Chapter 6

An Energy-aware Task Scheduling Algorithm

In this chapter, we propose an energy aware task scheduling algorithm called Energy Aware Edge Priority Scheduling (EAEPS) for multiprocessor environments which aims to reduce power consumption by exploiting DVFS technique. The EAEPS algorithm is an energy aware version of our EPS (Edge Priority Scheduling) algorithm. The proposed algorithm reduces the energy consumption by zeroing the edges of high priorities. The idea presented here leads to the following useful contributions:

- The concept of determining priority of edges and choosing that edge which connects clusters involving high communication costs when perform clustering. This method of prioritization results in obtaining meaningful clustering that can consequently help in minimizing the energy consumption.
- The presented work fruitfully achieves minimization of energy consumption, as shown by the results in comparison to results obtained by such similar approaches. Energy consumption is a significant optimization criterion for the problem of energy aware task scheduling of a parallel application in multiprocessor environments.
- A simulation study is performed on the proposed algorithm and the results are presented for benchmark task graphs which show that the proposed algorithm outperforms the other algorithms.

6.1 Related Work

In this section, we discuss related work of power aware scheduling algorithms for parallel and distributed systems. This work proposes the EAEPS algorithm which is a clustering-based scheduling algorithm with the aim of reducing power consumption.

A lot of work has been done for task scheduling focusing on power optimization. For example, Terzopoulos and Karatza [81] discussed scheduling of independent tasks, and proposed power aware versions of Min-Min and Max-Min algorithms by using DFVS technique.

Zong et al. [82] proposed two energy efficient duplication-based scheduling algorithms such as EAD (Energy-Aware Duplication) and PEBD (Performance-Energy Balanced Duplication) algorithms for parallel tasks on homogeneous clusters that leverage DVFS to conserve energy dissipation in processors.

Lee and Zomaya [83] addressed the problem of scheduling precedence constrained parallel tasks on heterogeneous computing systems and proposed an ECS (Energy-Conscious scheduling) heuristic that uses DVS (Dynamic Voltage Scaling) to minimize energy consumption.

Hu et al. [84] presented an energy aware scheduling algorithm named EASLA for dependent tasks in the context of SLA (Service Level Agreement) on DVFS enabled cluster systems. The algorithm minimizes energy consumption by scaling frequencies down and distributing each slack to a set of tasks.

Aupy et al. [85] proposed several algorithms to solve the problem of scheduling precedence constrained tasks on homogenous computing systems that aim to minimize energy consumption while thinking about a given bound on the makespan and a reliability threshold.

Kimura et al. [86] proposed an algorithm which reduces energy consumption by using DVFS and reclaiming slack times for non critical tasks in parallel programs executed on real power-scalable PC clusters. The algorithm reclaims slack time by controlling the voltages and frequencies.

Kaur et al. [87] proposed a duplication controlled static energy efficient scheduling algorithm called C-SEED for scheduling of dependent tasks on heterogeneous computing systems. The algorithm controls duplications by using a threshold value with DPM (Dynamic Power Management) technique.

Mei et al. [88] proposed an Energy Aware scheduling by Minimizing Duplication (EAMD) algorithm which reduces energy consumption without degrading makespan. The authors claim that the algorithm not only is easier to operate than both DPM and DVFS, but also produces no overhead of time and energy.

Tang and Tan [89] proposed a reliability and energy aware task scheduling algorithm for precedence constrained parallel applications on heterogeneous systems. The algorithm uses a single processor failure rate model based on DVFS and maintains better tradeoff among reliability, performance, and energy consumption with lower complexity.

Sharifi et al. [90] presented a two phase power aware algorithm called PASTA for scheduling precedence constrained tasks on heterogeneous computing resources. The algorithm doesn't require any special hardware support to reduce power consumption and it provides good tradeoff between makespan and power efficiency.

Wang et al. [91] studied the slack time for non-critical tasks to minimize the energy consumption and considered the Green Service Level Agreement in their work. The authors proposed two power aware scheduling heuristics called PALS (Power Aware List Scheduling) and PATC (Power Aware Task Clustering) for parallel applications.

We propose a novel power aware task scheduling algorithm called Energy Aware Edge Priority-based Scheduling (EAEPS) for multiprocessor environments which aims to reduce power consumption by exploiting DVFS technique.

6.2 System Model and Problem Formulation

This section introduces the system models and formalizes the scheduling problem.

6.2.1 Application Model

A parallel application having precedence constrained tasks is modeled as a Directed Acyclic Graph (DAG), $G = (T, E)$, where T represents a set of nodes in which each node denotes a task and E represents a set of communication edges between tasks which show precedence constraints on T:

$$T = \bigcup_{1 \leq i \leq N} \{T_i\} \tag{6.1}$$

where, T_i is i^{th} task and N is the number of tasks in a parallel application.

$e_{i,j}$ denotes an edge between T_i and T_j, i.e. task T_i must be finished before T_j can start, where $1 \leq i, j \leq N$ and $T_i, T_j \in T$. Here T_j is called a successor of T_i and T_i is called a predecessor of T_j. When two tasks are executed on a same processing unit, their communication cost is ignored. It is assumed in this work that the DAG has only one start task and one end task. Multiple start (multiple-end) tasks are handled by connecting them with a start (end) task having no computation and communication cost. Fig. 2.1 gives a sample DAG representing the application model.

6.2.2 Processor Model

This work considers a multiprocessor system containing homogeneous processors or processing units that are fully connected with the same communication links. Each processing unit can simultaneously perform execution and communication of tasks. The execution of tasks performed by processing units is non-preemptive. Each processing unit is enabled with software controlled DVFS and can operate on a set of operating frequencies f and a set of supply voltages V.

$$f = \bigcup_{1 \leq i \leq K} \{f_i\} \tag{6.2}$$

$$V = \bigcup_{1 \leq i \leq K} \{V_i\} \tag{6.3}$$

where, f_i is the i^{th} processor operating frequency; V_i is the i^{th} processor supply voltage;

$$f_{min} = f_1 \leq f_2 \leq ... \leq f_K = f_{max}$$

$$V_{min} = V_1 \leq V_2 \leq ... \leq V_K = V_{max}$$

and K represents the number of operating points for the processing unit.

6.2.3 Energy Model

The energy consumption of a processor is due to processing, leakage, and short-circuits. According to Mei et al. [64], power is mostly consumed in the execution of the instructions. Therefore, the energy model, adopted here, considers energy consumption only due to execution ignoring other causes. The energy consumption of processor for task execution, ξ, is the summation of static energy consumption ξ_{static}, and dynamic energy consumption $\xi_{dynamic}$ [91]. The dynamic power consumption causes due to the charging and discharging process involved in the CMOS capacitances whereas the static power consumption is caused by the running, bias and leakage currents.

$$\xi = \xi_{dynamic} + \xi_{static} \tag{6.4}$$

According to [92], the dynamic power consumption $P_{dynamic}$ is determined as follows:

$$P_{dynamic} = A \times C_L \times V_{dd}^2 \times f \tag{6.5}$$

where, A is the percentage of active logic gates; C_L is the total capacitance load; V_{dd} is supply voltage and f is the operating frequency.

Now, dynamic energy consumption $\xi_{dynamic}$ can be computed as follows:

$$\xi_{dynamic} = \sum_{\Delta t} P_{dynamic} \times \Delta t \tag{6.6}$$

where, Δt is a time period.

According to [93], ξ_{static} is directly proportional to $\xi_{dynamic}$:

$$\xi_{static} \propto \xi_{dynamic} \tag{6.7}$$

Hence, the total energy consumption will be directly proportional to the dynamic energy consumption and can be computed as follows:

$$\xi = \sum_{\Delta t} (\eta \times V_{dd}^2 \times f \times \Delta t) \tag{6.8}$$

where, η is a constant determined by processing unit, V_{dd} is the operating supply voltage of processing unit during Δt, f is the operating frequency during Δt and Δt is the time period.

In our energy model, processors have several voltage and frequency levels, and a scheduling algorithm may choose the appropriate voltage and frequency to save energy. When a processor is idle or in a communication phase, it is assumed that the processor will operate at its lowest frequency.

6.2.4 Problem Formulation

Given a parallel application G consists of n precedence-constrained tasks and a number of processing units, find a schedule which minimizes overall energy consumption in multiprocessors.

6.3 The EAEPS Algorithm

This section presents an Energy Aware Edge Priority-based Scheduling (EAEPS) algorithm for multiprocessor environments. The proposed algorithm is an energy aware version of our EPS (Edge Priority Scheduling) algorithm which is a clustering-based scheduling algorithm and focuses on makespan minimization only. The classical clustering-based scheduling algorithm executes the following steps: (1) clustering

of tasks is performed by zeroing the edges; (2) mapping of clusters to appropriate processing units; (3) scheduling of the tasks. The classical clustering-based scheduling algorithm minimizes the makespan whereas our proposed algorithm reduces power consumption by exploiting DVFS technique.

6.3.1 Priority Function for the Edges

In classical clustering-based algorithm and PATC algorithm, edges are selected for zeroing on the basis of their communication costs while in this work we define a priority function for the edges same as defined in EPS algorithm as follows:

$$p(e_{i,j}) = \frac{CT(e_{i,j})}{ET(T_i) + ET(T_j)} \tag{6.9}$$

6.3.2 Voltage Scaling for Non-critical Tasks

In this section, we discuss how to scale down voltages of non-critical tasks with DVFS technique. This concept is the basis of the EAEPS algorithm given in the next section. A non-critical task is a task which does not belong to the critical path of a task graph. For scaling voltages on a non-critical task, say for T_i, we need to compute slack of T_i and that is computed using Eq. 2.10. After finding slack for T_i, we require to calculate execution time of task T_i which can be extended due to slack without violating precedence constraints and is calculated as follows:

$$ET_{slack}(T_i) = ET(T_i) + slack(T_i) \tag{6.10}$$

where, $ET(T_i)$ is the execution time of a task T_i on processor′s maximum frequency, $slack(T_i)$ is the slack for task T_i and $ET_{slack}(T_i)$ is the new execution time of T_i after considering its slack.

For a task T_i executed at frequency f_k, we denote the voltage as $V_k(T_i)$, the frequency as $f_k(T_i)$. When execution time of the task T_i is extended due to slack, the corresponding operating frequency of processor for task T_i is calculated as follows:

Algorithm 5 The EAEPS Algorithm

```
1: Initially one cluster for each of the tasks is formed
2: Calculate initial energy consumption
3: Compute priority of each edge as Eq. 6.9
4: Sort all edges in non-increasing order by their priorities and make a list
5: repeat
6:     for all edges in the sorted list do
7:         Zero an edge if energy consumption reduces
8:         When two clusters are grouped, the order among tasks is decided by
   comparing their bottom-levels with each other
9:         if bottom-level of one task is equal to the bottom-level of other task then
10:            Both tasks are ordered according to their topological-order in the clus-
   ter
11:        end if
12:        Update energy consumption
13:        break
14:    end for
15:    Remove an edge from sorted list by which energy consumption reduces.
16: until energy consumption decreases
```

$$f_k(T_i) = f_{max} ET(T_i) / ET_{slack}(T_i) \tag{6.11}$$

where f_{max} is the maximum frequency at which processor is executing task T_i.

When a processor executes a non-critical task, it first calculates its slack as Eq. 2.10 and then its new execution time as Eq. 6.10. After that it attempts to scale its operating frequency as Eq. 6.11. Each processor has corresponding voltage level for each frequency level.

6.3.3 The Algorithm

In this subsection, we formalize the proposed algorithm as Algorithm 5.

As shown in Algorithm 5, the EAEPS algorithm initially assigns each task to distinct cluster and determines initial energy consumption. After that, the algorithm computes priority of each edge and sort edges in non-increasing order of their priorities, the EAEPS algorithm repeatedly groups tasks by zeroing the edges with high priority if total energy consumption is not increased. When two tasks or clusters

TABLE 6.1: Frequency and supply voltages used in this work

Frequency (GHz)	**Supply Voltage (V)**
0.8	0.90
1.0	1.00
1.2	1.05
1.4	1.10
1.6	1.15
1.8	1.20

are grouped, the order among tasks is decided by their bottom levels. The bottom level of a task T_i is the longest path from T_i to the end task of the graph and is computed as definition 2.6.

In EAEPS algorithm, the priority function tries to group two tasks from different clusters that have heavy communication cost between them w.r.t their execution costs.

6.4 Experimental Results

In this section, we provide a simulation study on the proposed EAEPS algorithm. We considered some benchmark task graphs given by Davidovic and Crainic [79]. Table 6.1 shows the operating frequencies and supply voltages used in this work that is taken from [91]. The experiments are carried out on a Dell PowerEdge R420 server with CentOS (version 7.3-1611), Intel(R) Xeon(R) CPU E5-2420 v2 @ 2.20 GHz processor, and 192 GB of memory. The performance is measured in terms of percentage of energy saving that is defined as follows:

$$\xi = \frac{\xi_{max} - \xi_{algo}}{\xi_{max}} \tag{6.12}$$

where ξ_{max} is the energy consumption when all tasks executed at the maximum frequency, and ξ_{algo} is the energy consumption when applying a particular algorithm.

Table 6.2 gives a comparison of our proposed algorithm with other energy aware scheduling algorithms in terms of percentage of maximum energy saving. The

TABLE 6.2: Energy saving comparison of EAEPS with other existing energy-aware algorithms

Energy aware scheduling algorithms	**Minimum energy saving (%)**
EADUS & TEBUS [94]	11.43
Energy reduction algorithm [86]	18.35
ECS [83]	29.47
PATC [91]	30.52
EAEPS	31.48

EAEPS can achieve up to 31.48 % energy saving for selected benchmark random task graphs used in the simulation. Here, the algorithms EADUS and TEBUS give least energy saving among all compared algorithms as these algorithms are duplication-based and do not use DVFS to minimize energy consumption. The EAEPS gives more energy saving as it minimizes energy consumption during the communication phase. It also reduces energy consumption when a processing unit is idle. The EAEPS algorithm tries to choose edge for zeroing that involves more communication cost and reduces energy consumption.

6.5 Summary

We have proposed a power aware clustering-based task scheduling algorithm that makes use of priority function for edge zeroing, and we called it Energy Aware Edge Priority-based Scheduling (EAEPS) algorithm, for the problem of scheduling in multiprocessor environments. The proposed algorithm exploited DVFS technique to reduce the power consumption. The performance of the EAEPS algorithm is examined by performing a simulation study for some selected benchmark random task graphs. The experimental results show that the EAEPS algorithm achieves more energy saving that other compared energy aware scheduling algorithms. Future work includes the study and deployment of the proposed algorithm in some real-world applications like Gaussian Elimination, and Fast Fourier Transform.