Vol.66: e23210795, 2023
https://doi.org/10.1590/1678-4324-2023210795
ISSN 1678-4324 Online Edition

Article - Engineering, Technology and Techniques

Epigenomics Scientific Big Data Workflow Scheduling for Cancer Diagnosis in Health Care Using Heterogeneous Computing Environment

Wakar Ahmad[1*]
https://orcid.org/0000-0003-1876-9419

Bashir Alam[1]
https://orcid.org/0000-0003-0479-682X

Swati Sharma[2]
https://orcid.org/0000-0002-5961-4045

Arvinda Kushwaha[2]
https://orcid.org/0000-0003-2426-6375

[1]Jamia Millia Islamia, Department of Computer Engineering, New Delhi, India; [2]Meerut Institute of Engineering and Technology, Department of Information Technology, Meerut, UP, India.

Editor-in-Chief: Alexandre Rasi Aoki
Associate Editor: Raja Soosaimarian Peter Raj

Received: 08-Dez-2021; Accepted: 22-Jul-2022.

*Correspondence: waqar.ahmad50@gmail.com; Mob. +91-9560193998 (W.A.).

HIGHLIGHTS

- Analysis of different scheduling techniques for Epigenomics computation.
- An efficient algorithm that minimize computation time of Epigenomics Application.
- Performance analysis in High Performance Computing (HPC) environment.
- Shows better performance over state-of-art algorithms.

Abstract: DNA methylation and Histone are the main constituents to oversee the stable maintenance of cellular phenotypes. Any abnormalities in these components could cause cancer development and, therefore, must be potentially diagnostic. The Epigenomics research field is the study of epigenetic modification which involves gene expression control for better understanding of human biology. The Epigenomics applications are considered quite complex Big Data workflow applications which represent the data processing pipeline for automating the innumerable genome sequencing computation. The infrastructure of high-performance computing imparts heterogeneous computing resources for deploying such complex applications. Scheduling of workflow applications in the complex heterogeneous computing resources is considered an NP-complete problem; therefore, it requires an efficient scheduling approach. In this research work, an efficient list-based scheduling algorithm is proposed which efficiently minimizes the running time (makespan) of the Epigenomics application. In order to identify whether clustering and entry task duplication techniques improve the performance of the proposed algorithm, four versions of the algorithm such as list-based scheduling with clustering and duplication (LS-C-D), list-based scheduling with clustering and without duplication (LS-C-WD),

list-based scheduling without clustering and with duplication (LS-WC-D), and list-based scheduling without clustering and without duplication (LS-WC-WD) has experimented. The experimental results prove that LS-WC-D is the best choice for scheduling Epigenomics applications. Further, the comparison of LS-WC-D and state-of-the-art algorithms also proves its significance.

Keywords: Epigenomics; Big data; Workflow scheduling; Heterogeneous computing; Makespan minimization.

INTRODUCTION

"Cancer Genomics and Epigenomes" has made significant contributions to our understanding of the fundamental mechanisms underlying various cancers. Multistep tumorigenesis is a series of actions that occur as a result of signaling system dysregulation and changes in genomic information processing. Changes in the genes that influence unwanted cell division and growth that further produce cancer, which is a diverse genetic disorder.

If genetic alterations are hereditary due to germline modifications and existence in the germ cell, the probability of gaining cancer increases. These alterations can be found throughout the progeny's cells. The inaccuracies in process of DNA repair, that are produced by tobacco chemicals, smoke and radiation such as UV emissions, can create cancer-causing genetic alterations over a person's lifetime. Somatic or acquired alterations refer to genetic changes that take place after conception at any point during a person's life. Different genetic alterations, such as chromosomal transformations or mutations, can occur in tumor cells.

In Epigenetics, we study genetic alterations in the gene which occur without modification in DNA sequence. Between cell divisions, the alterations in gene expression remain constant [1]. Some of these non-genetic changes are mostly regulated by two biological modifications: chemical changes to DNA's cytosine residues, known as histone and DNA methylation. One of the primary epigenetic processes that lead to epigenetic modifications and suppression in malignant cells is the intricate interaction between DNA methylation and chromatin dynamics. These changes are important for the epigenetic inheritance of transcription memory. Epigenetic patterns change may be used to describe gene expression and its activity [2, 3]. The majority of epigenetic changes are global regulators of gene expression, influence cell characteristics and behavior, and have a significant impact on cancer progression.

A collaborative work between the National Human Genome Research Institute (NHRI) and the Cancer Institute has created a repository of 2.5-petabyte data explaining cancers paired with normal cells of more than 11,000 patient(s) on a detailed multi-map of key genomic changes in 33 distinct cancers. This dataset is publicly accessible to scientists (https://cancergenome.nih.gov/). The Portal for cBio Cancer Genomics is a publicly accessible cancer genomic dataset platform that provides full rights for accessing 20 different types of cancer having 5000 samples of the tumor. These initiatives decreased the barriers to the translation of significant data to biological views with clinical applications [4, 5] between large genomic diverse data and cancer researchers. The Colorectal Cancer Database, which provides information on the genes of 2056 Colorectal Cancer (CRC) linked to different phases of CRC, was recently developed by Agarwal and coauthors [6]. It helps physicians to understand the diagnosis, treating and classifying CRCs (http://lms.snu.edu.in/corecg/).

Next generation systems (NGS) like HiSeq, MiSeq, methylated DNA Immunosulfite, lowered bisulfite sequencing depiction, Ion Torrent PGM sequencer, methyl sequencing, chromatin immunosal, RNASeq, and array-based methods were recently used and have been very useful in the development of early diagnosis, forecasts, and res-based biomarker discoveries [7].

The USC Epigenome center uses medical Epigenomics workflow application for analysis of genome sequence of humans. With the help of high-performance gene sequencing systems along with IlluminiaSolexa genetic analyzer and MAQ software, it collects short DNA segments using automated operations [8].

Epigenomics applications are considered complex Big Data workflow applications which represent the data processing pipeline for automating the various genome sequencing computation. Workflow applications are depicted as a model of various scientific and technical problems such as high energy physics, astronomy, earth science, business modeling and so on. Workflow applications can range from having very few tasks that need a small number of computing resources to having millions of tasks that are interdependent on each other and need terabytes of storage, a large number of high-performance computing resources, and thousands of computing hours to finish. In order to complete the computation in a fair amount of time, such extensive Big Data workflow applications require a high-performance computing environment. The high-performance computing systems offer heterogeneous computing resources for deploying these complex

applications. Workflow scheduling in heterogeneous computing systems is considered an NP-complete problem; therefore, it requires an efficient scheduling approach. Many workflow scheduling approaches have been proposed but most of them fail to produce better results in minimum time. This research work proposes an efficient list-based scheduling technique that reduces the computation time of the Epigenomics workflow applications by incorporating task duplication and clustering techniques. The major contributions of this research work are listed below:

- The proposed algorithm encompasses the task duplication and clustering techniques to identify whether they improve the performance by minimizing the makespan of the Epigenomics workflow application or not.
- The proposed algorithm examines the different number of processors for identifying the behavior during the application computations.
- Numerous sizes of Epigenomics workflow applications are considered for experimental analysis ranging from 50 to 1000 tasks.

Further, the research work is divided into the following sections: In section 2, we have discussed previous studies related to our contribution. The workflow scheduling problems and their properties are discussed in section 3 and the working of task duplication and clustering techniques are discussed in section 4. In section 5, a detailed explanation of the proposed algorithm is given while in section 6, we provided a detailed analysis of experimental results. Finally, the conclusion and future work are given in section 7.

Related work

Over the years, there has been a lot of research into workflow scheduling. Many complex real-world problems can be represented as workflows, which need further analysis to solve problems. Various workflow scheduling techniques have been designed by scientists using numerous analytical or approximation methods. Most scheduling algorithms, however, suffer from a longer makespan and low resource utilization due to the restricted set of computational resources on the servers. The high-performance heterogeneous computing paradigm offers useful features to tackle complicated workflow scheduling problems, such as heterogeneous computing elements, easy and quick access, scalability, flexibility, and so on. Furthermore, there are many scheduling approaches are available for scheduling workflow applications in which the heuristic-based approach generates results in polynomial time with acceptable performance [9]. Heuristic-based scheduling approaches are further divided as list [10-17], duplication [18-20], and clustering based scheduling algorithms [21-23]. This section discusses the strengths and weaknesses of different heuristic-based workflow scheduling algorithms in a heterogeneous computing environment.

In list-based scheduling algorithms, workflow tasks are scheduled depending on their priority. The list scheduling algorithms are preferred by the scholars because they provide the best scheduling techniques with the least amount of complexity. Topcuoglu and coauthors [10] have proposed Heterogeneous Earliest Finish Time (HEFT) algorithm that is a widely accepted list-based scheduling algorithm. The HEFT algorithm uses a raked-based policy to allocate priority to each task in a workflow and then uses an insertion-based policy to distribute the tasks to the computing elements that can compute them in the shortest amount of time. The insertion-based policy attempts to place a task on a processor at the earliest possible time between two other tasks that have already been scheduled on the processor, provided that the available slot is large enough to accommodate the task. Hagras and coauthors [11] have developed a list-based scheduling technique called Heterogeneous Critical Parent Tree (HCPT) that uses a critical path optimization-based policy that tries to assign the parent of the critical task before computing itself. This approach generates an efficient schedule that helps to reduce makespan. Further, it produces efficient schedule length, which is better than HEFT, as well as maintaining the same time complexity. Arabnejad and coauthors [12] have proposed Predict Earliest Finish Time (PEFT) determines an optimistic cost table (OCT) for the task prioritization and processor assignment steps. The OCT table enables the algorithm to predict the selection of a machine that provides quicker completion times for the subsequent task. AlEbrahim and coauthors [13] have proposed an extended version of the HEFT algorithm called Ext-HEFT that priorities the workflow tasks by taking the ratio of differences among the maximum computation time and minimum computation time of the task on given processors divided by speedup of the processors on which task is assigned. For the processor assignment phase, a randomized crossover technique has been introduced. In the crossover technique, a task is allocated to the computing resource that generates the lowest computation time rather than the computing resource that gives the shortest finish time. The crossover technique uses a cross-threshold value that lies between 0 to 1. If its value is nearer to 1, more crossover will occur during

computation and thus behave like the HEFT algorithm, while a value near to 0 does not allow frequent crossover. Jiahui Wang and coauthors [14] proposed a list-based heuristic algorithm called cost-efficient scheduling with deadline constraint (CESDC) that achieves deadline constraint by adopting a deadline distribution policy. CESDC separates the tasks into a number of Bag of Tasks (BoTs) [15] based on their levels and assigns a deadline to each task based on its computation and communication time. Further, it is based on two-phase policy i.e., ranking and task prioritization of list-based scheduling approach. Belgacem and coauthors [16] suggested a method called HEFT-ACO that is a hybrid version of two algorithms that optimizes both cost and time. The ACO approach is used to improve resource allocation, whereas HEFT is used to deal with workflow task dependencies. The pheromone update criteria have been developed to deal with no dominating solutions, preserving variation and assuring research efficiency. Furthermore, this method utilized the Pareto technique and crowding distance, making it effective at solving the multi-objective optimization problem. Ijaz and coauthors [17] developed multi-objective optimization scheduling technique that works on finding tradeoff between makespan and energy which are conflicting objectives. At first, a weighted cost function is used to choose a processing node that completes tasks quickly and uses the least amount of energy based on a weighting factor set by the user. In the next step, we reduce the energy consumption even more by using frequency scaling which is associated with deadline-constraint. The proposed work that has been presented guarantees that the workflow applications will be finished within the time limit that has been proposed while also lowering the amount of energy that will be consumed.

In parallel systems, two of the most common ways to schedule tasks based on heuristics are clustering and duplication. The scheduler uses the clustering method to reduce communication costs by grouping together communication-intensive dependent tasks and assigning them to the same VM/processor [18-20]. The scheduler improves the parallelism degree in the duplication strategy by assigning a key subtask on many processors [21-23].

Based on the above discussion we can easily analyze that heuristic-based scheduling approaches play an important role for scheduling Big Data workflow applications in order to achieve various objectives such as minimization of makespan and maximization of resource utilization. In this paper, we take the benefit of all three heuristic techniques (list, duplication and clustering) and identify which combination of heuristic techniques is best for scheduling Epigenomics workflow applications.

Workflow scheduling problem

The workflow application is characterized by a directed acyclic graph $G = (T, E)$. Where $T = (t_1, t_2, \dots, t_n)$ a collection of is interdependent tasks and $E = (e_1, e_2, \dots, e_m)$ denotes data dependencies across tasks. For example, an edge $e_{i,j} \in E$ reflects a restriction on the order of execution between tasks t_i and t_j where $t_i, t_j \in T$ that means task t_j can only start after the computation of its parent task t_i and all required data has been received from it. All immediate predecessors (parents) of task t_j is represented by $pred(t_j) = \{t_i \,|e_{i,j} \in E\}$, likewise, all immediate successors (children) of task t_i can be denoted as $succ(t_i) = \{t_j \,|e_{i,j} \in E\}$. It is not necessary that workflow application has an entry task or exit task. Whenever a workflow has many entry or exit tasks, then we add a dummy entry task (t_{entry}) and dummy exit task (t_{exit}) with zero computation time and zero data dependency edge [24]. The entry task (t_{entry}) has no parent task; similarly, the exit task (t_{exit}) has no child task. Generally, tasks are scattered randomly throughout the workflow application. In order to better analysis of workflow applications, it is required to arrange the tasks level-wise. A task's level is a numerical representation of the group to which it belongs. Since the level of each task is calculated by adding 1 to its parent tasks' highest levels, the level of the entry task (t_{entry}) is always 0. i.e., $max_{t_j \in pred(t_i)} \{L_{t_j}\} + 1$. Where, L_{t_j} represents the level of a parent task t_j [25].

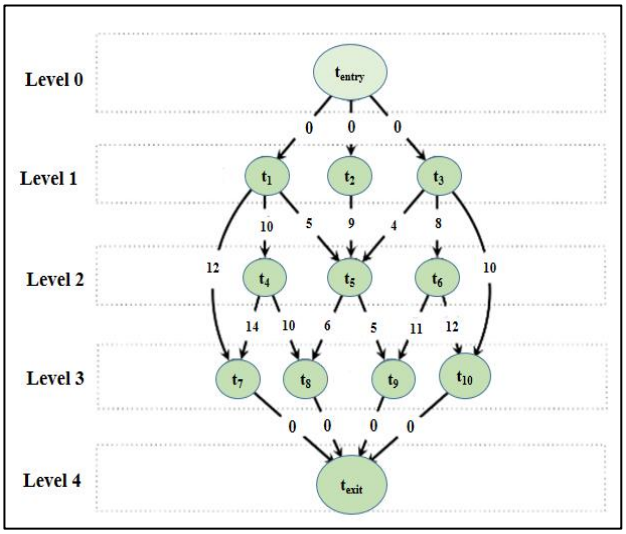


Figure 1. A sample workflow application

Table 1. Computation time of each task of sample workflow on the set of processors

Processors	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
p_1	11	23	19	12	9	3	6	8	9	13
p_2	15	18	12	7	12	15	14	10	11	5
p_3	7	16	17	15	7	8	9	16	18	20

Further, we assume that all the processors in the processor-list are completely connected to each other and there is no network latency. A sample workflow application model is depicted in Figure 1 and the computation time of each workflow task on different processors is given in Table 1.

It's important to recognize several key elements of task scheduling problems. These key elements are listed below:

Definition 1. A task's computation time is determined by dividing the number of instructions in the task by the processor's frequency. Consider the matrix W $(t \times p)$, where the matrix element represents computation time of a task $t_i \in T$ on a group of processor p i.e. $N = (n_1, n_2, \ldots, n_p)$. Thus the computation time of task ti is mathematically calculated as [12]:

$$\overline{w_i} = \left(\sum_{j=1}^{p} w_{i,j} \right) / \, p \tag{1}$$

Definition 2. Makespan of workflow application is the actual finish time (AFT) of the exit task [12]. That means, makespan is the maximum time it takes to complete all of the workflow application's tasks.

$$Makespan = AFT(t_{exit}) \tag{2}$$

Definition 3. The earliest start time of the task t_i on processor p_j is represented by $EST(t_i, p_j)$ [23]. The mathematical representation of EST is as follows:

$$EST\left(t_i, p_j\right) = max\left\{ T_{Available}\left(p_j\right), max_{t_m \in pred(t_i)} \left\{ AFT\left(t_m\right) + c_{m,i} \right\} \right\} \tag{3}$$

Where $T_{Available}\ (p_j)$ is the processor's earliest available time, and $max_{t_m \in pred(t_i)}\{ AFT(t_m) + c_{m,i} \}$ is the maximum value of data arrival time from its parent tasks. It is important to keep in mind that if task t_m is allocated to p_j then $c_{m,i}$ becomes zero.

Definition 4: The earliest finish time $EFT(t_i, p_j)$ of the task t_i on the processor p_j is equivalent to the sum of EST and computation time of task t_i on the processor p_j [23]. i.e.,

$$EFT\left(t_i, p_i\right) = EST\left(t_i, p_i\right) + w_{i,j} \tag{4}$$

Definition 5: To determine the upward rank of a particular task t_i, we must move the workflow in the upward direction. Basically, it is the longest path between task t_i and exit task t_{exit}. The mathematical representation of upward rank is [10]:

$$rank_u\left(t_i\right) = \overline{w_i} + \max_{t_j \in succ(t_i)} \left(\overline{c_{i,j}} + rank_u\left(t_j\right) \right), \tag{5}$$

Where $succ(t_i)$ is the set of immediate successors of task t_i. $\overline{c_{i,j}}$ represents average data transfer time and $\overline{w_i}$ is the average computation time of the task t_i.

Clustering and task duplication techniques

In this section, a detailed explanation of clustering and duplication techniques is discussed. These techniques are used in our proposed algorithm to check whether they improve the performance of the algorithm or not. For this purpose, four versions of the proposed algorithm such as list-based scheduling with clustering and duplication (LS-C-D), list-based scheduling with clustering and without duplication (LS-C-WD), list-based scheduling without clustering and with duplication (LS-WC-D), and list-based scheduling without clustering and without duplication (LS-WC-WD) are suggested. Further, a detailed explanation of the proposed algorithm is discussed in the next section.

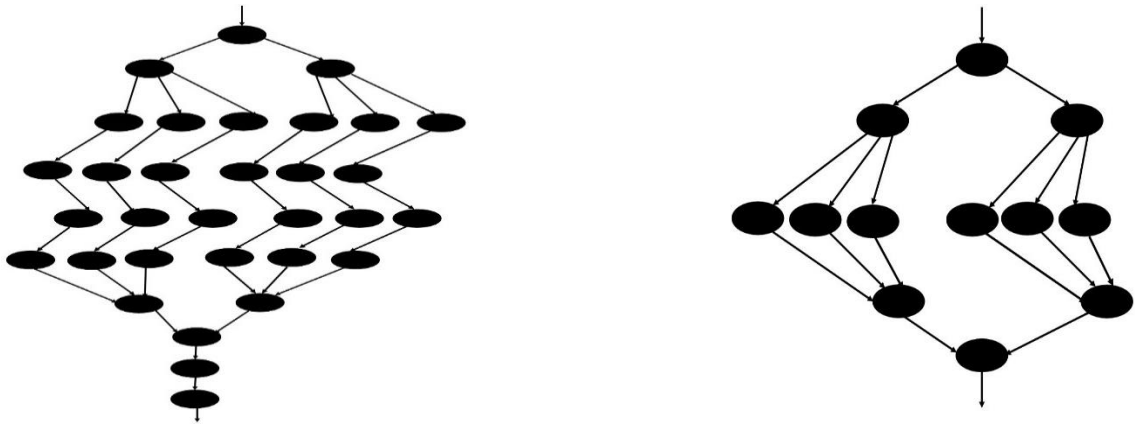
Clustering of Epigenomics workflow tasks

The main aim of clustering algorithms [18, 19, and 20] is to assign a set of tasks to the processor that has high communication costs with other processors. These types of task allocation to the processor have been done in order to reduce the data transfer time even if other processors are available. Thus, it does not correspond to the effective utilization of the available resources. In general, there are two phases of clustering algorithms. In the first phase, tasks that have higher communication costs are combined into a set of clusters. In the second phase, the set of clusters is scheduled on available computing resources. In order to reduce the runtime of the Epigenomics workflow application, we proposed a clustering algorithm. It significantly reduces data transfer costs between pipelined tasks which leads to increase in the algorithm performance. The clustering technique is mentioned in Algorithm 1 which allows child tasks to run on the same processor on which their parents were computed allowing them to use the intermediate results.

Algorithm 1: genome_Clustering (W(t, e))

1. **begin**
2. $task_{queue} \leftarrow \{task_{entry}\}$
3. **While** $(task_{queue} != Empty)$ **do**
4. $task_{parent} \leftarrow task_{queue}(front)$
5. $S_{child} = \{task_{child} | task_{child}$ is the child of $task_{parent}\}$
6. **if** $Cardinality\ (S_{child}) = 1$ and $task_{child}$ has only one parent $task_{parent}$ **then**
7. Replace $task_{parent}$ and $task_{child}$ with $task_{parent+child}$
8. Set $task_{parent+child}$ as the parent of $task'_{child}s$ children tasks
9. Update computation time of $task_{parent+child}$
10. Add $task_{parent+child}$ to the front of $task_{queue}$
11. **else**
12. Add $\text{task}'_{\text{parent}}s$ children to the rear of $task_{queue}$
13. **end if**
14. **end while**
15. **end**

Figure 2 shows the process of clustering technique on the Epigenomics workflow application. The LS-C-D and LS-C-WD versions of the proposed algorithm are applicable for the clustering technique.



(a) Epigenomics workflow before clustering **(b)** Epigenomics workflow after clustering

Figure 2. Clustering process of sample Epigenomics workflow

Entry task duplication on processors

The schedule length (makespan) of workflow application is minimized by using a conventional duplication approach. However, as noted in the literature [26], it strengthens the power consumption and processor overloading. In the condition, when only the entry task is duplicated, access power consumption and processor overloading issues do not arise. This is because only one processor from the processor set entails in the computation of the entry task. As a result, duplicating entry task on certain processors doesn't augment overload situation.

Since we have discussed in section 3 that entry and exit tasks are added when there are multiple entry and exit tasks present in the workflow. But in the case of Epigenomics workflow applications, there is only one entry and exit task with some computation and communication costs (see Figure 2). Hence, we can apply the entry task duplication technique as mentioned in Algorithm 2. The LS-C-D and LS-WC-D versions of the proposed algorithm are applicable for the duplication technique.

Algorithm 2: entry Task Duplication (t_{entry})

1. **begin**
2. Allocate task t_{entry} to the processors $proc_{min}$ that produces minimum EFT.
3. $dataArrivalTime = max_{t_p \in succ(t_{entry})}\{(EFT(t_{entry}, vm_{min}) + TT(e_{entry,p})\}$
4. **foreach** processor p_i in the processor$_{\text{set}}$ excluding p_{min} **do**
5. **if**$(dataArrivalTime > ET(t_{entry}, vm_{min}))$ **then**
6. Duplicate the entry task t_{entry} to the processor p_i
7. **end if**
8. **end For**
9. **end**

Proposed list-based scheduling algorithm

Algorithm 3 represents the pseudo-code for the suggested algorithm. It starts by calling a clustering technique for the Epigenomics workflow, which is applicable for LS-C-D and LS-C-WD versions of the algorithm (steps 2). In the next step, it makes a priority list that accommodates the tasks in the increasing order of the upward rank (steps 3-4). A while loop is maintained which selects the tasks from the priority list for scheduling on the processors. The loop is terminated when it gets empty. If the entry task is encountered during iteration, it calls the entry-task duplication method, which is applicable for LS-C-D and LS-WC-D versions of the algorithm (step 8). If the current unscheduled task is other than an entry task, the algorithm finds the last parent of the unscheduled task and its assigned processor (steps 11-12). In step 13, the

algorithm determines the data arrival time to the current unscheduled task. If the data arrival time is equivalent to its parent's processor available time, then the current unscheduled task is assigned to it for the computation (steps 14-15). In this way, the data transfer time between parent and child task is neglected, and thus performance is magnified. If the condition given in step 14 is not satisfied, then the algorithm finds the processor in the processor set that computes the current unscheduled task at the earliest time (steps 16-21). In step 23, the algorithm updates the task priority list accordingly.

Algorithm 3: Proposed List-based Scheduling Algorithm

Input: Set of Epigenomics workflow tasks and set of processor in $\text{processor}_{\text{set}}$
Output: Schedule length (makespan) of the workflow

1. **begin**
2. Call $pre - clustering\ (W(t,e))$
3. Compute $rank_u$ of each interconnected task of the workflow
4. Insert workflow tasks in $priority - list$ based on increasing values of $rank_{proposed}$
5. **while** $(priority - list\ != empty)$ **do**
6. Select the task t_i from $priority - list$
7. **If** task t_i is an entry task t_{entry} **then**
8. Call $entryTaskDuplicaiton(t_{entry})$
9. **else**
10. Task t_i is not an entry task t_{entry}
11. $getLastParent = arg\ \{max_{t_p \in pred(t_i)}(EFT(t_p))\}$
12. p_{parent} = processor on which the last parent was assigned
13. $dataArrivalTime = \max\{(EFT(getLastParent\ ) + \ max_{t_p \in pred(t_i)}\{EFT(t_p) + TT(\ e_{p,i})\}\}$
14. **if**$(dataArrivalTime == AT_{p_{parent}})$ **then**
15. Assign task $t_i\ on\ p_{parent}$ and calculate its EST and EFT
16. **else**
17. **foreach** processor p_i in the $\text{processor}_{\text{set}}$ **do**
18. Calculate EFT of the task t_i
19. **end for**
20. Processor p_i that has minimum EFT is selected for the assignment of task t_i
21. **end if**
22. **end if**
23. Update $priority - list$
24. **end while**
25. **end**

Performance evaluation of proposed algorithm

The suggested algorithm's performance is evaluated using Epigenomics workflow applications running on heterogeneous computer platforms. Numerous sizes of Epigenomics workflows are considered in the experiment to identify usefulness of proposed algorithm over existing competitive algorithms. Two experiments are performed in which the first experiment is to identify the best version of the proposed algorithm, whereas the second experiment is performed to observe the performance of suggested algorithm over state-of-the-art algorithms such as HCPT [11], PEFT [12], and the latest algorithms Ext-HEFT [13], LSTD [23].

Comparison Metrics

The following metrics are considered for the performance evaluation of the proposed algorithm.

Schedule Length Ratio (SLR): One of the most key determinants for measuring the quality of workflow scheduling approaches is the scheduled length (makespan). Since we need to take into account a large number of workflows, each of which has its own unique characteristics and structure. During the experiment, the schedule length must be reduced to a lower limit, which is called the schedule length ratio [12]. SLR is represented mathematically as follows:

$$SLR = \frac{makespan}{\sum_{t_i \in CP_{MIN}} \min_{p_j \in P} (w_{(i,j)})} \quad (6)$$

The denominator of SLR is the minimal computing cost of critical path tasks (CP_{MIN}). The critical path in a workflow is the path that is recognized as the longest path from the entry task to the exit task. Thus, no makespan can be achieved less than denominator of SLR equation. As a result, the best algorithm is the one with the lowest SLR.

Efficiency: The ratio of speedup to the number of processors is used to measure the efficiency of a heterogeneous system, where speedup signifies the ratio of sequential computation time divided by parallel computation time of workflow applications. The sequential computation time of workflow application is measured by summing the computation time of the tasks after assigning them to a single processor. Thus the mathematical formula of the speedup is mentioned below [12]:

$$Speedup = \frac{\min_{p_j \in P}[\sum_{t_i \in T} w_{(i,j)}]}{makespan(solution)}$$

Hence,

$$Efficiency = \frac{Speedup}{Number\ of\ Processors}$$

Average Running Time: The time spent by the algorithms to obtain a schedule for a given workflow application is known as the running time or (makespan). This measurement can be used to calculate the average computation time of algorithms.

Number of occurrences of better schedules: The results are compared with other competitive strategies using a paired table to recognize the proportion of better, equal and worst results obtained by the proposed technique.

Simulation Setup

All algorithms are coded in the python language and tested on the system having Ubuntu OS (18.04), 2.70 GHz Core i5 (Intel) processor and 8 GB RAM. The workflow generator program available at [27] generates different sizes of Epigenomics workflows. Workflow size varies from 50 to 1000 tasks. For producing the workflow application, the following factors are considered:

No. of Tasks = [50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
CCR = [0.1, 0.5, 1, 2, 5, 10]
Processors = [2, 4, 8, 16, 32, 64]

Here the term CCR refers to ratio of total edge costs (communication time) to the total task computation costs (compute time) in a particular workflow application. The CCR parameter is generally used to represent a wide range of computing machines. The lower value of CCR represents high-speed computing machines while a higher value of CCR represents slow computing machines. The combination of these parameters will generate 462 Epigenomics workflow applications. For each size of the Epigenomics workflow, 10 different workflows are generated with different computation costs and communication costs. In this way, there are 4620 Epigenomics workflow applications are considered for identifying effectiveness of the suggested algorithm in the heterogeneous computing environment.

Evaluation of Experimental results

This section summarizes experimental outcomes of proposed algorithm along with its competitive algorithms. There are three types of experiments conducted in which the first experiment identifies the best version of the suggested algorithm, the second experiment demonstrate the performance improvement of

the best version of the suggested algorithm over competitive algorithms and the third experiment analyze the number of occurrence better schedule.

Experiment 1: *For finding the best version of the proposed algorithm*

Figure 3 shows how long different versions of the proposed algorithm take to run on average as the size of Epigenomics workflow tasks grows. The outcome of the experimental results shows that the LS-WC-D version (list scheduling without clustering and duplication) performs well over other versions of the proposed algorithm. It shows 27.84%, 28.42%, and 1.65% improvements over LS-C-D, LS-C-WD, and LS-WC-WD versions of the proposed algorithm, respectively.

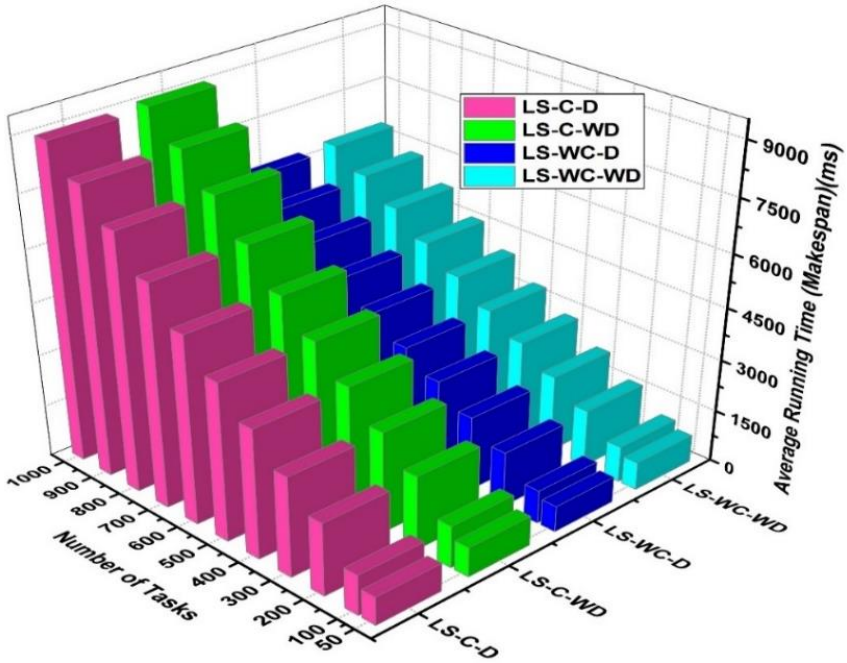


Figure 3. Average running time of the different versions of the proposed algorithm with respect to increasing size of the Epigenomics workflow applications.

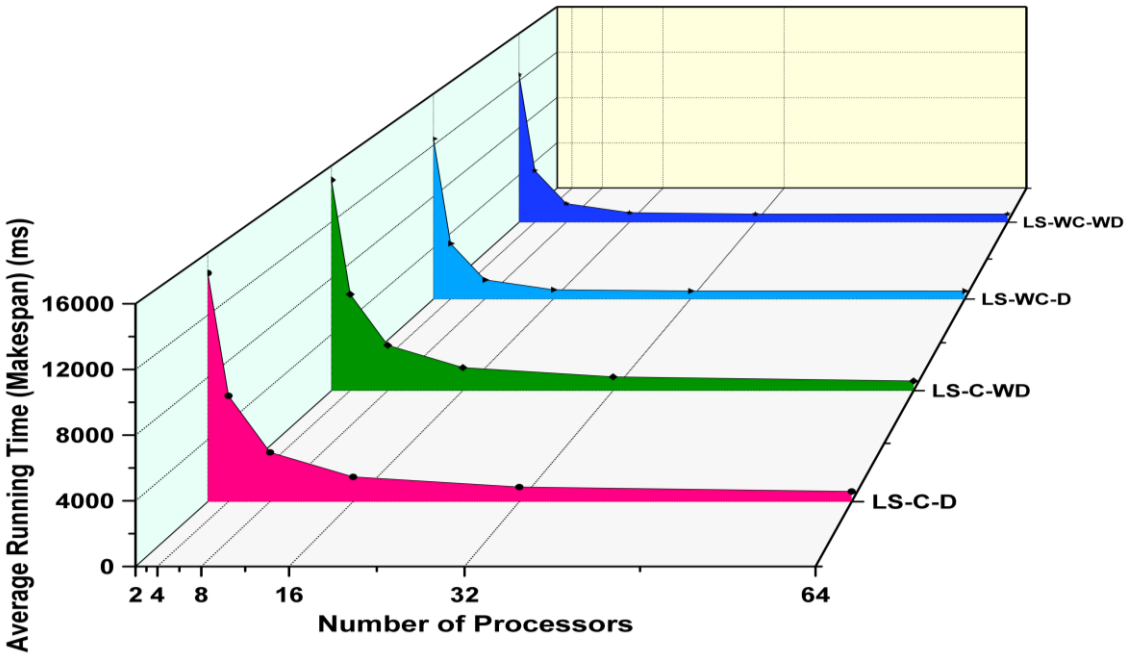


Figure 4. Average running time of different versions of the proposed algorithm with respect to increasing number of processors.

Figure 4 depicts the average amount of time it takes for the different versions of the proposed algorithm to run with regard to an increased number of processors. As we can see in the figure, there are variations in the makespan achieved by the different versions of algorithms with a low number of processors, but in the case of increasing number of processors, the performance of all versions becomes almost the same. This is because, as the number of processors grows, the scheduling policy of all versions has many options to choose the best processor for the assignment of tasks. If we talk about the overall performance of the algorithms, the LS-WC-D version further shows improvement over other versions in terms of average running time. It shows 25.14%, 26.42%, and 2.04% improvement over LS-C-D, LS-C-WD, and LS-WC-WD versions of the proposed algorithm, respectively.

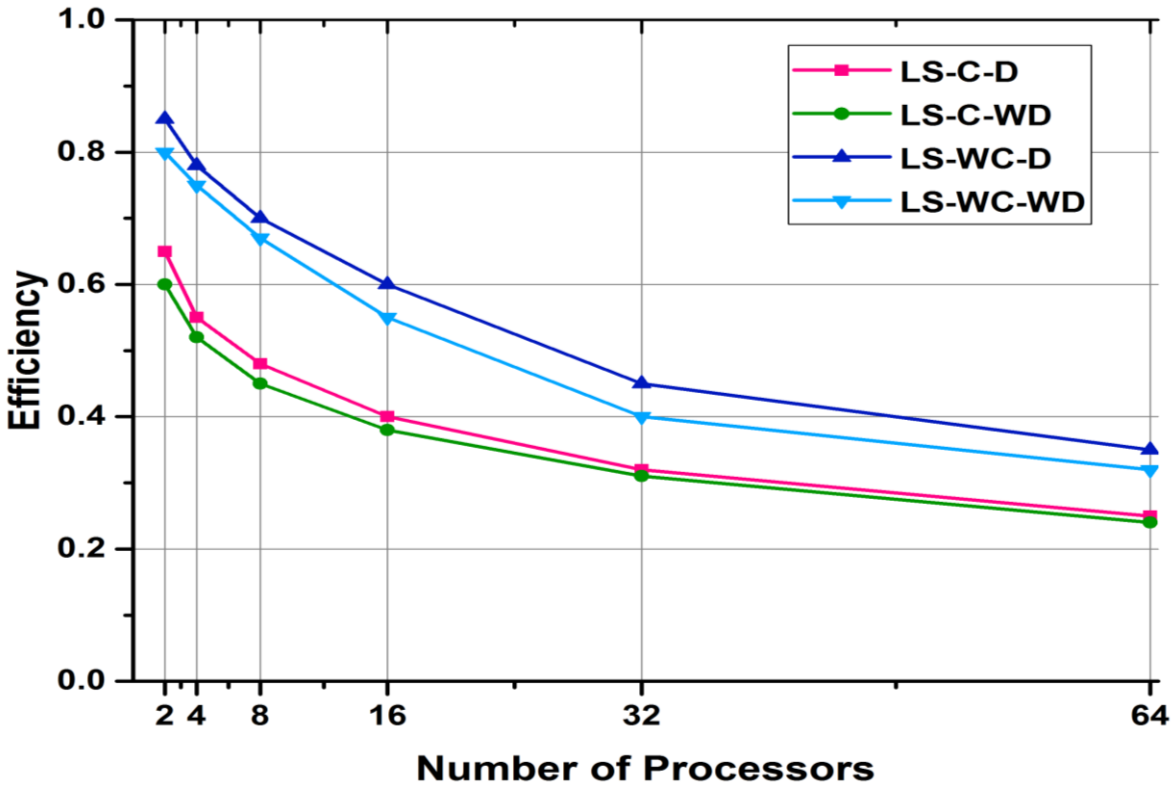


Figure 5. Efficiency of different versions of the proposed algorithm with respect to increasing number of processors.

Figure 5 shows how well the proposed algorithm works in its different versions as the number of processors increases. As we can see in the figure, the LS-WC-D shows higher efficiency throughout the experiment. However, with the increasing number of processors, the gap between efficiency achieved by LS-WC-WD and LS-WC-D becomes minimal. Also in the case of LS-C-D and LS-C-WD, they achieved almost equal efficiency with a higher quantity of processors. Thus, we can conclude that with a higher number of processors the resource utilization becomes minimal. Further, the LS-WC-D shows 28.72%, 27.23%, and 1.75% more efficiency than the LS-C-D, LS-C-WD, and LS-WC-WD versions of the proposed algorithm, respectively.

Figure 6 signifies average SLR of the scheduling approaches with increasing number of CCR. Initially, all versions of the algorithms show similar performance with the lower value of CCR. Here, lower value of CCR represents a high-performance computing system. With the increasing value of CCR, the LS-WC-D algorithm achieve a lower value of average SLR while LS-C-WD achieves a higher value of SLR. The lower value of average SLR represents the best schedule generated by algorithms for scheduling workflow applications. Further, experimental results show that LS-WC-D again shows better performance in terms of Average SLR.

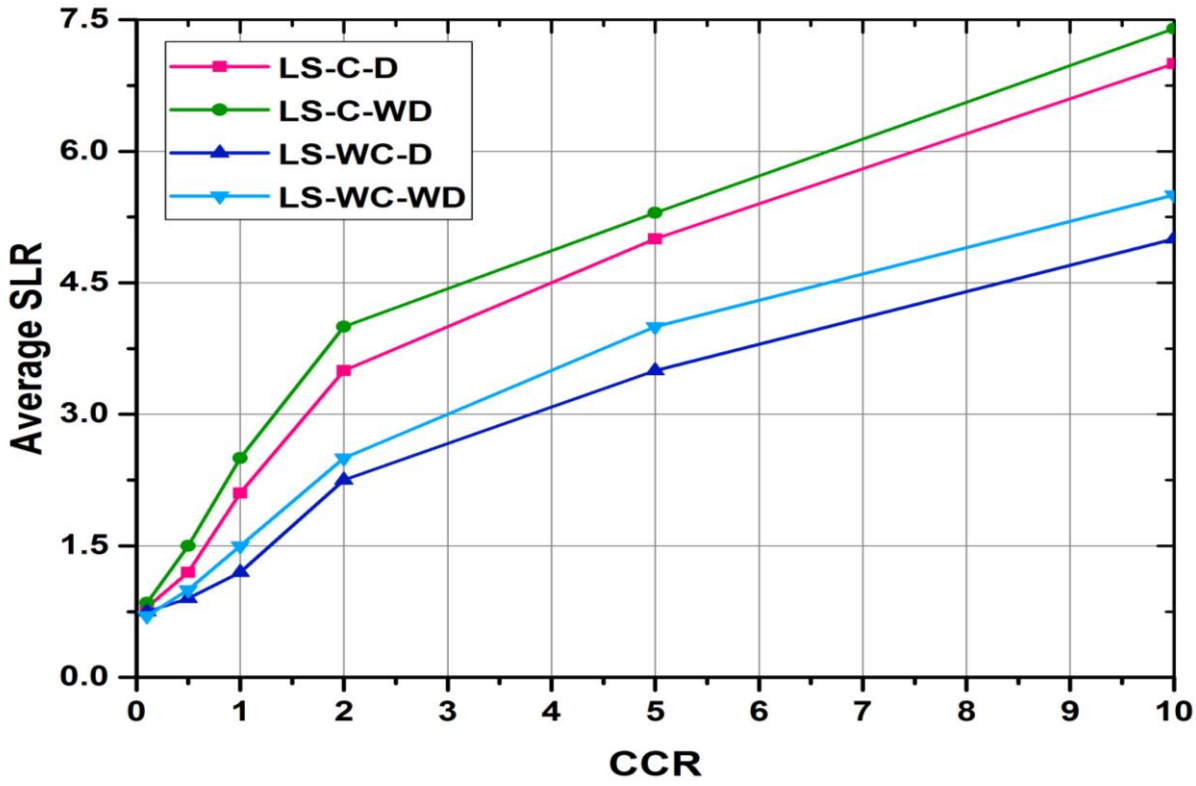


Figure 6. Average SLR of different versions of the proposed algorithm with respect to increasing value of CCR

After analysis of all experiment results, we observed that LS-WC-D (List Scheduling without Clustering and Duplication) version of proposed algorithms is the best choice for scheduling Epigenomics workflow applications. Thus, the clustering strategy we recommended does not contribute to better outcomes, but that the duplication technique is a preferable option for achieving better results.

Experiment 2: *Performance analysis of LS-WC-D with competitive algorithms*

In the last experiment, LS-WC-D is identified best version among the other versions of the proposed algorithm for scheduling Epigenomics applications in the heterogeneous computing environment. The performance of LS-WC-D is now examined further using state-of-the-art algorithms (HCPT and PEFT) as well as new suggested algorithms (Ext-HEFT and LSTD). A detailed analysis of experimental results is discussed below:

Figure 7 shows the average amount of time it takes for an algorithm to run as the number of tasks in the Epigenomics workflow increases. The HCPT takes a longer time to compute the Epigenomics applications as compared to other algorithms, while LS-WC-D produces the shortest schedule. However, LSTD gives tough competition to the LS-WC-D but when we calculate the overall average running time, the LS-WC-D produces

3.34% better schedule than LSTD. Also, PEFT and Ext-HEFT show almost similar performance but do not achieve the performance as compared to the LS-WC-D algorithm. Further, LS-WC-D produces 8.34%, 10.15% and 39.37% better schedule than HEFT, PEFT, and HCPT algorithms.

Figure 8 shows how long algorithms take to run on average as the number of processors increases. When we schedule Epigenomics applications of varying sizes (From 50 to 1000 tasks) on a low number of processors (2, 4, 8, or 16 processors) then HCPT again shows the worst performance while LS-WC-D produces a better schedule among all the algorithms. With the increasing number of processors (32 or 64 processors), all algorithms show almost similar performance. Thus, the computation of Epigenomics applications on a computing system with a large number of processors does not depend on scheduling policies.

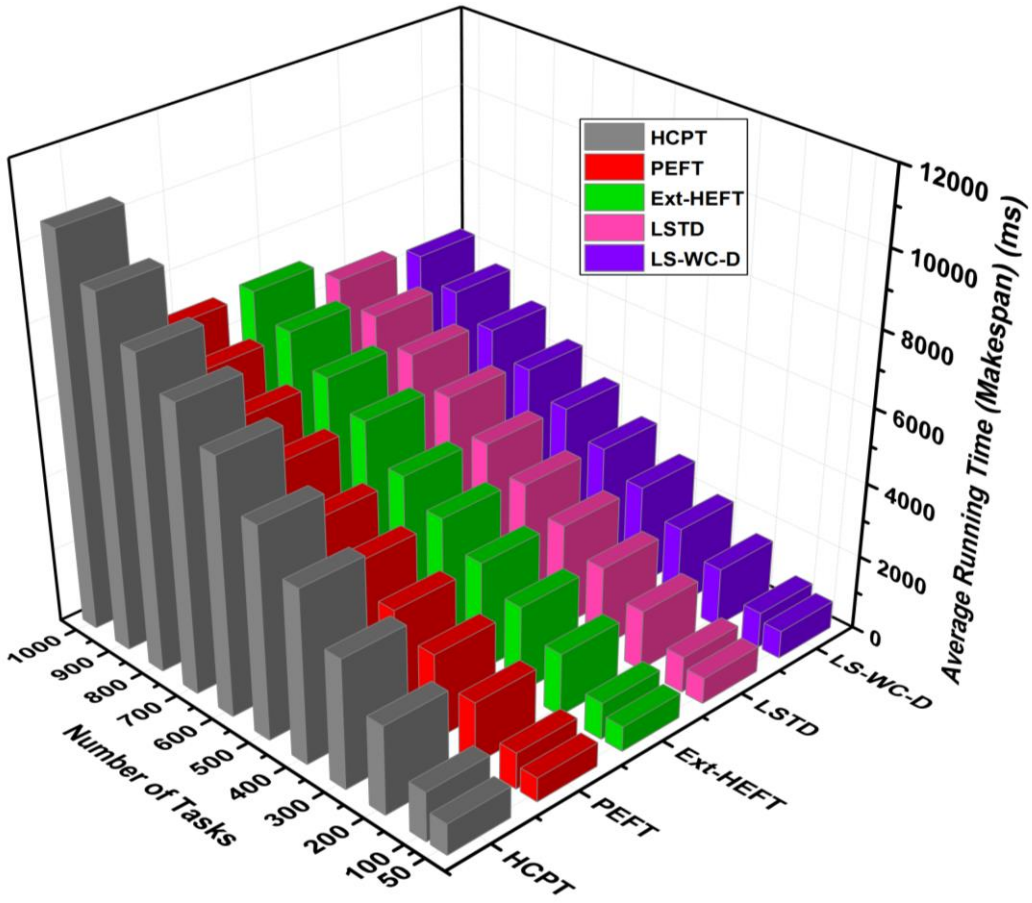


Figure 7. Average running time with respect to increasing size of the Epigenomics workflow applications.

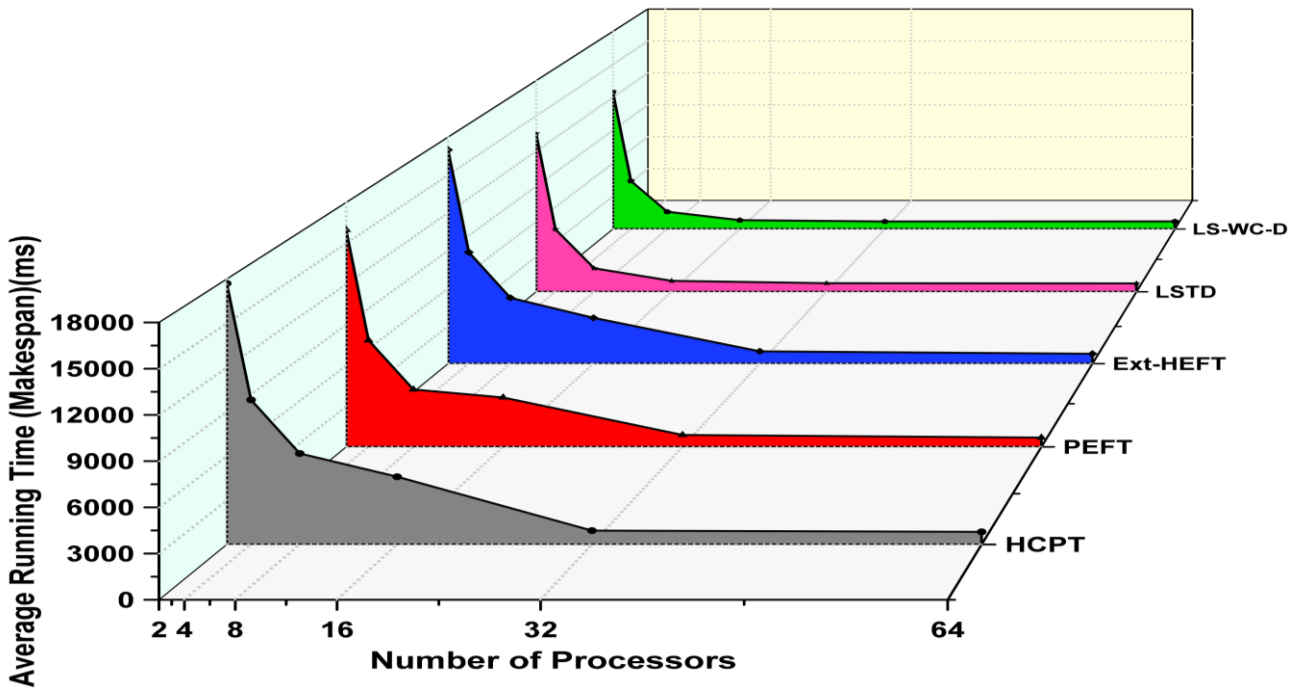


Figure 8. Average running time of algorithms with respect to increasing number of the processors.

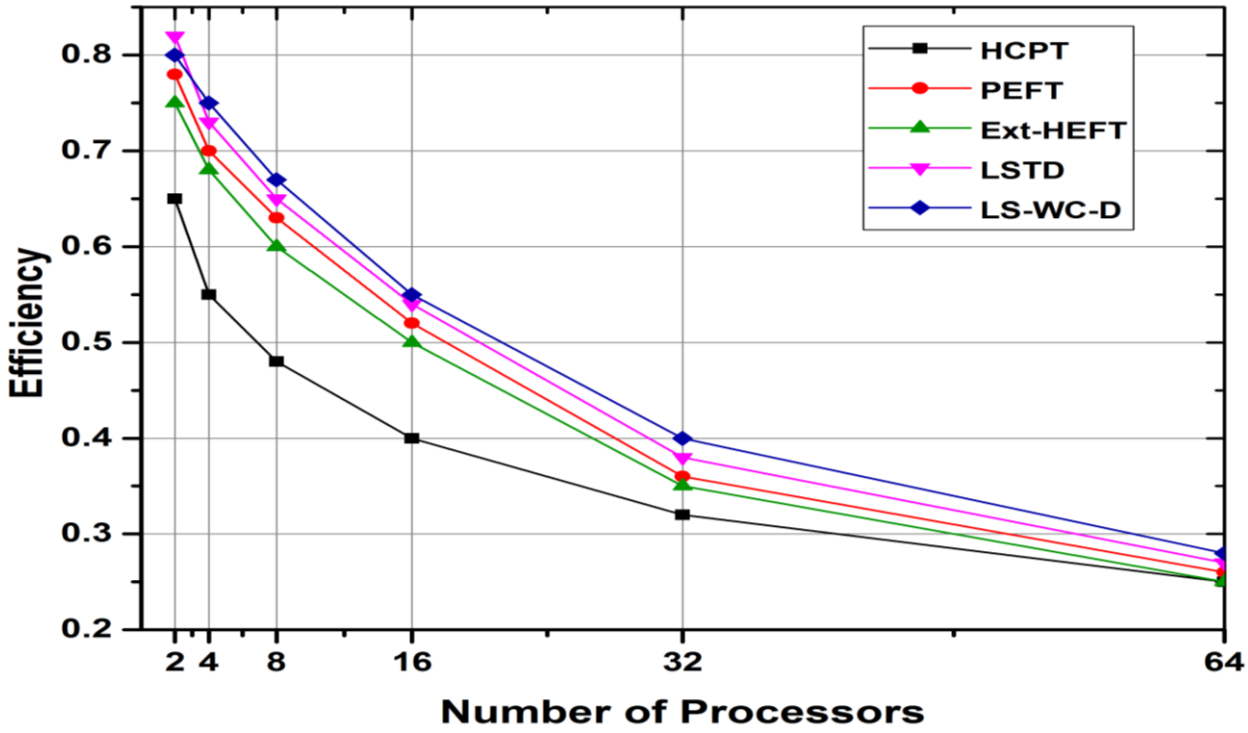


Figure 9. Efficiency of algorithm with respect to increasing number of processors.

The effectiveness of algorithms is illustrated in Figure 9 in relation to the growing number of processors. The LSTD algorithm shows the best performance over other approaches when the number of processors in computing system is 2. But when the number of processors growing from 4 to 64, the LS-WC-D continuously shows better performance. The HCPT shows the worst performance in achieving efficiency. When we schedule Epigenomics application on the system with 64 processors, all algorithms achieve similar efficiency which is the lowest one. Further, when we talk about overall performance, our proposed algorithm LS-WC-D shows 23.30%, 5.51%, 9.39%, and 1.73% higher efficiency than HCPT, HEFT, Ex-HEFT, and LSTD algorithms.

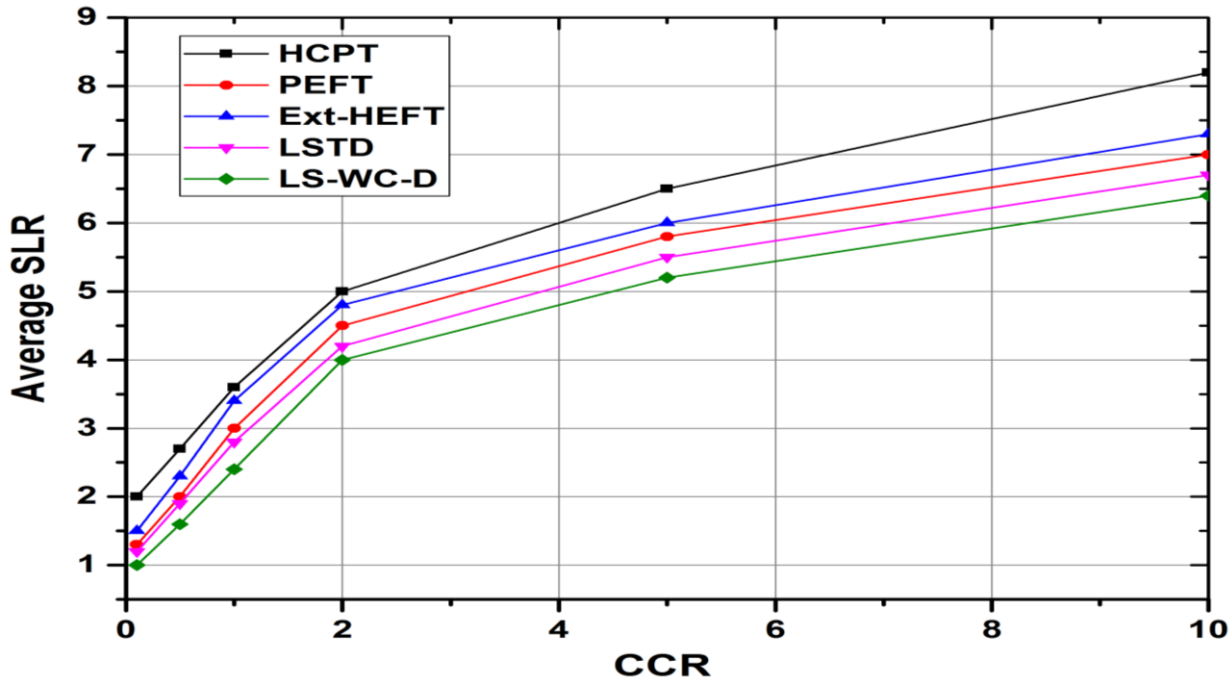


Figure 10. Average SLR of algorithms with respect to increasing value of CCR

Figure 10 represents average SLR of algorithms with the increasing number of CCR values. As we already know the low CCR value represents a system in which processors are connected with a high-speed network connection while a higher value CCR represents processors are connected to a low bandwidth network. When the CCR value is minimal (i.e., 0.1 or 0.5), all algorithms produce average SLR with less difference, whereas for higher value CCR (i.e., 5 or 10), there is a marginal difference between them. Our proposed algorithm LS-WC-D shows better performance in both types of systems and produces 35.92%, 14.56%, 22.81%, and 8.25% better average SLR over HCPT, PEFT, Ex-HEFT and LSTD algorithms, respectively.

Experiment 3: *Percentage of occurrence of better schedule*

In this experimental evaluation, a pairwise table is created to recognize the proportion of better, equal, and worst results generated by LS-WC-D along with its completive algorithms. In Table 2, we can see that our proposed algorithm LS-WC-D produces 59.112%, 77.735%, 72.555%, and 93.23% better results over LSTD, Ex-HEFT, PEFT, and HCPT, respectively.

Table 2. Pairwise makespan comparison of the algorithms

		LS-WC-D	LSTD	Ex-HEFT	PEFT	HCPT
LS-WC-D	Better	----------	59.112%	77.735%	72.855%	93.231%
	Worse		40.140%	20.741%	24.721%	5.768%
	Equal		0.746%	1.524%	2.424%	1.001%
LSTD	Better	40.141%	----------	76.254%	70.317%	90.677%
	Worse	59.113%		21.655%	26.227%	8.123%
	Equal	0.745%		2.091%	3.456%	1.200%
Ex-HEFT	Better	20.741%	21.655%	----------	39.631%	70.477%
	Worse	77.735%	76.254%		57.893%	25.089%
	Equal	1.524%	2.091%		2.476%	4.434%
PEFT	Better	24.721%	26.227%	57.893%	----------	85.897%
	Worse	72.855%	70.317%	39.631%		13.983%
	Equal	2.424%	3.456%	2.476%		0.120%
HCPT	Better	5.767%	8.124%	25.089%	13.983%	----------
	Worse	93.232%	90.676%	70.477%	85.897%	
	Equal	1.002%	1.201%	4.434%	0.120%	

Based on the analysis of experimental results, we observed that the LS-WS-D version of the proposed algorithm produces better results in all experiments than its competitive algorithms. Hence, LS-WS-D is the best choice for the computation of Epigenomics applications in the heterogeneous computing environment.

CONCLUSION

Epigenomics applications are considered complex workflow applications that are used in data processing pipelines for automating the various genome sequencing computations. These applications demand high-performance computing resources for their computation. This article introduces how to schedule Epigenomics workflow applications in a heterogeneous computing environment using a list-based algorithm. To find whether entry-task duplication and clustering techniques improve the performance of the scheduling process, we introduced four versions of the proposed algorithm in which the LS-WC-D (List based scheduling without clustering and with duplication) version shows better performance over other versions. Further, various experiments are conducted and results are analyzed with state-of-the-art algorithms such as HCPT, PEFT, and newly proposed algorithms such as Ex-HEFT, and LSTD. The experimental results show that LS-WS-D is the best option for computing Epigenomics applications in a heterogeneous computing environment.

In the future, we are planning to observe the behavior of the LS-WS-D algorithm in the real cloud environment. To do this, we have to consider numerous cloud features such as VM heterogeneity, VM acquisition latency, lease period, VM failure, etc.

REFERENCES

1. Goldberg AD, Allis CD, Bernstein E. Epigenetics: a landscape takes shape. Cell. 2007 Feb 23;128(4):635-8.
2. Bernstein BE, Meissner A, Lander ES. The mammalian epigenome. Cell. 2007 Feb 23;128(4):669-81.
3. Kouzarides T. Chromatin modifications and their function. Cell. 2007 Feb 23;128(4):693-705.
4. Gao J, Aksoy BA, Dogrusoz U, Dresdner G, Gross B, Sumer SO, et al. Integrative analysis of complex cancer genomics and clinical profiles using the cBioPortal. Science signaling. 2013 Apr 2;6(269):pl1-.

5. Cerami E, Gao J, Dogrusoz U, Gross BE, Sumer SO, Aksoy BA, et al. The cBio cancer genomics portal: an open platform for exploring multidimensional cancer genomics data. Cancer discovery. 2012 May;2(5):401-4.
6. Agarwal R, Kumar B, Jayadev M, Raghav D, Singh A. CoReCG: a comprehensive database of genes associated with colon-rectal cancer. Database. 2016 Jan 1;2016.
7. Chen GG, Gross JA, Lutz PE, Vaillancourt K, Maussion G, Bramoulle A, et al. Medium throughput bisulfite sequencing for accurate detection of 5-methylcytosine and 5-hydroxymethylcytosine. BMC genomics. 2017 Dec;18(1):1-2.
8. Deelman E, Vahi K, Juve G, Rynge M, Callaghan S, Maechling PJ, et al. Pegasus, a workflow management system for science automation. Future Generation Computer Systems. 2015 May 1;46:17-35.
9. Kokash N. An introduction to heuristic algorithms. Department of Informatics and Telecommunications. 2005 Aug:1-8.
10. Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst. 2002 Aug 7;13(3):260-74.
11. Hagras T, Janecek J. A simple scheduling heuristic for heterogeneous computing environments. InParallel and Distributed Computing, International Symposium on 2003 Oct 1; IEEE Computer Society:104.
12. Arabnejad H, Barbosa JG. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans Parallel Distrib Syst. 2013 Mar 7;25(3):682-94.
13. AlEbrahim S, Ahmad I. Task scheduling for heterogeneous computing systems. J. Supercomput. 2017 Jun;73(6):2313-38.
14. Wang J, Han P, Chen J, Du C. Cost-Efficient Scheduling of Workflow Applications with Deadline Constraint on IaaS Clouds. In 2021 Workshop on Algorithm and Big Data 2021 Mar 12. (p. 34-9).
15. Michael LP. Scheduling: theory, algorithms, and systems. Springer; 2018.
16. Belgacem A, Beghdad-Bey K. Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost. Clust. Comput. 2022 Feb;25(1):579-95.
17. Ijaz S, Munir EU, Ahmad SG, Rafique MM, Rana OF. Energy-makespan optimization of workflow scheduling in fog–cloud computing. Computing. 2021 Sep;103(9):2033-59.
18. Singh V, Gupta I, Jana PK. A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. Future Gener Comput Syst. 2018 Feb 1;79:95-110.
19. Abdulhamid SI, Abd Latiff MS, Madni SH, Abdullahi M. Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Computing and Applications. 2018 Jan;29(1):279-93.
20. Adhikari M, Nandy S, Amgoth T. Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. J. Netw Comput. Appl. 2019 Feb 15;128:64-77.
21. Yao F, Pu C, Zhang Z. Task duplication-based scheduling algorithm for budget-constrained workflows in cloud computing. IEEE Access. 2021 Mar 2;9:37262-72.
22. Chen H, Wen J, Pedrycz W, Wu G. Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds. IEEE Trans. Big Data. 2018 Oct 8;6(1):131-44.
23. Ahmad W, Alam B. An efficient list scheduling algorithm with task duplication for scientific big data workflow in heterogeneous computing environments. Concurrency and Computation: Practice and Experience. 2021 Mar 10;33(5):e5987.
24. Arabnejad H, Barbosa J. Fairness resource sharing for dynamic workflow scheduling on heterogeneous systems. In 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications 2012 Jul 10. p. 633-9.
25. Omranian-Khorasani S, Naghibzadeh M. Deadline constrained load balancing level based workflow scheduling for cost optimization. In2017 2nd IEEE Int Conf Comput Intell Appli. 2017 Sep 8:113-8.
26. Wang G, Wang Y, Liu H, Guo H. HSIP: A novel task scheduling algorithm for heterogeneous computing. Scientific Programming. 2016 Mar 17;2016.
27. WorkflowGenerator – Pegasus Workflow Management System https://confluence.pegasus.isi.edu/display/pegasus/Deprecated+Workflow+Generator. Accessed June 17, 2022.