CHAPTER 1

1. INTRODUCTION

The rapid increase of processing power in low cost computers and the tremendous development in high-speed networking technologies have boosted the use of distributed computing environments for solving complex problems in the last fifteen years. The developments from small clusters of homogeneous computers in the 1980's till the present day's distributed computing environments include platforms formed by thousands of heterogeneous computing resources. It is widespread around the globe providing the computing power needed for solving complex problems arising in many areas of applications.

Nowadays, a common platform for distributed computing usually comprises a heterogeneous collection of computers that were capable of working cooperatively. In higher level abstraction, the concept of grid computing has become popular, representing the set of distributed computing techniques. This works over large loosely coupled virtual supercomputers that are formed by combining many heterogeneous components of different characteristics and computing powers.

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities [1]. It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the sharing of resources within the distributed communities. Resources can be computers, storage space, instruments, software applications and data, all connected through the Internet.
1.1 THE GRID SCHEDULING PROCESS AND COMPONENTS

A Grid Scheduler (GS) receives applications from grid users, selects feasible nodes for these applications according to the acquired information from the Grid Information Service (GIS) module, and finally generates application to node mappings, based on certain objective functions and predicted node performances.

A task scheduling is the mapping of tasks to a selected group of nodes which may be distributed in multiple administrative domains. The grid scheduling process can be generalized into three stages, namely node discovering and filtering, node selecting and scheduling according to certain objectives, and job submission [2]. Figure 1.1 depicts a model of grid scheduling system. Grid scheduler is referred to as Meta scheduler in the literature [3].

The role of the Grid Information Service is to provide information about the status of available nodes to grid schedulers. GIS is responsible for collecting and predicting the node state information such as CPU capacities, memory size, network bandwidth, software availabilities and load of a site in a particular period. GIS can answer queries for node information or push information to subscribers.

Besides raw node information from GIS, application properties such as approximate instruction quantity, memory and storage requirements, subtask dependency in a job and communication volumes and performance of a node for different application species are also necessary for making a feasible schedule. Application Profiling (AP) is used to extract properties of applications, while Analogical Benchmarking (AB) provides a measure of how well a node can perform a given type of job [4, 5]. Cost estimation module computes the cost of candidate schedules. On the basis of knowledge from AP, AB and cost estimation module, the scheduler chooses those that can optimize the objective functions.
Figure 1.1 A logical Grid scheduling architecture

The Launching and Monitoring (LM) module is known as the “binder” which implements a finally-determined schedule by submitting applications to selected nodes, staging input data and executables, if necessary, and monitoring the execution of the applications.

A Local Resource Manager (LRM) is mainly responsible for two jobs: local scheduling inside a node domain, where not only jobs from exterior grid users, but also jobs from the domain’s local users are executed, and reporting node information to GIS.

This research work is concentrated on the development of task scheduling algorithms for the grid scheduler.

1.2 CLASSIFICATION OF GRID SCHEDULING ALGORITHM

In general, task scheduling algorithm for heterogeneous systems is classified into two classes: static and dynamic [6,7,8]. In static scheduling algorithms, all information needed for scheduling such as the structure of the parallel application, the execution times of individual tasks and the communication costs between tasks must be known in advance [9,10]. Static task scheduling takes place during compile time before running the parallel application. In contrast, scheduling decisions in dynamic scheduling algorithms are made at run time. The objective of dynamic
scheduling algorithms includes not only creating high quality task schedules but also minimizing the run time scheduling overheads [6-8, 11-14].

This work deals with the static scheduling problem, in which all tasks can be independently performed. Static scheduler acts as the basic building block to develop a powerful dynamic scheduler, able to solve more complex scheduling problems. The concept of static scheduling frequently appears in many scientific research problems, especially in Single-Program Multiple-Data applications used for multimedia processing, scientific computing, data mining, parallel domain decomposition of numerical models for physical phenomena, etc. [15]. The overview of the heuristic and meta-heuristic algorithms experimented for the grid task scheduling problem is described in Figure 1.2.

1.3 RESEARCH SCOPE

Grid computing involves coordinating, sharing computing applications with data available from storage or network nodes across dynamic and geographically dispersed organizations [1]. Users can share grid nodes by submitting computing tasks to the grid system. Nodes can be computers, storage space, instruments,
software applications or data. All nodes are connected through the Internet and a middleware layer that provides basic services for security, monitoring and node management. The nodes of computational grid are dynamic, and they belong to different administrative domains. The participation of nodes may be active or inactive within the grid. This makes it impossible for anyone to manually assign jobs to computing nodes in grids. Therefore, grid job scheduling is one of the challenging issues in grid computing.

One of the main motivations of the grid computing paradigm has been the computational need for solving many complex problems from science, engineering and business that include hard combinatorial optimization problems, protein folding, financial modelling, etc. [16,17,18]. As a cooperative environment of solving problem, it is necessary for the grids to develop efficient task scheduling schemes and resource management policies with regard to their objectives, scope and structure.

Multiprocessor scheduling problems are widely studied in operational research. Also, numerous heuristic methods have been proposed for finding accurate schedules in reasonable times [19, 20]. In classic formulation, scheduling problems assume a computing environment composed of homogeneous resources. But the research community during the 1990's started to pay attention on scheduling problems in Heterogeneous Computing (HC) environments, specially due to the popularization of distributed computing and the growing use of heterogeneous clusters [21,22]. In the recent decade, significant effort has been made to study the scheduling problem on HC environments, since this platform provides the efficiency required for distributed and grid computing techniques.
Scheduling problem in heterogeneous environments had been shown to be NP complete [23]. Thus classic exact methods are only useful for solving such problems. But they only deal with problem instances of reduced size. Hence there are many research efforts aiming at new task scheduling techniques on the grid that are able to improve upon the traditional exact ones. The low efficiency often makes them useless in practice for solving large dimension scheduling problems in reasonable times. When dealing with large dimension computing environments, ad-hoc heuristic and meta-heuristic techniques have shown up as promising methods for solving the HC and grid scheduling problems.

It is known that meta-heuristic techniques usually require longer execution times in the order of few minutes than ad-hoc heuristics. But they consistently yield better solutions than ad-hoc heuristic methods. Hence they are considered as competitive schedulers for distributed HC and grid systems where large tasks with execution times in the order of minutes, hours and even days are submitted for execution [24, 25].

Evolutionary Algorithms (EA) and other meta-heuristics have been frequently applied to the Heterogeneous Computing Scheduling Problem (HCSP) and related problem variants in the last fifteen years. The most relevant proposals included Genetic Algorithms (GA)[24,26-29], Memetic Algorithms (MA)[30], cellular Memetic Algorithms (cMA)[31], and also hybrid methods combining EAs with other optimization techniques.

Two relevant works have obtained the best known results when facing a set of low-sized de facto standard HCSP instances using non-evolutionary meta-heuristics: a hybrid combining Ant Colony Optimization (ACO) algorithm and Tabu Search (TS) [32] that took a long time nearly over 3.5 hours to perform the search
and a hierarchic TS [33] that used a time limit of 100 seconds to run the scheduling algorithm. The other HCSP variants have been tackled using EAs, mostly remarkable precedence-constrained task scheduling problems in multiprocessors[34,35], Direct Acyclic Graph (DAG) scheduling in grid environments [36], real-time grid scheduling [37], economy-based scheduling [38], and other complex HCSP versions regarding many task attributes[39,40].

Despite the numerous proposals on applying EAs and other meta-heuristics to the HCSP and related variants, some of the works have tackled realistic instances in grid environments mainly due to the inherent complexity of dealing with the underlying high dimension optimization problem.

In order to further improve the efficiency of meta-heuristics, parallel implementations became a popular option to speed up the search that allows reaching high quality results in a reasonable execution time even for hard-to-solve optimization problems [41,42]. Nesmachnow et al. [15] proposed parallel Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation (pCHC) algorithm to solve the realistic HCSP instances.

Till now none of the sequential execution algorithms outperformed the recently published results of parallel algorithm [15] for the de facto standard problems by Braun et al. [24]. This paves the way to contribute in these lines of research by studying sequential execution algorithms which are able to deal with large size scheduling problem instances by using innovative concepts embedded with sequential execution algorithms.

This thesis presents the application of sequential meta-heuristics for solving the HCSP in grid environments. The experimental study was initially aimed to analyze the efficiency of the existing meta-heuristics which are not applied to the
grid scheduling problem, namely Biogeography Based Optimization (BBO) algorithm, Differential Evolution (DE) algorithm, Real valued Augmented Simulated Annealing (RASA) algorithm, Variable Neighbourhood Search (VNS) algorithm to solve a de facto standard set of small-sized HCSP instances.

This analysis led to the proposal of a novel algorithm such as Two Phase Variable Neighbourhood Search (TPVNS) algorithm that uses problem-specific novel neighborhood structures and a novel crossover heuristic algorithm in order to efficiently solve the HCSP. In addition, a new hybrid algorithm, namely Differential Evolution/Real valued Augmented Simulated Annealing (DE/RASA) algorithm has been proposed to deal with large size scheduling problem instances.

The experimentation of this work covers the development of meta-heuristics for both independent and dependent task applications in order to optimize the system related optimization criterion, makespan. In addition, the performance of the proposed meta-heuristics has been improved with a novel local search method to optimize a linear combination of both system-related and user-related objectives, namely makespan and flowtime.

After that, the effectiveness of the proposed meta-heuristics was evaluated for solving large heterogeneous computing scheduling problem instances [8]. The performance of the proposed algorithms was compared with the well-known deterministic heuristics and meta-heuristics over a large set of instances.

1.4 RESEARCH OBJECTIVE

The main objectives of the research work are

i. to analyze sequential meta-heuristic techniques for solving the scheduling problem on heterogeneous computing and grid environments,
ii. to introduce a novel TPVNS algorithm to deal with large size scheduling problem instances,

iii. to introduce the hybrid DE/RASA algorithm, a novel meta-heuristic for solving the HCSP which exploits the characteristics of both DE and RASA algorithm,

iv. to improve the effectiveness of the meta-heuristic algorithms using a novel local search method so as to deal with multi-objective metrics,

v. to evaluate the performance of meta-heuristics for the independent and dependent task benchmark applications,

vi. to improve the quality of the best well-known results of the HCSP instances by embedding innovative concepts with the meta-heuristic techniques, and

vii. to introduce a flexible novel two-phase DAG scheduling algorithm, which is the outcome of the combination of well-known heuristic algorithm, Heterogeneous Earliest Finish Time (HEFT) with the meta-heuristic algorithm and VNS algorithm.

1.5 ORGANIZATION OF THE THESIS

The content of the thesis is structured as follows.

Chapter 2: Reviews of previous works that have recently tackled the heterogeneous and grid computing scheduling problem are discussed in this chapter. The review of related works identifies three major areas, based on the dependency of task and the use of the metrics. The reviewed works are organized according to whether they tackle a single-objective independent task scheduling problem, or a multi-objective
independent task scheduling problem using a single-objective approach, or a single-objective dependent task scheduling problem.

**Chapter 3:** Grid task scheduling formulation is presented in this section. Independent and dependent task scheduling models are described in detail. The independent task scheduling problem is formulated based on the “Expected time to compute” \((ETC)\) model, whereas in the dependent task scheduling problem, the application is represented as the Direct Acyclic Graph. The properties of \(ETC\) matrix and the methodology to find the objective function for the dependent task scheduling algorithms are explained in detail.

**Chapter 4:** This chapter describes the implementation of various makespan-based static independent task scheduling algorithms. First, the schema of hybrid DE/RASA algorithm and the Two-Phase Variable Neighborhood Search algorithm for scheduling independent jobs on computational grid is explained. Second, the determination of algorithmic specific parameters and the order of usage of the neighborhood functions are determined. Then, the representation of the solution and the generation of initial solution are explained. Next, the experimental study to analyze the efficiency of the Biogeography-Based Optimization algorithm, Differential Evolution algorithm, RASA algorithm, DE/RASA algorithm and TPVNS algorithm to solve a de facto standard set of HCSP instances is presented. Then, the efficient numerical results of the proposed algorithms and their comparison with the best existing heuristics and meta-heuristic algorithms are reported.
Chapter 5: This chapter discusses the multi-objective static scheduling algorithms to optimize a linear combination of both system-related and user-related objectives, namely makespan and flowtime. The experimental evaluation of BBO, DE, RASA, DE/RASA and VNS algorithms for solving the HCSP is presented.

Chapter 6: This chapter presents the application of the VNS algorithm and other heuristic algorithms to the DAG scheduling problem aimed at optimizing the makespan metric. Two sets of graphs are used as the workload to test the performance of DAG scheduling algorithms: the randomly generated application graphs and the real world application graphs, in which three numerical applications are considered. The details of the experimental analysis performed to evaluate the heuristic and meta-heuristic algorithms are reported.

Chapter 7: Conclusion of the research is presented in this chapter. A summary of the main contributions of the research along with the possible lines to continue the work in the near future is also given.