

# Extended Optimization Procedures for Static List based Task Scheduling Algorithms for HeDCS

K. Vijaya Kumar, E. Laxmi Lydia, P. Amaranatha Reddy

**Abstract---** No matter how powerful a single system is efficient at processing, there are still reasons to Control the power of multiple computational units. The Distributed computational system performs scheduling tasks achieved by the processors to minimize the execution time in any application. Despite the problem in determining NP-Complete the execution time in Scheduling is minimized. This paper identifies, a specific different algorithm Sorted Nodes in Leveled DAG Division (SNLDD) based on Task-Scheduling. The fundamental principle of this algorithm is to partition the data as a Directed Acyclic Graph (DAG) two stages and categorize each task of every stage in decreasing order depending upon the estimated size. Outcomes of the proposed algorithm are processed using correlative analysis and productive outcome with respect to HEFT with CPOP is implemented among existing algorithms. With respect to the comparative analysis of the outcomes, the performance of the suggested algorithm with SPOP implements improved execution in the aspect of speedup, effectiveness, complexity, and excellence. Further, a new algorithmic strategy SPOP and CPOP has been developed and executed in the proposed SNLDD in HEFT.

**Keywords---** Task Scheduling, Sorted Nodes in Leveled DAG Division, Superior Performance Optimization Procedure, Heterogeneous Earliest Finish Time, Critical Path on Processor.

## I. INTRODUCTION

Multiprocessor system in the Distributed Computing system is a system with more processors residing apart. Every Individual processor has its private, confidential memory, allowing security. Data Transmission is transferred through message passing between the processors. The processing elements interact with each and every processor in order to accomplish the common outcome. Distributed Computing systems support the following aspects:

- Providing efficiency in communication delay, balancing of load, distributed algorithms for huge messages.
- Providing flexibility in terms of modularity, scalability, portability, and interoperability.
- Computational Consistency that gives correct outputs irrespective of time at which the computation is finished for the identical input conditions.
- Ensuring robustness in certain circumstances of security violations and also in flexibility failures.

**Revised Version Manuscript Received on September 16, 2019.**

**K. Vijaya Kumar**, Associate Professor, Computer Science and Engineering, Vignan's Institute of Engineering For Women, India. (e-mail: elaxmi2002@yahoo.com)

**E. Laxmi Lydia**, Associate Professor & Big Data Consultant, Computer Science and Engineering, Vignan's Institute of Information Technology, India. (e-mail: elaxmi2002@yahoo.com)

**P. Amaranatha Reddy**, Assistant Professor, Department of Computer Science & Engineering, VFSTR University, Andhra Pradesh, India.



Figure 1: Distributed Computing Systems

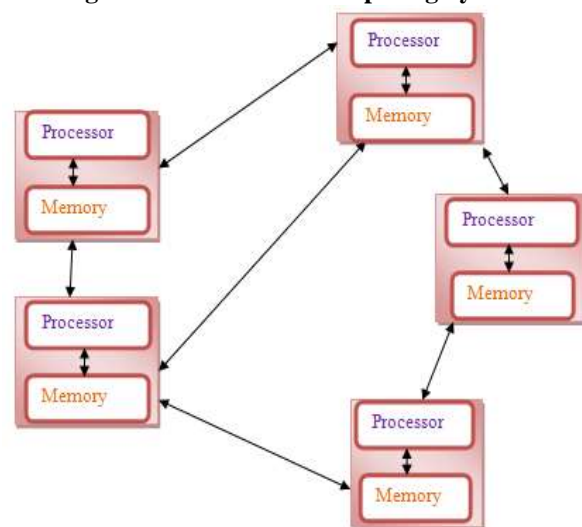
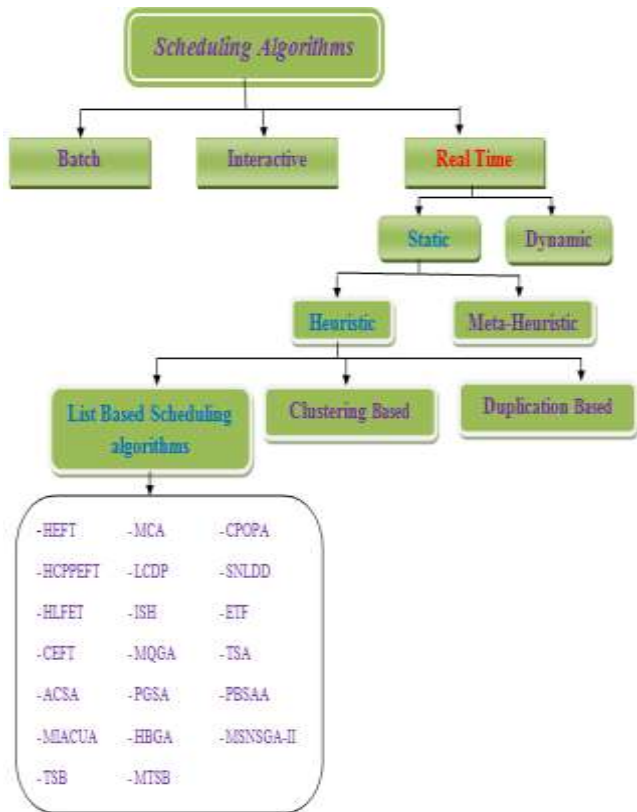


Figure 2: Sharing of results with different Processors

### Task Scheduling

Task scheduling deals with the process of organizing tasks among processors and minimizes its length in scheduling, i.e., the time is taken to execute every task corresponds to the starting time of the initial task. Queries generated on the outcome is the selection of tasks to processors. Real-time task scheduling originally deals with regulating of order in which the different tasks are to be reserved up for implementation by the operating system. Every operating system depends on more than one task, schedules to prepare the schedule of application of different tasks it process to execute. Each task scheduler is identified by the scheduling algorithm it operated. A massive number of designed algorithms for scheduling, real-time tasks have been refined. Real-time scheduling

# EXTENDED OPTIMIZATION PROCEDURES FOR STATIC LIST BASED TASK SCHEDULING ALGORITHMS FOR HEDCS



**Figure 3: Scheduling algorithms classification**

## II. LITERATURE SURVEY

Distributed computing systems are the number of an individual collectivity of processors interconnected through a tremendous system of connections that maintains the implementation of parallel operations [1]. The effectiveness of implementing various parallel operations on the systems desperately relies on the adopted approach to perform the tasks of the various parallel operations on existing processors [2]. Every System in DCs, the execution of parallel computations in inter-processor communications is liable [3]. This burden exists whenever tasks are positioned to various processors in the transfer of data. Accordingly, in Heterogeneous distributed Systems generating high capacity task performances in parallel implementation are more critical [4]. Such Conditions, an extensional Arrangement among achieved speedup by automatic parallelization, the communicational inter mechanisms over systems, HeDCSs having scheduling algorithms by considering the different processors executing at different times of the same task. Some unreliable Scheduling arrangement in HeDCSs may restrict the effectiveness of the system when they perform in the very slow processors [5]. Mostly, Distributed Computing Systems in Scheduling algorithms have Static scheduling and Dynamic scheduling as two different existing task scheduling.

The primary Static scheduling algorithms, perform operations on the entire existing data for scheduling, for the framework of parallel operations, processing operations with respect to execution time, the cost among the tasks of communication is acknowledged prior [5]. To evaluate specific information various approaches are used [4, 6]. In the parallel implementations, Static task scheduling

concurrently has an activity of running processes at compile time [2,3,7].

Lately, a modernized algorithm known as the Longest Dynamic Critical Path was initiated [6]. Based on the algorithm, an improved attribute has identified the priorities accurately in HeDCSs of each task. The outcome of the LDCP is measured for comparison of two algorithms HEFT [11] and the DLS [5] algorithms related to performance.

This paper provides, a modern beneficial algorithm known as Sorted Nodes in Leveled DAG Division (SNLDD) to support HeDC systems with a defined total estimated number of processors for static implementation of task scheduling. The goal or achievement of using this algorithm is to obtain the large capacity scheduling of tasks with necessary to produce a tremendous performance in the Heterogeneous Distributing Computing Systems.

SNLDD is calculated with a parallel correlative study among SNLDD and the HEFT algorithm for execution. Based on these results, the SNLDD algorithmic approach excels the HEFT algorithm in conditions of schedule distance, time, performance, and characteristics of system behavior.

The SNLDD design approach has been changed by recommending a new process known as Superior Performance Optimization Procedure (SPOP) for the reduction of the sleek time by applying the idle time concurrently referring tasks to the processors by producing high-quality scheduling of tasks, and to reduce scheduling length. In addition, both updated algorithms are correlated and altered SNLDD algorithm confirms in excellent performance than transformed CPOP.

### *Problem Definition*

Parallel operations are expressed by DAG for HeDCSs, in static task scheduling.

*Direct Acyclic Graph* is determined by considering tuple (T,E), T defines the collection of 'n' tasks and E defines the collection of 'e' edges. Every task,  $t_i$  corresponds to T in a parallel function, and every edge  $(t_i, t_j)$  corresponds to an order from the beginning of the constraint and a transfer of information among task  $t_i$  and  $t_j$ . With the condition edge  $E(t_i, t_j)$ , the implementation of  $t_j$  T is implemented after  $t_i$  T completes its implementation. The task that is started first  $t_i$  as an edge  $(t_i, t_j)$  is represented as the parent to the sink task  $t_j$ , at the same time  $t_j$  is represented as a child of the  $t_i$ . Any individual task performing operations with no parents are named entry task, and any individual task performing operations with no children are named exit task. Most quantities of data that is transmitted from one task to another task (i.e.,  $t_i$  to  $t_j$ ) are related to every edge  $(t_i, t_j)$  [5, 11].

*Heterogeneous Distributed Computing Systems* is determined by a group of m processors with P that has distinct result performances. Computation cost matrix (W) (i.e.,  $n \times m$ ) keeps the implementation costs of all the 'n' tasks in m processors. Every component  $w_{i,j}$  W produces the time of task processor with predicted performance.



Now, every individual processor for the performance is simulated altogether linked. Information exchange among processors can take place through individual links; by this simultaneous execution of evaluation tasks and interactions among processors are approved [3, 12, 13]. Computational costs of the tasks are acknowledged without changing. Especially, when task ( $t_i$ ) computation costs at the respective processor ( $p_j$ ) and processor ( $p_k$ ). The communication cost between any two processors ( $p_k$ ) and ( $p_i$ ) rely on the network initialization at processors  $p_k$  and  $p_j$  further to the transfer time on the network. Processors ( $p_k$ ) and ( $p_i$ ) communication cost rely on processors ( $p_k$ ) and ( $p_i$ ) at network initialization and the time needed to communicate. Sender initially starts at some point in time and time at the receiver side is ignored when compared to the residing network time [14]. The data transfer rate inside the network among the two processors predicted as constant and permanent [3, 5]. Thus, edges in transmission correspond to the data transferred from ( $t_i$ ) to ( $t_j$ ), separated by the network at a transfer rate and the network is united without failure of generality while achieving data transfer rate mostly in the inter-processor network [14, 15]. Accordingly, every single edge is equivalent to  $d_{i,j}$  with respect to tasks on various processors at communication cost mostly scheduled. Considering, the connections identical processors among any two tasks are hired at zero. The Processor implements its individual tasks exclusively parents from the entire data applicable to exact processors; in such a case the task is ready to run and marked as running. The total run time of the parallel operations is allocated to achieve minimum runtime [3,9,11].

### III. IMPLEMENTED METHODOLOGY

#### The Sorted Nodes in Leveled DAG Division Algorithm (SNLDD)

One of the advanced task scheduling approaches known as SNLDD has been introduced and is related to partitioning graphs into different levels altogether concentrating on the need of priorities among tasks in the DAG. These tasks at every level are categorized into a list positioned based on computation size. These tasks are allocated to the first initiated processors arranged in precedence to the list. The computational size of every individual task is evaluated as follows:

$$S_j(n_i) = (w_j(n_i))_{p_f} + \{C_j[(n_i)_j, \sum_{k=1}^t (n_k)_{j-1}] + C_j[(n_i)_j, \sum_{x=1}^q (n_x)_{j+1}]\} \quad (1)$$

Where  $S_j(n_i)$  describes the computation size pointing to a specific task ( $n_i$ ) in the  $j^{\text{th}}$  level

i.e,  $1 \leq j \leq R$ , here  $R$  defines the number of levels in the graph and

$1 \leq i \leq T$ ,  $T$  defines the number of tasks in the schedule.

The primary part of the mentioned equation evaluates the performance task time, considering at processor  $p$  from  $j^{\text{th}}$  level with the high speed of the system.

The second part of the above-mentioned equation describes the sum of transmission among the task  $n_i$  in  $j^{\text{th}}$  level. Parent nodes in that  $j-1^{\text{th}}$  level, and Child nodes in  $j+1^{\text{th}}$  level of communications.

Considering LDCP algorithm, tasks estimating the longest path based on the DAG. Every task performs lots of

arithmetic evaluations because of repeated computations causing overheads in communications [6]. For this reason, the introduced SNLDD algorithm considers DAG algorithm as a base and use the graph by sub dividing the levels of DAG and allocating each level task to processors. Thus, overheads are evaluated in the computations. In SNLDD algorithm, tasks in each level are partitioned by obtaining the sizes of the computation and sorted checking the priority, which is easy to classify.

That simplifies SNLDD is more efficient when compared to the LDCP algorithm as a result, the time for selecting each task which was returned is allowed will be evaluated in every step.

Most often, after partitioning DAG graphs into subdivisions, tasks are allotted to each level, introduced SNLDD algorithm grows with higher efficiency compared to the LDCP algorithm. Following are some demonstrations for the algorithm:

- It aims to restore the entire directions, operating time, tasks and communications at every appointing step that is not required to introduce SNLDD algorithm, overheads at execution time are removed in SNLDD algorithm.
- Processors based on their computational size comforting not only adequate scheduling of tasks further accepts to develop the entire system of allocation of tasks basing on many characteristics like its cost for communications, reliance, and execution time at authorized tasks.
- Partitioning every level tasks are based on computing size by managing scheduled tasks with substantial computation size and then tries to reduce the need among tasks.
- Processors performing sleek time are reduced as long as DAG subdivision of levels and allocating to the processors.
- The introduced algorithm is designed to have more effective when compared to other algorithms like LDCP, HEFT, and LCD.
- Several different designs of extremely high currently used algorithms like sorting list [6], tree algorithms [9], and clustering algorithms [2] are confirmed. Therefore, it is confirmed that SNLDD algorithm is a set of selected algorithms.

To overcome the repeated computations on each step of LDCP, SNLDD algorithm, the computation size of every task is executed only once in DAG, and restoring them by estimating on every step.

This takes time in the evaluation of SNLDD algorithms for better calculations.

$\Theta(m \times n^2)$  is the time complexity, where  $m$  describes a number of processors used in the scheduling algorithm, and  $n$  describes the number of tasks considered in the scheduling algorithm.

The following figures 4 and 5 are the sample assumed DAG with computational costs.





## EXTENDED OPTIMIZATION PROCEDURES FOR STATIC LIST BASED TASK SCHEDULING ALGORITHMS FOR HEDCS

Task	P1	P2	P3
1	12	14	7
2	11	17	16
3	9	11	17
4	11	6	15
5	10	11	8
6	11	14	7
7	5	13	9
8	3	9	12
9	16	10	18
10	19	5	14

Figure 4: Example of Direct Acyclic Graph

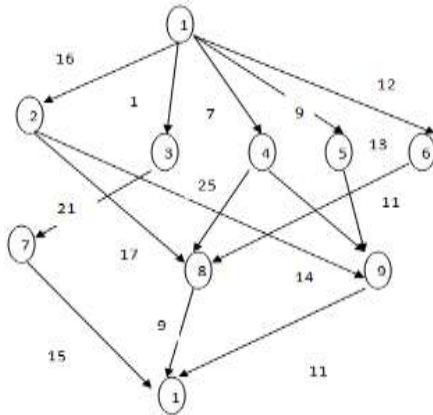


Figure 5: DAG with computation matrix

### Implementing the Superior Performance Optimization Procedure (SPOP)

SPOP is known for a modernized optimization procedure with high-performance, extending itself to specify algorithms. It concentrates on the reliability of the tasks assigned to the processors by reducing its time considering scheduling policy[6]

- The scheduling policy assigned to the tasks by the processors at insertion based assumes every feasible idle time on the processor is achieved data time slot that is similar and higher length to the computation/implementation time. Precedence constraints among tasks are very required in performing.
- The idle time is computed by exploring the ready state time by defining the length and inserted back to the final task that is scheduled on the processors.

### HEFT (Heterogeneous Earliest Finish Time)

HEFT is one of the straightforward and best approaches in the static task scheduling both environments like heterogeneous and homogeneous even when few processors are considered. HEFT deals with crossing rather evaluating two phases: one is Prioritization phase and another is processor selection stage.

In the first stage of Prioritization: HEFT estimates the importance of the tasks by using the upward ranking ( $rank_u$ ). An application operation is moving over in upward path and achieves entire list nodes with its rank with the service provided by the mean communication and performance cost. An obtained list is organized in systematized descending order of the  $rank_u$ . Every task with upward rank is defined as:

$$Rank(n_i) = W_i + \max_{n_j \in succ(n_i)} (C_{ij} + rank_u(n_j))$$

$W_i$  describes the mean computation cost,  $succ(n_i)$  is described as the immediate child of node  $n_i$ ,  $c_{i,j}$  is described as the mean communication cost of node  $(i,j)$ . If any two nodes have when two nodes obtain the similar rank value, it will prefer indiscriminately. The attained graph is moved from the end node to start node. The maximum rank value is similar to the end node:

$$rank_u(n_{exit}) = W_{exit}$$

### Critical Path On Processor (CPOP)

CPOP manipulates into two methods, namely upward ranking and downward ranking. CPOP calculates the rank by the value of every node by combining paired the ranking approaches rank + ranked and assign them to a queue. CPOP also has two levels of calculation: task prioritization level and task allocation level.

The first phase of task prioritization, every task is prioritized based on the rank value ( $rank_u + rank_d$ ) by use of transmission and computational costs that is obtained from DAG. Later, they are placed in a queue (in low order). CPOP performs Critical path to obtain the longest path from the start node to last node.

The second phase of task allocation, all the tasks are picked concentrating on maximum rank and chosen the minimum execution time of the task giving appropriate processor.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

Result analysis is based on the attributes, namely: schedule length of the tasks, speedup of the processors and efficiency of the entire algorithm.

Comparison metrics of the algorithms SNLDD after modification through the performance of Superior performance optimization Procedure and with the Heterogeneous Earliest Finish Time with respect to Critical Path On Processor.

Schedule length finds overall execution time of a considered Direct Acyclic Graph (DAG), which is also known as makespan. Speedup defines the ratio of processed schedule length with the total number of processors. Efficiency gives the estimated by dividing speed with the total processors each time when they run.

Figures 7 and 6 are the schedule length with 10 tasks and the speedup with processors.

Figure 6 describes the speed of the processors in the distributed computing systems when 20, 40, 60, 80, 100 are considered. It shows that when the number of processors is where the speed is high. The figure thus concludes that the algorithm SNLDD with optimized procedure has given maximum result than the algorithm HEFT.

Figure 7 describes the schedule length of the considered DAG gives overall execution time. Here the comparison is done among the algorithms SNLDD without Optimization Procedure, i.e., only SNLDD, with SPOP, HEFT, and CPOP. Among all the algorithms the execution time of the modified SNLDD gives the fastest evaluation time.



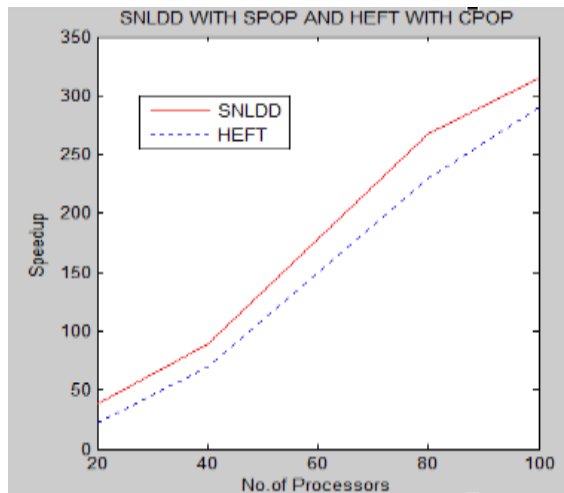


Figure 6: Speedup of the processors

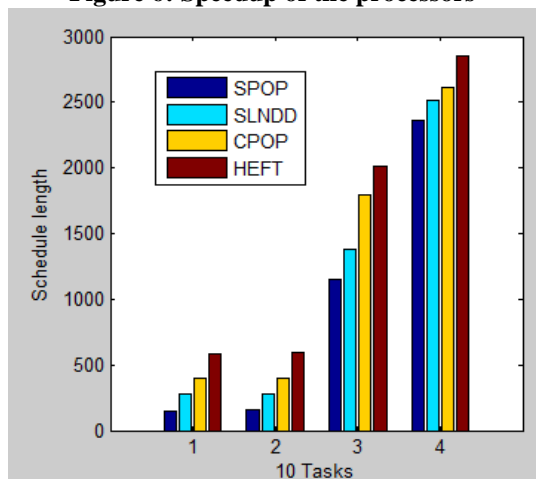


Figure 7: Schedule length with obtained by SPOP with SNLDD, SNLDD, CPOP, HEFT

## V. CONCLUSION

Scheduling Algorithms, mostly List based static algorithms are considered for HDCS. Based on the algorithms, SNLDD, HEFT, CPOP, and implementation of SNLDD with Superior Performance Optimization Procedure are studied. In this paper, the outcome performance of the developed SNLDD algorithm is correlated with current existing algorithms mainly for HeDCSs. The comparative study between the proposed SNLDD algorithm with modified optimization procedure SPOP and HEFT with CPOP are evaluated based on the schedule length, speedup, efficiency of running programs and quality parameters with respect to memory in parallel computer systems has achieved a high speed up and fast execution time by SNLDD.

## REFERENCES

- Attiya Gamal, Hamam Yskandar. Task Allocation For Maximizing Reliability Of Distributed systems: a simulated annealing approach. *J Parallel Distributed Comput* 2006;66:1259-66.
- Bansal S, Kumar P, Singh K. Dealing with heterogeneity through limited duplication for scheduling precedence-constrained task graphs. *J Parallel DistribComput* 2005;65(4):479-91.
- Bajaj R, Agrawal DP. improving scheduling of tasks in a heterogeneous environment. *IEEE Trans Parallel DistribSyst* 2004; 15(2):107-16.
- Shivle S, Castain R, Siegel HJ, Maciejewski AA, Banka T, Chindam K, et al. Static mapping of subtasks in a heterogeneous ad hoc grid environment. In: *Proc of parallel and distributed processing symposium*; April 2004.
- Hwang K. *Advanced computer architecture: Parallelism, scalability, programmability*. New York: McGraw-Hill, Inc; 1993.
- Mezmaiz M, Melab N, Talbi E-G. An efficient load balancing strategy for grid-based branch and bound algorithm. *Parallel comput* 2007;3:302-13.
- Boyer WF, Hura GS. Non-evolutionary algorithm for scheduling dependent tasks in distributed heterogeneous computing environments. *J Parallel DistribComput* 2005;65(9):10350-1046.
- Zelkowitz Marvin. "Advances in computers" Elsevier journal. *Computer Science*, vol. 78; 2010.p.368.
- Kim J, Rho J, Lee J-O, Ko M-C. CPOC: effective static task scheduling for grid computing. In: *Proceedings of the 2005 international conference on high-performance computing and communications*, Italy; 2005.p. 477-86.
- Kwok Yu-Kwong. *High-performance algorithms for compile-time scheduling of parallel processors*. The Hong Kong University of Science and Technology in Partial Fulfillment of the requirements for the degree of Doctor of philosophy in computer science Hong Kong; May 1997.
- Topcuoglu H, Hariri S, Wu MY. Performance-effective and low complexity task scheduling for heterogeneous computing. *IEEE Trans Parallel DistribSyst* 2005;13(3):260-74.
- Daoud Mohammad I, Kharna Nawwaf. A high-performance algorithm for static task scheduling in heterogeneous distributed computing systems. *IEEE Trans Parallel DistribSyst* 2007;28:39-49.
- Tian Yuan, Boangoat Jarupan, Ekici Eylem, Ozguner Fusun. Real-time task mapping and scheduling for collaborative in-network processing in DVS-enabled wireless sensor networks. *IEEE Trans Parallel Distributed Systems*, Rhodes, Greece (IPDPD 2006).
- Doaa M. Abdelkader, Fatma Omara, Dynamic task Scheduling algorithm with load balancing for the heterogeneous computing system. *Egyptian Informatics Journal* (2012) 13, 135-145.
- Aida A. Nasr, Nirmeen A. El-Bahnasawy, Ayman El-Sayed, Optimization procedure for algorithms of task scheduling in high performance heterogeneous distributed imputing system. *International Journal of Computer Applications*, Volume 107-No 4, December 2014.
- Nirmeen A. Bahnasawy, Magdy A. Koutb, Mervat Mosa and Fatma Omara, A new algorithm for static task scheduling for heterogeneous distributed computing systems. *African Journal of Mathematics and Computer Science research* Vol.4(6), pp.221-234, June 2011.
- Radulescu A, van Gemund AJC. Fast and effective task scheduling in heterogeneous system. In: *9th Heterogeneous computing, Workshop*; 2000. p. 299-238.
- Zhao Rizos Henan, Sakellariou Rizos. An investigation into rank function of the heterogeneous earliest finish time (HEFT) algorithm. University of Manchester, UK: Department of Computer Science; 2003.
- Cirou Bertrand, Jeannot Emmanuel. Triplet: a clustering scheduling algorithm for heterogeneous systems. In: *IEEE symposium on reliable distributed systems*; 2001. p. 231-6.

## EXTENDED OPTIMIZATION PROCEDURES FOR STATIC LIST BASED TASK SCHEDULING ALGORITHMS FOR HEDCS

20. OmaraFatma A, Arafa Mona M. Genetic algorithms for task scheduling problem. *Parallel DistribComput J* 2010; 70(1):13-22.
21. Rudolph Larry, Slivkin Miriam, Upfal Eli. A simple load balancing scheme for task allocation in parallel machines. In: *Proceedings of the third annual ACM symposium on parallel algorithms and architectures*; 1991. p. 237-45.
22. Nour Mohamed, OmaraFatma A. Performance evaluation of parallel task scheduling algorithms for heterogeneous processors. *Egypt Inform J* 2005;6(1): 98-122.