CHAPTER 1

OBJECTIVES AND OUTLINE OF THE RESEARCH

1.1 INTRODUCTION

A distributed system is a computing platform where hardware or software components located at networked computers communicate and coordinate their actions by passing messages. It enables users to access services and execute applications over a heterogeneous collection of computers and networks. Heterogeneity applies to networks, computer hardware, operating systems etc. In this research, Heterogeneous Distributed Computing Systems (HDCS) such as Grid computing systems which utilizes a distributed set of high performance machines, connected with high speed networks to solve computationally intensive applications coordinately, (Freund and Siegel 1993, Foster and Kesselman 2003) is considered.

Applications consist of a set of tasks, with or without dependencies among them. The set of tasks with data dependencies among them is represented by a Directed Acyclic Graph (DAG). The estimated computation time for each task on each machine is assumed to be known a priori. The amount of data to be transferred from one task to other task is also assumed to be known a priori.

Task scheduling in distributed computing systems is defined as the process of assigning tasks of a distributed application into the available processors. The challenge of task scheduling is to find a spatial and temporal
assignment of the tasks onto the processors, which results in the fastest possible execution, while confirming the precedence constraints expressed by the edges of the DAG.

One of the most important components for achieving high performance with HDCS is the mapping strategies they adopt. Mapping of an application involves the matching of tasks to machines and scheduling the order of execution of these tasks (Braun et al 2001). In general, the scheduling problem is computationally intractable even under simplified assumptions (Garey and Johnson 1979). Finding a schedule of minimal length for a given task graph is, in its general form, an NP-hard problem. Its associated decision problem is NP-complete and an optimal solution cannot be found in polynomial time. As a consequence of the NP-hardness of scheduling, a vast area emerged that deals with all aspects of task scheduling, ranging from its theoretical analysis, to heuristics, and approximation techniques, which produce near optimal solutions. Many heuristics have been proposed (Kwok and Ahmad 1999) to solve this problem. The complexity of the problem increases when the application is executed in a HDCS due to the fact that the processors and network connections in the system may not be identical and it takes different amounts of time to execute the same task or transfer the same amount of data.

The task scheduling problem is classified into static and dynamic based on the time at which the scheduling decision is made. This research focuses on static task scheduling problem. The static task scheduling problem is solved by heuristic based methods and metaheuristics based methods.

Figure 1.1 shows the classification of important approaches used for solving the task scheduling problem. Metaheuristics based methods are designed to find good approximations to the optimal solution in the large complex search spaces. Research work carried out in this thesis for solving
task scheduling problem using heuristic and metaheuristics approaches are represented as shaded boxes in the Figure 1.1.

List scheduling based heuristics usually generate good quality schedules at a reasonable cost. In its simplest form, the first part of list scheduling sorts the nodes of the DAG to be scheduled according to a priority scheme, while preserving the precedence constraints of the tasks—that is, the resulting task list is in topological order. In the second part, each task of the list is successively scheduled to a processor chosen for the task. Usually, the chosen processor is the one that allows the task to start as early as possible.

![Classification of scheduling algorithms for heterogeneous distributed computing systems](image)

**Figure 1.1 Classification of scheduling algorithms for heterogeneous distributed computing systems**

In the clustering heuristics, the tasks are grouped into clusters and the clusters are mapped to the processors. After mapping the task clusters to
the processors the tasks are scheduled on the processor. The main idea in creating the task cluster is based on reducing the communication cost among the tasks. The basic idea behind task duplication based (TDB) scheduling algorithms is to use the idle time slots on certain processors to execute duplicated predecessor tasks that are also being run on some other processors, such that communication delay and network overhead can be minimized.

The task-scheduling problem is a search problem where the search space consists of an exponential number of possible schedules with respect to the problem size. Metaheuristics based methods are a class of search methods based on enumerative techniques with additional information used to guide the search. They have been used extensively to solve very complex problems. Generally, metaheuristics can be divided into trajectory methods (also named local search heuristics) and population-based methods. Population-based methods deal with a set of solutions in every iteration of the algorithm while trajectory methods only deal with a single solution.

As one of the most studied population-based methods, Genetic Algorithm (GA) shows robust performance with various scheduling problems, for it has a powerful global exploration ability of concurrently tracking a set of solutions. Plenty of empirical results demonstrate that GA-based methods always outperform traditional heuristic-based scheduling algorithms regarding the schedule quality. Efficient solutions have resulted from the use of GA in task scheduling algorithms (Hou et al 1994, Siegel et al 1996, Zomaya and Teh 2001, Oh and Wu 2004, Singh et al 2009, Omara and Arafa 2010). However; GA usually takes more computing efforts to locate the optimal in the region of convergence, owing to the lack of local search ability. Hence, it is a natural choice to consider the hybridization of metaheuristics, also named memetic algorithm (MA), which has been applied to solve scheduling problems. Evolutionary Programming (EP) is a mutation-based evolutionary
algorithm applied for discrete search spaces and used in task scheduling (Fogel and Fogel 1996, Kwok and Ahmad 1999).

The static scheduling algorithms for heterogeneous distributed computing systems are originally developed for handling the main objective of minimizing the schedule length (makespan) of the application. But, as the Quality of Service (QoS) requirements in the Heterogeneous Distributed computing systems increases, it is essential to consider more than this single objective of minimizing the schedule length.

As the heterogeneous systems become larger and larger, processors and network failures are inevitable and can have adverse effect on applications executing on the system. Indeed, the number of possible failures increases with the size of the hardware. Therefore, it is not possible to ignore the fact that an application running on a very large system can crash due to hardware failure. (Plank and Elwasif 1998) found out that the estimated value of mean time-to-failure of a machine connected to a local area network ranged from 4 to 33 days. Several approaches are employed to solve this problem.

One approach is based on task duplication where each task is executed more than once in order to decrease the probability of failure. The main problem of this approach is that it increases the number of required processors. Alternatively, it is possible to checkpoint the application and restarts the application after a failure (Dogan and Ozguner 2006, Dongarra et al 2007). However, in case of failure, the application is slowed down by the restart mechanism, which requires the user to restart the application on a subset of processors and repeat some communications and computations. Hence, in order to minimize the impact of the restart mechanism, it is important to reduce the risk of failure. Moreover, even in the case where there is no checkpoint-restart mechanism, it is important to guarantee that the probability of failure of the application is kept as low as possible.
Unfortunately, increasing the reliability implies, most of the time, an increase of the execution time. This justifies the search for the algorithms that both minimizes makespan and maximizes reliability.

A reliable scheduling technique has been applied to allocate distributed programs (Kumar et al 1988, Hwang and Tseng 1993, Chang et al 1999), undirected task graphs (Shatz et al 1992, Kartik and Murthy 1997) and Directed Acyclic Graph (DAG) (Srinivasan and Jha 1999, Dogan and Ozguner 2002, Dongarra et al 2007, Girault et al 2009) to distributed systems. In addition reliability has been considered for real-time systems by (Hou and Shin 1997, Qin and Jiang 2005, Prodan and Wieczorek 2008, Yoo 2009).

It was shown by (Dogan and Ozguner 2002) that a scheduling algorithm that minimizes only the schedule length may lead to a high failure probability, and that a reliable scheduling algorithm that minimizes only the failure probability may yield a high schedule length for an application running on HDCS. This result implies that a scheduling algorithm must account for both the execution time and the failure probability of an application. In addition, there are usually conflicting requirements between minimizing the execution time and the failure probability of an application, and it may not be possible to simultaneously minimize both. Consequently, a scheduling algorithm must be capable of balancing the execution time and failure probability of the application, i.e. it should be able to produce task assignments, whereby the execution time is decreased at the expense of higher failure probability or vice versa.

There are several algorithms in the literature that addresses the problem of minimizing both the execution time and the failure probability of applications. (Dogan and Ozguner 2002) proposed a bi-criteria heuristic called Reliable Dynamic Level Scheduling (RDLS). The solution was improved using a genetic algorithm based approach by (Dogan and Ozguner
2005). In the study made by (Qin and Jiang 2005), the two objectives are not considered simultaneously, this algorithm first tries to guarantee the timing constraints of the tasks. Then, among the processors on which the task’s deadline is guaranteed, the task is mapped to a processor which minimizes the failure probability of the application.

A Bi-objective Scheduling Algorithm (BSA) by (Hakem and Butelle 2007) is a bi-criteria heuristic that outperforms RDLS. The problem of maximizing the reliability and minimizing the makespan on related machines where processors are subject to crash fault is studied by (Dongarra et al 2007). A Bi-criteria scheduling algorithm for scientific workflows is proposed by (Prodan and Wieczorek 2008). Multiobjective Differential Evolution approach is used by (Talukder al 2009) to solve the task scheduling problem on global grids considering the conflicting objectives of execution cost and time.

1.2 MOTIVATIONS

The current trend in designing scheduling algorithms is with respect to user’s demands, that is, provide Quality of Service (QoS) based scheduling and this research study can be considered to be a step in that direction. In implementing a scheduler, challenging problems such as including task profiling for a given application, analytical benchmarking of the machines in the system, resource management, etc., need to be addressed. Although individual solutions to each of these problems have been proposed, growing size of heterogeneous computing system makes it difficult to find efficient solutions to these problems that can be implemented in a practical system.

In this thesis, an application is modeled by a task graph. A task graph is a Directed Acyclic Graph (DAG) in which nodes represent tasks and edges represent the data dependencies among the tasks. The heterogeneous
system is modeled by a resource graph in which nodes represent processors and edges represent the links with a defined Data Transfer Rate (DTR) among the processors. The task scheduling problem is treated as a multiobjective optimization problem with the objectives of minimizing the makespan and maximizing the reliability. This justifies the need to come up with an algorithm that utilizes the stochastic information about the task execution times, failure rate of the resources and the data transfer rate to produce a better schedule in terms of minimizing the makespan and maximizing the reliability.

Heterogeneous Earliest Finish Time (HEFT) algorithm is a performance effective task scheduling algorithm, addressing the objective of minimizing makespan. Reliable Dynamic level Scheduling (RDLS) algorithm is a bi-objective scheduling algorithms that maximizes the reliability more effectively. The increase in makespan obtained by RDLS is very high compared to the improvement in reliability. Hence, both algorithms are combined and a new algorithm is proposed.

The traditional heuristics attempts to obtain a single solution compared to a set of pareto-tradeoff solution using Multiobjective evolutionary algorithms. Multiobjective evolutionary algorithms are well suited for solving multiobjective optimization problems. There are different Multiobjective evolutionary approaches available and a comparison of these approaches in task scheduling problem is required for selecting a particular approach. A comparison of Multiobjective Evolutionary Programming (MOEP) and Multiobjective Genetic Algorithm (MOGA) is presented.

Hybridization of Multiobjective evolutionary algorithms improves the convergence speed to Pareto front. Hence, there is a need to study the performance of the pareto-based algorithms in their pure and hybrid forms for solving the task scheduling problem. The convergence and diversity of the
obtained non-dominated solutions are important performance metrics for evaluating the multiobjective evolutionary algorithms.

1.3 CONTRIBUTIONS

In this research work, scheduling algorithms using traditional heuristics and random search heuristics are developed to address the problem of producing efficient schedules in HDCS. The contributions of this research are summarized as follows:

1. A framework for evaluating different scheduling algorithms has been developed for comparing our algorithms with other existing algorithms. A random task graph generator is designed to generate task graphs with specific parameters for the performance study. It can generate many types of graphs in order to perform unbiased comparisons of different algorithms. Real time application graphs of variable sizes are designed for the Gauss elimination and Fast Fourier Transform application.

2. A modification to the traditional list scheduling heuristic, Heterogeneous Earliest Finish Time algorithm (HEFT) is proposed and a new algorithm called Improved Reliable HEFT (IRHEFT) for minimizing the schedule length, balancing the load and maximizing the reliability of schedule is reported. IRHEFT takes into account the load difference factor for maximizing the reliability. The load on a processor is defined as the amount of time the processor is engaged in completing the scheduled subtasks. The algorithm is compared for its performance with RDLS algorithm for
randomly generated task graphs and a real application task graph.

3. Two multiobjective evolutionary algorithms namely, Multiobjective Genetic Algorithm (MOGA) and Multiobjective Evolutionary Programming (MOEP) both based on non-dominated sorting are compared with the existing bi-objective genetic algorithms in the literature. The pareto optimal front of MOGA and MOEP for a real application task graph is obtained and the diversity and convergence are obtained.

4. A comparison of multiobjective evolutionary algorithms (MOEAs) such as NSGA-II, SPEA2 and their hybrid versions for solving this multiobjective task scheduling problem is reported. A simple neighborhood search algorithm is used as local search procedure for hybridizing. The algorithms are compared for randomly generated task graphs and real application task graph. The performances of the MOEAs are measured in terms of the Convergence and Diversity of the obtained solutions.

1.4 OUTLINE OF THE THESIS

In this chapter, the scope and objectives of the research work, the application of heuristic and metaheuristics approaches to solve the bi-objective task scheduling problem used in this thesis are discussed. The rest of the thesis is organized as follows:

- Chapter 2 provides relevant background information about task scheduling in HDCS considering both single and bi-objective. An overview of the DAG scheduling problem is
presented first. Then, different existing scheduling algorithms are categorized and briefly reviewed.

- Chapter 3 explains the proposed Improved Reliable HEFT (IRHEFT) algorithm. It improves the reliability by considering the load difference factor in the processor selection phase of task scheduling. The performance of the proposed algorithm for random and real application task graphs is presented.

- Chapter 4 demonstrates the use of the two multiobjective evolutionary algorithms, MOGA and MOEP for solving the bi-objective task scheduling problem. The ability of the MOEAs to obtain Pareto optimal solutions at reduced computational times are reported. The performance comparison of the MOEAs for random and real application task graphs is presented.

- Chapter 5 focuses on the Pareto-based multiobjective evolutionary algorithms namely, NSGA-II and SPEA2 for solving the task scheduling problem. Then it describes the hybrid version of these algorithms. A Simple Neighborhood Search algorithm is proposed as the local search procedure. The performance comparison of the pure and hybrid version for random and real application task graphs are presented.

- Chapter 6 presents the conclusion of this research work and an indication of future research directions.