CHAPTER 1

INTRODUCTION

The advances in hardware and software technologies have led to the increased use of distributed heterogeneous computing systems in the recent years for executing larger application programs. Distributed heterogeneous computing systems have the potential to provide low cost and high performance computing and hence it is considered as an economical substitute for high performance computing machine. However, in order to effectively exploit the computing power of distributed system, it is very much essential to employ a proper scheduling algorithm for allocation and sequencing of tasks of an application program to the available processors. This research work is mainly focused on the design and development of scheduling algorithms for distributed heterogeneous computing system. In this chapter, the research problem is briefly introduced by recounting distributed heterogeneous computing system, parallel program representation and task scheduling. The motivation, objectives and the problem statement of the research work are also formulated and explained. Finally, the contributions out of this research and the organization of the chapters in thesis are presented in a nutshell.

1.1 Distributed Heterogeneous Computing System

Diverse portions of an application task often require different types of computation. In general, it is impossible for a single machine architecture with its associated compiler, operating system, and programming tools to satisfy all the computational requirements in such an application equally well. However, a Distributed Heterogeneous Computing System (DHCS) that consists of a heterogeneous suite of processors, high-speed interconnections, interfaces, operating systems, communication protocols and programming environments provides a variety of architectural capabilities, which can be orchestrated to perform an application that has diverse execution requirements [1, 2]. DHCS is now well
recognized as an important computing paradigm in meeting the computational requirements of many applications in science, engineering and commerce such as weather modeling, mapping of the human genome, image processing, modeling of semiconductors, superconductors and banking systems [3-6]. While the distributed computing systems offer the promise of vastly increased performance, it introduces additional complexities such as scheduling of parallel program, balancing load among the processors, process synchronization, communication, etc. which are not encountered with sequential processing. To effectively harness the computing power of DHCS, it is crucial to employ a judicious scheduling algorithm for proper allocation of tasks onto the DHCS.

1.2 Task Scheduling Problem

Task scheduling is of vital importance in DHCS since a poor task scheduling algorithm can undo any potential gains from the parallelism present in the application. Moreover, an inappropriate scheduling can result in the hardware being used inefficiently or worse, the program could run slower in distributed computing systems than on a single processor [7]. A well known strategy behind efficient execution of extremely large application on a DHCS is to partition the application into multiple independent tasks with some precedence relationship among the tasks, and schedule such tasks over a set of available processors. A task partitioning algorithm can be employed to partition a parallel application into a set of precedence-constrained tasks represented in the form of a Directed Acyclic Graph (DAG), whereas a scheduling algorithm can be used to schedule the DAG onto the processors of the DHCS in order to minimize the makespan or schedule length of an application. Makespan is defined as the time at which all tasks have finished execution. The process of scheduling a DAG onto a distributed system is depicted in Figure 1.1. Here the vertices in the DAG represent the tasks and the edges represent the amount of data (output) to be transferred from one task to another. Each task is labelled with a task number and the estimated computation time for execution. The target system consists of four completely connected processors and for illustration it is assumed that each task takes the same amount of execution time on all the four processors. The Gantt chart shows the makespan generated by the scheduler.
The problem of task scheduling has been proven to be NP-complete [8, 9] in the general case as well as in some restricted cases such as scheduling tasks with one or two time units to two processors and scheduling unit time tasks to an arbitrary number of processors [10]. Approaches based on graph-theoretic techniques [11] and integer programming techniques [12, 13] to determine the optimal solution also exist. However, these methods are computationally very expensive. Hence attention has been focused on heuristic methods, which provide sub-optimal but acceptable performance with the benefit of polynomial time complexity.

Figure 1.1 Task scheduling process

The task scheduling problem is broadly classified into deterministic scheduling problem and nondeterministic scheduling problem [14] based on the characteristics of the program and tasks to be scheduled, system architecture and the availability of the information. A deterministic scheduling problem is one in which all information about the tasks and their relations among them, such as execution
time and precedence relations are known to the scheduling algorithm in advance. A
deterministic scheduling problem is also known as static scheduling or compile-time
scheduling. A nondeterministic problem is one in which some information about
tasks and their relations may be undeterminable until run-time. A nondeterministic
scheduling problem is also known as dynamic scheduling or run-time scheduling.
When the characteristics of an application, which includes execution time of tasks
on different processors, the data size of communication between the tasks and the
task dependencies, are known a priori, it is represented with a static model. Task
profiling and analytical benchmarking tools play an important role in providing the
right estimation of costs on tasks and edges, prior to scheduling [15]. The major
advantage of static scheduling is that the overhead of the scheduling process is
incurred at the compile-time, resulting in a more efficient execution time
environment compared to dynamic scheduling [5, 16].

In general, the objective of the static task scheduling is to assign the tasks of
a DAG onto the processors and order their execution so that task precedence
requirements are satisfied and a minimum overall Schedule Length (SL) or makespan
is obtained. A schedule is to be considered efficient if the schedule length is less as
well as the number of processors used is minimal.

1.3 Motivation

The efficient scheduling of tasks is paramount to maximizing the benefits of
executing an application in a DHCS. Because of this, the task scheduling problem
has been extensively explored and consequently a variety of scheduling algorithms
such as list-scheduling, clustering, task duplication-based scheduling and the
algorithms based on genetic approaches have been proposed in the past.
Interestingly significant amount of work has been carried out for the development of
task scheduling algorithms for homogeneous processors [3, 5, 12, 17, 18] and the
same has not been fully explored for heterogeneous processors [19-21].
List-scheduling algorithms generally perform well at a relatively low cost compared
to other categories. Clustering algorithms are generally well suited for the
homogeneous processors. Task duplication-based scheduling algorithms are
generally used when the program is communication intensive. Genetic algorithms are the most widely studied random search techniques for the task scheduling problem. They provide good quality of schedules, but their execution times are significantly higher than the other alternatives and hence they are applied in systems where optimal schedule is needed.

The list-scheduling algorithms such as, Dynamic Level Scheduling (DLS) [20], Heterogeneous Earliest Finish Time (HEFT) [21], Critical Path On a Processor (CPOP) [21] and the Heterogeneous Critical Parent Trees (HCPT) [22] are the well known and widely referred algorithms for heterogeneous processors. The major drawback of the DLS and the HCPT algorithms is that they do not consider scheduling a task between two previously scheduled tasks on the same processor even if enough free slots are available. The CPOP algorithm allocates all critical tasks onto a single processor in an attempt to minimize the total execution time. However, the selection of a processor in advance, for critical path tasks may not work well, owing to the involvement of precedence and communication cost constraints in calculating the start time of tasks on different processors. The HEFT algorithm though it uses insertion-based scheduling policy would select a task randomly when there is more than one task with the same priority. This may result in inconsistent schedules. One of the motivations of this research work is to devise new list-scheduling algorithms using simple task prioritization scheme to provide better schedules than the well known existing task scheduling algorithms.

The essence of clustering algorithms is to cluster tasks that communicate highly among themselves onto the same processor and thereby reduce the schedule length. The clustering algorithms developed for homogeneous systems have proved to be better than the list-scheduling especially when the task graph is communication intensive [23, 24]. However, applying the clustering technique to the heterogeneous systems is a difficult job. This is because, in the homogeneous systems, formation of cluster is generally based on communication cost and the duration of a communication depends only on the amount of data to be exchanged and the duration of a task execution depends only on the number of operations to be
performed. This is no longer true in the case of heterogeneous systems [25]. In the heterogeneous system, the duration of a communication cost depends on the speed of the network link taken and the duration of a task execution depends on the processor that will execute this task. Therefore, the techniques used for clustering the tasks for scheduling onto the homogeneous systems are not suited for the heterogeneous systems. Hence, in this research work, improving the existing task scheduling algorithms using clustering technique has not been considered.

Task duplication-based scheduling algorithms are preferred to reduce the completion time of an application by duplicating the task on more than one processor. It is an interesting approach that has been blended with both list-scheduling and clustering-based scheduling by various researchers. Task duplication-based scheduling algorithms have been studied extensively and many heuristics were developed for the homogeneous processors [26-29] and these algorithms form the basis for the development of task duplication-based scheduling algorithms for heterogeneous processors [30-34]. The algorithms such as the Levelized Duplication-Based Scheduling (LDBS) [30], Heterogeneous Critical Node First (HCNF) [31], Task duplication-based scheduling Algorithm for Network of Heterogeneous systems (TANH) [32], Heterogeneous Critical Parents with Fast Duplicator (HCPFD) [33] and the Dynamic Critical Path Duplication (DCPD) [34] are some well known duplication-based scheduling algorithms developed for completely connected heterogeneous processors.

There are two versions of LDBS algorithms and they are referred in this thesis as LDBS1 and LDBS2 algorithms. The major drawback of the LDBS1 and LDBS2 algorithms is that they have higher complexity than the other algorithms. The HCNF algorithm is similar to the LDBS2 algorithm and the difference between these two algorithms is that the HCNF algorithm schedules the critical path task in a level first, then it selects the task with higher execution times next. The main drawback of the TANH algorithm is that it imposes some conditions for obtaining optimal schedule, but these conditions are not practical in heterogeneous processors. Thus, devising a new algorithm which overcomes the drawbacks of the existing
algorithms is very much essential. Further, though the task duplication-based scheduling has been widely explored for the completely connected DHCS, only less attention has been given to the arbitrarily connected DHCS. Hence, devising a new task duplication-based scheduling algorithm for arbitrarily connected DHCS is also very important.

Genetic Algorithm (GA) is the most widely studied guided random-search technique for task scheduling problem. GA has been applied successfully to solve scheduling problems in many fields such as, job shop scheduling, task scheduling on to homogeneous and heterogeneous processors [35-41]. GA provides good quality schedules but, the execution times are generally higher than other alternatives. Hence, this kind of algorithms is generally employed where the optimal schedules are needed. GA has been widely explored for both the homogeneous and heterogeneous processors. Majority of the existing genetic algorithms combines the search capability of genetic algorithms with a well known list-scheduling heuristic to provide the best possible solution. The Problem Space Genetic Algorithm (PSGA) is one of the widely referred GA-based algorithm which uses earliest finish time heuristic to generate schedules and it is proved to be better than Genetic Simulated Annealing (GSA) [35] and Levelized Min Time (LMT) [42] algorithms. The major drawback of PSGA algorithm is that it does not utilize the insertion-based scheduling mechanism. Further, the development of task scheduling algorithm using GA-based approach combined with task duplication has not been fully explored for DHCS. Hence, devising a new algorithm which gives better performance than the existing algorithms and to explore the development of new task scheduling algorithm using genetic approach combined with task duplication are very much essential.

In recent times, Mobile Computing System (MCS) which consists of battery operated portable mobile computing devices (nodes) interconnected by wireless mediums are increasingly being used in many areas such as battlefields, for disaster management and peacekeeping operations [43]. The advancements in computing and communication technologies excel the mobile computing devices with the
potential to execute a larger application. Energy consumption is the key issue in MCS in addition to mobility of the mobile devices due to their unique features such as limited, irreplaceable energy sources and lifetime requirements [44]. To alleviate the problem of energy limitation, several hardware based techniques have been proposed and used by today’s mobile computing nodes [45-47]. Many emerging applications in MCS require distributed processing with considerable computation demands. Execution of independent tasks with an objective to minimize the execution time and energy consumption have been explored widely in MCS [48], whereas only very few investigations have been carried out for the execution of dependent tasks. Recently, an attempt has been made for the execution communication subtasks represented by a DAG onto a heterogeneous ad hoc grid environment to minimize the total energy consumption and consequently heuristics were proposed for the allocation of subtasks in heterogeneous ad hoc environment [49]. However, these heuristics do not address the node mobility issue. Moreover, these heuristics cannot be directly applied for executing a larger application in MCS. In MCS, minimization of energy consumption and the completion time of an application are equally important. Hence, developing an algorithm that will minimize either the energy consumed or the schedule length of the application or both is very essential. Since node mobility is a critical issue in MCS, the algorithm should handle this issue in order to provide fault tolerant execution of an application in MCS.

Thus, in the present scenario, developing various categories of task scheduling algorithms for DHCS to provide better performance than the existing task scheduling algorithms and a task scheduling algorithm for MCS which minimizes either the energy consumed or the schedule length or both have become indispensable.

1.4 Objectives of the Research Work

DHCS have the potential to provide high performance computing whenever a larger application is broken into tasks that can be distributed to the various machines for parallel execution. This potential can be realized only with an
efficient scheduling algorithm. In the recent years mobile computing systems are widely popular and are used in many areas. The execution of larger applications on MCS is not explored like the DHCS by the research community. Hence, this research work has two major objectives. The first objective of this research work is to develop various categories of task scheduling algorithms such as list-scheduling, task duplication-based scheduling and task scheduling using genetic approach for DHCS to provide better performance than the existing algorithms in each category. The second objective of the research is to formulate a new task allocation strategy for MCS and devise new task scheduling algorithm which minimizes the schedule length or the energy consumed to complete the application or both, based on the needs. Therefore, the research work is carried out with the following objectives:

(i) To provide a solution to the task scheduling problem, by devising new task scheduling algorithms for completely connected DHCS using list-scheduling approach to generate better schedules than the existing list-scheduling algorithms with lesser scheduling overhead.

(ii) To develop a new task duplication-based scheduling algorithm for completely connected DHCS to provide better results than the existing algorithms in the task duplication-based scheduling category.

(iii) To develop a new task duplication-based scheduling algorithm for arbitrarily connected DHCS to provide better results than the existing

(iv) To devise new genetic algorithms for task scheduling with task duplication and without task duplication for the completely connected DHCS.

(v) To formulate the task scheduling problem for MCS with the objective of minimizing the schedule length and the energy consumed to complete the application and provide solution by devising new task scheduling algorithm.

1.5 Problem Statement

Based on the objectives, the task scheduling problem is formulated. The task scheduling problem consists of three major components namely, a task graph model, a target system and a scheduling algorithm. The task graph model and the target
system are the input to the scheduling algorithm. The objective of the scheduling algorithm is to effectively schedule the tasks of the application on the target system to minimize the *schedule length*. The task graph model and the target systems (DHCS and MCS) considered in this research work are described below and then the task scheduling problem is formulated.

1.5.1 Task Graph Model

An application (parallel program) is divided into a set of tasks with some precedence relationships among them and can be represented by a DAG also called a task graph. A DAG is represented by the five-tuple \( G (V, E, P, T, C) \) where:

- \( V \) is the set of vertices \( v_i \), and each vertex \( v_i \in V \) represents an application task, which is a sequence of instructions that must be executed serially on the same processor in non-pre-emptive manner.

- \( E \) is the set of communication edges \( e(v_i, v_j) \), and each \( e(v_i, v_j) \in E \) represents an edge from task \( v_i \) to task \( v_j \). The edge from task \( v_i \) to task \( v_j \) also represents the precedence constraints such that task \( v_j \) cannot start until \( v_i \) finishes and sends its data to \( v_j \).

- \( P \) is the set \( \{ p_i : i = 1, \ldots, p \} \) of \( p \) heterogeneous processors.

- \( T \) is the set of computation costs \( T'(v_i, p_j) \), which represent the estimated computation times of task \( v_i \) on processor \( p_j \).

- \( C \) is the set of communication costs \( c(v_i, v_j) \), which represent the communication cost associated with the edges \( e(v_i, v_j) \).

Task \( v_p \) is a *predecessor* of task \( v_i \) if there is a directed edge originating from \( v_p \) and ending at \( v_i \). Likewise, task \( v_s \) is a *successor* of task \( v_i \) if there is a directed edge originating from \( v_i \) and ending at \( v_s \). Let \( \text{pred}(v_i) \) be the set of all predecessors of \( v_i \) and \( \text{succ}(v_i) \) be the set of all successors of \( v_i \). Without loss of generality, it can be assumed that there is one *entry task* and one *exit task* for a DAG, if there are multiple entry or exit tasks, the multiple tasks can always be connected through a dummy task which has zero computation cost and zero communication cost edges. A
sample task graph is given in Figure 1.2 and the computation costs of the tasks are given in Table 1.1.

![Sample Task Graph](image)

**Figure 1.2** A random task graph with 11 tasks [32]

<table>
<thead>
<tr>
<th>Task</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 )</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>( v_2 )</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>( v_3 )</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>( v_4 )</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( v_5 )</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>( v_6 )</td>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>( v_7 )</td>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>( v_8 )</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>( v_9 )</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>( v_{10} )</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>( v_{11} )</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

**Table 1.1 Computation costs of the tasks in Figure 1.2 on three processors**

### 1.5.2 Target Computing Systems

Two types of target computing systems are considered in this thesis:

i) A DHCS consists of a completely connected set of \( p \) processors and a DHCS consists of an arbitrarily connected set of \( p \) heterogeneous processors.

ii) A MCS with a set of \( p \) heterogeneous mobile computing processors (the term processors and nodes are used interchangeably in this thesis), in which a node communicates with other nodes in the system through the wireless communication media.

**Target Computing System: DHCS**

The target computing system DHCS consists of set \( P \) of \( p \) heterogeneous machines (processors) interconnected by heterogeneous high-speed communication links in a fully connected topology in which all inter-machine communications are assumed to be performed without contention. i.e., multiple messages can use the
same link simultaneously without affecting each other. Further it is assumed that, each machine has a separate input-output processor and hence communication and computation can be performed concurrently. It is assumed that memory space is not a problem and there is no fault in the system. Once a processor has started task execution, it continues without interruption and after completing the execution, it immediately sends the output data to all the children tasks in parallel. A completely connected distributed heterogeneous computing system with four processors is shown in Figure 1.3.

![Figure 1.3 Model of DHCS consisting of completely connected four processors](image)

The bandwidth (data transfer rate) of the links between different processors in a heterogeneous system may be different depending on the kind of network. The data transfer rate is represented by a $p \times p$ matrix $R_{p \times p}$. The communication cost between two processors $p_x$ and $p_y$, depends on the channel initialization at both sender processor $p_x$ and receiver processor $p_y$ in addition to the communication time on the channel and can be assumed to be independent of the source and destination processors. The communication startup costs of processors are given in a $p$-dimensional vector $S$. Let $Data(v_i, v_j)$ be the size of data to be transferred from task $v_i$ to task $v_j$ (the weight of the edge). The communication cost of the edge $e(v_i, v_j)$, which is for transferring data from task $v_i$ (scheduled on processors $p_x$) to task $v_k$ (scheduled on processor $p_y$) is defined by

$$c(v_i, v_j) = S_x + \frac{Data(v_i, v_j)}{R[p(x), p(y)]} = 0, \text{ when } x = y.$$  

(1.1)

where $S_x$ is the communication startup cost of processors $p_x$. 

12
In order to define the objective of the task scheduling problem, two attributes namely, Earliest Start Time (EST) and Earliest Finish Time (EFT) are defined. The EST of task $v_i$ on processor $p_j$ is represented as $EST(v_i, p_j)$. Likewise the EFT of task $v_i$ on processor $p_j$ is represented as $EFT(v_i, p_j)$. Let $EST(v_i)$ and $EFT(v_i)$ represent the earliest start time upon any processor and the earliest finish time upon any processor, respectively. For the entry task $v_{\text{entry}}$, the $EST(v_{\text{entry}}) = 0$, for other tasks in the task graph the EST and EFT values are computed starting from the entry task to exit task by traversing the task graph from top to bottom. To compute the EST of a task $v_i$ all immediate predecessor tasks of $v_i$ should be scheduled. The task scheduling problem is mathematically defined as follows:

$$EST(v_i, p_j) = \max\{P_{\text{available}}[v_i, p_j], \max(EFT(v_p, p_k) + C(v_p, v_j))\}.$$ 

where $v_p \in \text{pred}(v_i)$, $EFT(v_p, p_k) = EFT(v_p)$,

$$C(v_p, v_j) = 0 \text{ when } k = j.$$

$$EFT(v_i, p_j) = T(v_i, p_j) + EST(v_i, p_j).$$ \hspace{1cm} (1.2)

$P_{\text{Available}}[v_i, p_j]$ is defined as the earliest time that processor $p_j$ will be available to begin executing task $v_i$. The inner $\max$ clause in the $EST$ equation finds the latest time that a predecessor’s data will arrive at processor $p_j$. If the predecessor finishes earliest on a processor other than $p_j$, communication cost must also be included in this time. $EST(v_i, p_j)$ is the maximum of times at which processor $p_j$ becomes available and the time at which the last message arrives from any of the predecessors of $v_i$.

In the case arbitrarily connected DHCS, the computation of communication cost between the tasks, $C(v_p, v_j)$ in the Eqn. (1.2) is computed by finding the shortest communication path between the processors $p_j$ and $p_k$.

**Target Computing System: MCS**

Target computing system MCS consists of a set of heterogeneous mobile computing nodes (processors) equipped with wireless communication and networking facilities. There is a centralized access point or the control module through which every node can communicate directly with any node within the range of access point and a node can also communicate indirectly with those nodes outside
the range of access point. With indirect communication, other access points are used to relay (forward) data from source to destination. The MCS considered in this research work assumes the nodes are heterogeneous and they have computing power on par with the other computers. The nodes are close enough to each other so that single-hop communication is possible.

![Diagram of MCS](image)

Figure 1.4 Model of MCS consisting of four processors

Every node submits its profile information such as the type of processor, duration of availability, etc., to the control module (scheduler) when it joins the network. Similarly a node has to notify the scheduler whenever it leaves the network. Mobility of a node is permitted at any instance of time, but once started executing a task, it should complete the task execution and send the output to all the successor tasks scheduled onto different nodes. A separate communication component is assumed to be available in each node and hence a node can perform both communication and computation at the same time. Any initial data (i.e., data not generated during execution of the application task) is preloaded before the actual execution of the application task begins and a node consumes no energy if it is idle. Further, the energy consumed by the node for computation and communication is alone considered for energy minimization and the energy consumed by other components is assumed to be negligible. In order to define the objective of the task scheduling problem on MCS the following attributes have been defined and the problem is mathematically formulated.
$P = \{p_1, p_2, \ldots, p_p\}$ be the set of $p$ mobile nodes in the MCS.

$B(j)$ be the initial battery energy of the node $p_j$.

$R_{\text{comp}}(p_j)$ be the rate at which the node $p_i$ consumes energy for executing a task, per execution time unit.

$T(v_i, p_j)$ be the computation cost (time) of task $v_i$ on node $p_j$. Then the energy consumed for executing the task $v_i$ on the node $p_i$, $E_{\text{comp}}(v_i, p_j)$ is computed using the equation

$$E_{\text{comp}}(v_i, p_j) = T(v_i, p_j) \times R_{\text{comp}}(p_j).$$

$R_{\text{comm}}(p_j)$ be the rate at which the node $p_i$ consumes energy for transmitting one byte of data, per communication time unit.

$\text{Data}(v_i, v_j)$ be the amount of data to be transferred from the predecessor task $v_i$ scheduled on node $p_i$ to successor task $v_j$ scheduled and node $p_j$.

$C(v_i, v_p)$ be the communication time involved for receiving input data by task $v_i$ scheduled on $p_i$ from its predecessor task $v_p$ scheduled on $p_p$. $C(v_i, v_p) = 0$, when both the tasks are scheduled onto the same processor. Similarly task $v_i$ has to send the output data after completion to the successor task $v_i$ schedule on $p_i$. Hence, the Total Communication Time (TCT) involved by the processor $p_i$ for receiving input data from predecessor task of $v_i$ and sending output data to the successor task of $v_i$ is computed using the equation

$$TCT(v_i, p_j) = C(v_i, v_p) + C(v_p, v_j), \text{ where } v_p \in \text{pred}(v_i) \text{ and } v_j \in \text{succ}(v_i).$$

The total Energy Consumed for Communication (ECC) by a processor $p_i$ for sending and receiving data for task $v_i$ is computed using the equation

$$ECC(v_i, p_j) = TCT(v_i, p_j) \times R_{\text{comm}}(p_j).$$

The Total Energy Consumed (TEC) by a processor for computation and communication for the task $v_i$ is computed using the equation

$$TEC(v_i, p_j) = E_{\text{comp}}(v_i, p_j) + ECC(v_i, p_j) \text{ such that } TEC(v_i, p_j) < B(i).$$
Minimization of schedule length and energy consumption is equally important for achieving better performance in MCS. Two system attributes $\lambda$ and $\gamma$, such that $\lambda + \gamma = 1$, representing the relative weights of timing requirements (performance) and energy consumption respectively are introduced to fix the objective function. This decides the trade-off between schedule length and energy consumption. The $\lambda$ and $\gamma$ values are also used to map time units and energy units to generic cost unit. The problem is mathematically formulated as follows:

$$\text{cost} = \lambda \times EFT(v_i, p_j) + \gamma \times TEC(v_i, p_j) \quad \forall v_i \in V \text{ and } p_j \in P.$$  

(1.7)

The objective of the task scheduling algorithm is to minimize the cost, i.e., the schedule length or the energy consumption or both based on the values of $\lambda$ and $\gamma$. The $EFT(v_i, p_j)$ is computed using the Eqn. (1.2) and $TEC(v_i, p_j)$ is computed using the Eqn. (1.6).

When $\lambda = 1$ or $\gamma = 0$, then the cost function is minimization of $EFT$ of the exit task, which is nothing but the schedule length of the application.

$$\text{cost} = EFT(v_i, p_j), \text{ where } v_i \text{ is the exit task.}$$

When $\gamma = 1$ or $\lambda = 0$, then the cost function is minimization of total energy consumed by all the processors involved in the program execution and is given in the equation

$$\text{cost} = \sum_{i=1}^{V} \min \{TEC(v_i, p_j)\}. \forall p_j \in P.$$  

(1.8)

The relative importance given to minimizing the schedule length or energy consumption can be altered by varying the values of $\lambda$ from 0 to 1, or $\gamma$ from 1 to 0.

1.5.3 The Objective of the Scheduler

The objective of the task scheduling algorithm (scheduler) for the target computing system DHCS is to minimize the makespan or schedule length of the DAG. i.e., minimize the $EFT(v_i)$, where, $v_i$ is an exit task in the task graph. The objective of the task scheduling algorithm for the target system MCS is to minimize
the schedule length or the total energy consumed to complete the program represented by a DAG or both, i.e., to minimize the cost in Eqn (1.7) based on the weight values \( \lambda \) and \( \gamma \).

1.6 Summary of the Research Contributions

In this research work new task scheduling algorithms in various categories such as list-scheduling, task duplication-based and genetic approach have been developed for DHCS. Further the task scheduling problem for the MCS has been formulated and the solution is provided by developing a new task scheduling algorithm for MCS.

(i) List-scheduling algorithms namely, *High Performance task Scheduling (HPS)* algorithm and *Performance Effective Task Scheduling (PETS)* algorithm have been developed. The HPS and PETS algorithms are compared with the existing widely referred list-scheduling algorithms such as HEFT, CPOP, DLS and HCPT. The comparison based on the random task graphs and some real world numerical problems such as Gaussian elimination algorithm, Fast Fourier Transformation and the Molecular Dynamics code show that the HPS and the PETS algorithms outperform the existing algorithms in terms of various performance metrics.

(ii) Task duplication-based scheduling algorithm namely, *Highly Communicating and Dependant-Based Task Scheduling (HCDBTS)* algorithm has been developed for arbitrarily connected DHCS. The performance of this algorithm is compared with *Mapping Heuristic (MH)* and *Modified Mapping Heuristic (MMH)* algorithms. The comparison based on random task graph and some real world numerical problems show that the HCDBTS algorithm surpasses the existing algorithms in terms of various performance metrics.

(iii) Task duplication-based scheduling algorithm namely, *High Performance Duplication-based Compile-time Scheduling (HPDCS)* has been developed for the completely connected DHCS. The effectiveness of the HPDCS
algorithm is compared with the existing task duplication-based scheduling algorithm such as LDBS, HCNF and the DCPD algorithms. The comparison shows that the HPDCS algorithm is better than the existing algorithms in terms of various performance metrics.

(iv) Genetic Algorithms namely, Genetic Algorithm for Task Scheduling (GATS) and Duplication-based Genetic Algorithm for Task Scheduling (DGATS) have been developed for the completely connected DHCS. Both DGATS and GATS algorithms are compared with PSGA, HPDCS, HEFT and CPOP algorithms. The comparison results show that DGATS and GATS provide better results than the other scheduling algorithms.

(v) Task scheduling algorithm namely, High Performance task Scheduling algorithm for Mobile computing system (HPSM) has been developed for scheduling DAG structured applications onto the mobile computing system. The effectiveness of the HPSM is experimentally tested by simulation.

1.7 Organization of Chapters in the Thesis

The present chapter explores the research problem by recounting the distributed heterogeneous computing system, parallel program representation and task scheduling. The motivation, objectives and the problem statement of the research work are also formulated and explained.

Chapter 2 presents the survey analysis on task scheduling algorithms. This chapter is organized into seven sections. The first section introduces a preamble about the task scheduling. The classification of the existing task scheduling algorithms is briefly explained in the second section. Discussions in the third and fourth sections are focused on the existing heuristic-based scheduling algorithms random guided search-based algorithms respectively. The fifth section describes the mobile computing system with some related works in task scheduling. This section also points out the need for the development of task scheduling algorithm for scheduling the DAG structured application on the mobile computing system.
sixth section presents the extract of the literature survey which helps in defining the research problem. Finally, the summary of the chapter is given in section seven.

Chapter 3 describes the proposed HPS and the PETS algorithms. The time complexity analysis of the HPS and the PETS algorithms are derived and explained. The performance metrics used for evaluating the algorithms, task graph generator and simulation setup is also described in this chapter. Finally, the performance of the HPS and the PETS algorithms are compared with the existing algorithms and the results obtained by various experiments are presented.

Chapter 4 exemplifies the task duplication-based scheduling algorithm HPDCS developed for the DHCS consisting of completely connected processors. The time complexity of the algorithm is derived and explained. Subsequently, the performance comparison of HPDCS algorithm with the existing algorithms is presented. This chapter also presents the HCDBTS algorithm developed for the DHCS consisting of arbitrarily connected processors. Finally, the experimental results are compared with the existing algorithms and reported.

Chapter 5 describes the proposed genetic algorithms GATS and the DGATS. Followed by the description of the algorithms, the performance comparison of the GATS and DGATS algorithms with the existing genetic algorithm PSGA, list-scheduling algorithms HEFT and CPOP, and the task duplication-based scheduling algorithm HPDCS are reported.

Chapter 6 presents the task scheduling algorithm HPSM developed for the mobile computing system. Subsequently, the operation of the HPSM algorithm is illustrated with example. The experimental results with an objective to minimize either the schedule length or the energy consumption or both based on the weight value are presented. Rescheduling of the tasks at the run-time during node mobility is also presented and discussed in this chapter.

Chapter 7 concludes the research work by highlighting the findings that are facilitated to accomplish the objectives. The limitations of the research work have been identified to carry out the possible future research to improve further.