# Chapter 7

# Conclusions and Future Directions

In distributed computing, a big application is solved by dividing it into many tasks and executing them onto different processors. The multiprocessor environments for distributed computing may be homogeneous in which all processors have same processing capabilities, or it may be heterogeneous in which all processors are comprised of different processing capabilities. It involves potentially a great deal of communication overhead which restricts the performance of applications if tasks are not scheduled efficiently. The scheduling of tasks, with precedence constraints, on to different processors is one of the core concerns for distributed computing in multiprocessor environments and significantly relies on the techniques employed to schedule the tasks with the aim of optimizing makespan and energy consumption. The task scheduling problem is known to be NP-complete. Therefore, many task scheduling algorithms are proposed in literature to solve this problem and new methods keep coming in. It is always useful to look for a fresh approach, towards understanding and interpretation of the existing algorithms and such an effort may lead to some possible newer ways of solving the problem. The thesis benchmarks some existing task scheduling algorithms and proposes a possible framework for this purpose. The work also attempts to propose some new approaches for working out possible scheduling, of tasks that optimize makespan. Further, a scheduling algorithm that minimizes energy consumption has also been proposed. Performance of the proposed algorithms are evaluated and compared with some existing algorithms for various types of applications.

Chapter 1 introduces distributed computing and task scheduling in multiprocessor environment. It presents some background concepts, motivation, and summarized the contributions made. To provide a comprehensive understanding of the existing body of knowledge, Chapter 2 provides the preliminary concepts and a survey of related work regarding task scheduling algorithms for distributed computing in multiprocessor environments. Here, some existing algorithms are reviewed, compared, and classified. This was done by studying the scheduling, application, and types of computing systems considered by state-of-the-art algorithms.

Chapter 3 evaluated and compared the performance of six well-known list scheduling algorithms for distributed computing systems. The algorithms used are HEFT, PETS, LDCP, Lookahead, CEFT and PEFT. The analysis of results is performed on different performance metrics like scheduling length ratio and efficiency for randomly generated graphs and the graphs generated from real-world applications such as FFT, Gaussian Elimination, Montage and Epigenomics workflows. This chapter further explored possibility of a framework for benchmarking of task scheduling algorithms for distributed computing systems. Such a framework ascribes a collection of activities that need to be taken up for the benchmarking purpose, including random graph generation and workflow generation as part of DAG generator, a scheduler and an analyzer under the control of an environment provider. The proposed framework is general in nature.

Chapter 4 presented our proposed Edge Priority Scheduling (EPS) algorithm for scheduling of tasks, with precedence constraints, in multiprocessor environments. This algorithm optimizes normalized schedule length and speedup. EPS extends and enriches the idea of Sarkar's algorithm, defining and using a concept of edge priority. The idea of computing priority of edges by their associated computation and communication costs gives preference to that edge which links communication-intensive clusters when performing clustering. Our scheme of prioritization provides an appropriate technique for obtaining meaningful clustering that can consequently help in improving the execution characteristics. A comparison has been carried out to obtain the possible performance improvement of the proposed EPS algorithm over six well-known algorithms, namely EZ, LC, CPPS, DCCL, RDCC and LOCAL. The results are presented for two types of task graphs - randomly generated benchmark task graphs and task graphs generated from real-world applications such as

Gaussian Elimination and FFT. The results demonstrate that overall, EPS produces considerably better results than the compared algorithms for randomly generated graphs and Gaussian Elimination graphs. For FFT graphs, EPS gives performance similar to that of EZ, CPPS and LOCAL whereas EPS gives better performance than DCCL and RDCC.

Chapter 5 presented our proposed Effective Critical Path (ECP) algorithm for scheduling of tasks with precedence constraints in multiprocessor environments. ECP is a clustering-based task scheduling technique. It makes use of edge zeroing concept on the critical path to reduce the communication cost among the tasks of the task graph with the goal of optimizing normalized schedule length and speedup. A comparison has been carried out to know the performance improvement of the proposed ECP algorithm over four well-known algorithms, namely EZ, LC, CPPS, and LOCAL and the algorithm proposed in the previous chapter 4. The results are presented for two types of task graphs namely, randomly generated benchmark task graphs and task graphs generated from real-world applications such as Gaussian Elimination, FFT and Systolic array. The ECP algorithm proposed here significantly outperforms the said algorithms in terms of normalized schedule length and speedup for all types of task graphs.

Chapter 6 presented an Energy Aware Edge Priority Scheduling (EAEPS) algorithm for scheduling of tasks with precedence constraints on multiprocessor environments that aims to minimize power consumption by exploiting DVFS technique. EAEPS is an energy aware version of our proposed EPS algorithm that is discussed in chapter 4. EAEPS uses the same priority function defined for EPS and reduces the energy consumption by zeroing the edges with high priorities. Here, prioritization scheme aims to obtain meaningful clustering that can consequently help in minimizing energy consumption. The simulation experiments conducted with four well-known energy aware scheduling algorithms for some selected benchmark random task graphs demonstrated that the proposed algorithm achieved more energy saving than compared energy aware algorithms.

For future work, the proposed framework in chapter 3 may be improved by working on scheduling algorithms and evaluating them for different workflows like Cybershake, LIGO and SIPHT. The proposed framework is general in nature and may be used for performance assessment and comparison of other groups of scheduling algorithms like duplication-based and clustering-based algorithms. The framework may also be extended for different computing system environments like cluster computing, grid computing and cloud computing. The clustering-based algorithms proposed in chapter 4 and 5 may be integrated with the existing list scheduling and duplication-based scheduling techniques for different real-world applications. These algorithms may also be implemented for bounded number of processors. The energy-aware scheduling algorithm proposed in chapter 6 may be deployed in some real applications, for example, Montage workflow and Epigenomics workflow.