributed systems. *ACM SIGMETRICS Performance Evaluation Review, 11*(1), 47-55.

Lorpunmanee, S., Sap, M., N., Abdullah, A., H., & Chompoo-inwai, C. (2007). An ant colony optimization for dynamic job scheduling in grid environment. *International Journal of Computer and Information Science and Engineering, 1*(4), 207-214.

Moallem, A. (2009). *Using swarm intelligence for distributed job scheduling on the grid* (unpublished master thesis). University of Saskatchewan, Canada.

Moallem, A., & Ludwig, S. (2009). Using artificial life techniques for distributed grid job scheduling. *Proceedings of the 2009 ACM Symposium on Applied Computing,* 1091-1097.

Naik, V., K., Garbacki, P., Kummamuru, K., & Zhao, Y. (2006). On-line evolutionary resource matching for job scheduling in heterogeneous grid environments. *Proceedings of the 12th International Conference on Parallel and Distributed Systems* (ICPADS'06), 103-108.

Perretto, M., & Lopes, H. (2005). Reconstruction of phylogenetic trees using the ant colony optimization paradigm. *Genetic and Molecular Research, 4*(3), 581–589.

Rose, C., A., F., D., Ferreto, T., Calheiros, R., N., Cirne, W., Costa, L., B., & Fireman, D. (2008). Allocation strategies for utilization of space-shared resources in bag of tasks grids. *Future Generation Computer Systems, 24*(5), 331-341.

Sadhasivam, S., & Meenakshi, K. (2009). Load balanced, efficient scheduling with parallel job submission in computational grids using parallel particle swarm optimization. *World Congress on Nature & Biologically Inspired Computing,* 175-180.

Salehi, M., & Deldari, H. (2006). Grid load balancing using an echo system of intelligent ants. *Proceedings of the 24th IASTED International Conference on Parallel and Distributed Computing and Networks*, 47-52.

Sathish, K., & Reddy, A. (2008). Enhanced ant algorithm based load balanced task scheduling in grid computing. *International Journal of Computer Science and Network Security, 8*(10), 219-223.

Sharma, A., & Bawa, S. (2008). Comparative analysis of resource discovery approaches in grid computing. *Journal of Computers, 3*(5), 60-64.

Somasundaram, T. S., Balachandar, R., Kandasamy, V., Buyya, R., Raman, R., Mohanram, N., et al. (2006). Semantic-based grid resource discovery and its integration with the grid service broker. *International Conference on the Advanced Computing and Communications, 1-8.*

Stützle, T., & Hoos, H. (2000). MAX-MIN ant system. *Future Generation Computer Systems, 16*(9), 889-914.

Subrata, R., Zomaya, A., & Landfeldt, B. (2007). Artificial life techniques for load balancing in computational grids. *Journal of Computer and System Sciences, 73*(8), 1176-1190.

Tangmunarunkit, H., Decker, S., & Kesselman, C. (2003). Ontology-based resource matching in the grid–the grid meets the semantic web. *The Semantic Web-ISWC 2003*, 706-721.

Wang, Q., Gao, Y., & Liu, P. (2006). Hill climbing-based decentralized job scheduling on computational grids. *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, 1,* 705-708.

Wenming, H., Zhenrong, D., & Peizhi, P. (2009). Trust-based ant colony optimization for grid resource scheduling. *Proceedings of the Third International Conference on Genetic and Evolutionary Computing*, 288-292.

Xu, Z., Hou, X., & Sun, J. (2003). Ant algorithm-based task scheduling in grid computing. Proceedings of the *Canadian Conference on Electrical and Computer Engineering, 2*, 1107- 1110.

Yan, H., Shen, X., Li, X., & Wu, M. (2005). An improved ant algorithm for job scheduling in grid computing. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, 5,* 2957-2961.

Yan, K., Wang, S., Wang, S., & Chang, C. (2009). Towards a hybrid load balancing policy in grid computing system. *International Journal of Expert Systems with Applications, 36*(10), 12054-12064.

YarKhan, A., & Dongarra, J. (2002). Experiments with scheduling using simulated annealing in a grid environment. *Proceedings of the Third International Workshop on Grid Computing*, 232-242.

Zhan, Z., Zhang, J., Li, Y., & Chung, H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39*(6), 1362-1381.

Zhu, Y., & Hu, Y. (2004). Towards efficient load balancing in structured P2P systems. *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 20-29.

Zhu, Y., Xu, W., & Shen, P., (2009). Research of grid resource discovery method based on Adjacency list and ant colony Algorithm. *International Conference on Intelligent Human-Machine Systems and Cybernetics, 201-205.*