Multi-Level Queue for Task Scheduling in Heterogeneous Distributed Computing System

Tarun Biswas
Department of CSE,
NIT Sikkim, India
tarun.nitskm@gmail.com

Pratyay Kuila
Department of CSE,
NIT Sikkim, India
pratyay_kuila@yahoo.co.in

Anjan Kumar Ray
Department of EEE,
NIT Sikkim, India
akray.nits@gmail.com

***Abstract*—In last several years, significant amount of attention has been devoted to task scheduling process in heterogeneous distributed computing system (*HDCS*). The task scheduling is an important problem, where different tasks are scheduled to target processor in such a way that the overall tasks computation is completed within a shortest possible time. In this paper, we have presented a multi-level queue (MLQ) task scheduling algorithm to minimize the makespan for parallelizing the subtasks without violating the precedence relationships. Here, our main objective is to exploit the advantages of heuristic-based task scheduling algorithms in terms of makespan, time complexity, resource utilization, system throughput and dynamic nature. Our contribution is analyzed and evaluated through experimental results**

***Keywords*— Distributed Systems, Task scheduling, Directed Acyclic Graph (DAG), Earliest Time First (ETF), Makespan.**

I. Introduction

The recent computing world inherits services from a heterogeneous distributed computing platform, where the computational units may be situated in different geographical regions. All this computational units are connected through a communication network [1]. There are several kinds of distributed systems, such as grids [2], peer-to-peer (P2P) systems [3], ad hoc networks [4], cloud computing systems [5], pervasive computing systems [6], and online social network systems [7]. Generally, an application is divided into a number of smaller sub-tasks. These divided smaller tasks are scheduled to be executed on different computing nodes. The tasks can be dependent or independent. As per the task requirement tasks are matched with target machines in two different ways, static and dynamic. The static technique is scheduling process in which a set of tasks are mapped with prior knowledge but in dynamic task scheduling process there is no prior are associated with them. The dynamic task scheduling solely based on the arrival time of the tasks and non-preemptive in nature. The problem of task scheduling is how well the task assignment are done to the heterogeneous computing processors, so that all task can be executed within a shortest possible time period or the schedule length can be minimized. A task scheduling algorithms [8] can be explained as a process in a distributed system, which helps to make effective use of the resource capabilities by proper task schedule among different live machines. Therefore, efficient task scheduling has a direct impact on the performance of HDC system. The performance in respect overall completion time, response time, resource utilization and load balancing of a distributed system is solely dependent on how the tasks are scheduled on their respective machines [9]–[12].

There are a numbers of resources in the system, such as data, replicas of popular web objects, processors, computational capacity, and disk storage. Resources are distributed among nodes and are often agree to share their local resources with each other so that the scheduled tasks can be completed within a shortest period of time. In general, a large application is divided into a set of smaller subtasks. The subtasks are prior to parallel processing based on their precedence relationship [13]. According to their precedence the smaller subtasks are scheduled to execute on multiple units concurrently. In the paper, we addressed the task scheduling problem and proposed a task scheduling algorithm using a multi-level queue (MLQ) on heterogeneous computing systems. We have also shown that our algorithm has improvements in terms of makespan, time complexity and system throughput with some existing algorithms [12], [14].

The remainder of this paper is organized as follows. Section II describes the literature survey of the different task scheduling algorithms. In Section III, we present the system model, application model and problem statement of our proposed work. The new task scheduling Algorithm is presented in Section IV. In Section V, we have shown and analyze the performance evaluation of

978-1-5090-4559-4/17/$31.00©2017IEEE

the proposed work. We conclude our work with future possibilities in section VI.

II. Related Works

The Min-Min algorithm begins with the set of meta-task (MT) tasks [14]. These algorithms are used to schedule tasks by considering the execution time of the tasks on the corresponding machines. It computes minimum completion time for all the tasks in MT on all the machines. Next, the task which needs minimum time to complete its execution is selected and assigned to the respective machines. The whole process is continued until the entire processes are scheduled. The maxmin heuristic is alike to min-min algorithm. Here also each jobs minimum completion time all the machines is computed and then the task with maximum minimum completion time is selected and assigned corresponding machines. The Min-min and Max-min algorithms are considered as a hypothetical scheduling of tasks to machines. Both the algorithms have time complexities of $O(T^2XM)$, where, T is the number of tasks in the system and M is the number of machines. The list-based algorithms, such as heterogeneous earliest finish time (HEFT) and critical path on a processor (CPOP) [12], where scheduling of subtasks is done by assigning priority to each subtasks. The algorithm is solely dependent on two main phases- partition of tasks and selection of processors. At first stage a rank value is calculated based on the mean computation and mean communication costs. A descending order ranked based task list is generated. In the next stage the processor is selected on which a task schedule to execute in an earliest finish time. The performance of these algorithms is heavily dependent on the effectiveness of the heuristics. The time complexity of HEFT algorithm is equal to $O(T^2XM)$ where, T is the number of tasks in the system and M is the number of machines [9].

Heterogeneous Critical Parents with Fast Duplicator (HCPFD) works on a heterogeneous fully connected computing system consist of a limited number of machines. Another task duplication based task scheduling algorithm is introduced, known as Critical Parents with Fast Duplicator (CPFD) where processor allocation is done by the priorities of the tasks. The complexity in duplication based approach is $O(T^2XM)$ where, T is the number of tasks in the system and M is the number of machines [11], [12].

III. Models

A. Syestem Model

We have assumed a distributed computing system consists of a set of heterogeneous processors that are fully interconnected with a network. Each task of any application is executed only on a single processor. The tasks can be independent or dependent. We are concerned about the earlier completion time (*ECT*) of dependent tasks. The tasks are scheduled on different processors in a static environment where task's dependencies and other behaviors are known a priori. It is also assumed that the communication cost is same for all the data transfers between two processors. The computing system can be categorized in two way, i.e., processor-based heterogeneity model (*PHM*) and task-based heterogeneity model (*THM*). In a *PHM* computing system, a processor executes the tasks at the same speed, regardless of the type of the tasks. On the contrary, in a *THM* computing system, how fast a processor executes a task depends on how well the heterogeneous processor architecture matches the task requirements and features [12]. The target system topology is shown in the Fig.1

B. Application Model

In our work an application is represented in form of a Directed Acyclic Graph (*DAG*) $G = (T, E)$, where T is the set tasks to be executed on different processors and E is the of edges representing the precedence constraints between tasks. Let a dependent pair of subtasks t_i and t_j, where an edge $e(t_i, t_j)$ € E implies that the execution of t_j can only start after completion of task t_i, i.e. the execution of subtask t_j is solely dependent on the output of task t_i. The DAGs of our example application are shown in Fig. 2.

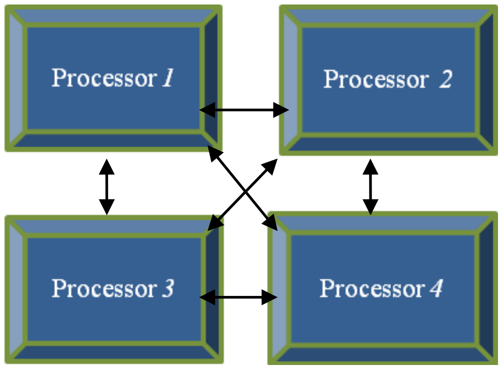


Fig. 1: A fully connected four heterogeneous processors system

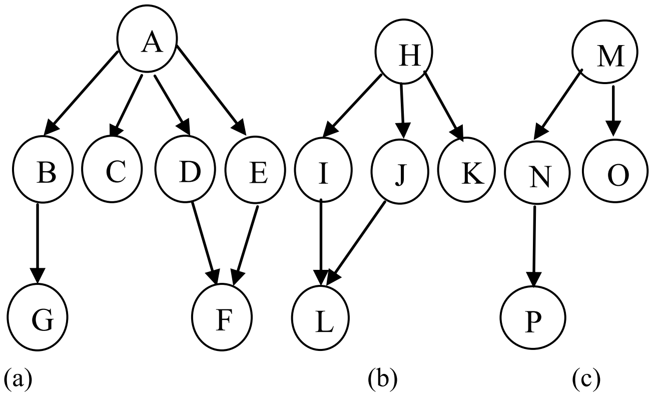


Fig. 1: Different applications DAG

C. General Notations

The following notations are used to describe the proposed work.

- $P = (p_1, p_2, p_3 \ldots\ldots p_m)$ denotes the set of m number of processors.
- $T = (t_1, t_2, t_3\ldots\ldots t_n)$ denotes the set of n number of tasks.
- $E(t_i, p_j)$ denotes, the execution time of task t_i on processor p_j.
- $C(t_i)$ denotes the completion time of task t_i.
- $t_i \rightarrow t_j$ denotes the t_i is said to be an immediate predecessor of t_j, where e_{ij} € E.
- $Pd(t_j)$ denotes the set of immediate predecessors of t_j.
- $S(t_i)$ denotes the set of immediate successors of task t_i.
- t_{entry} denotes the starting subtask of the application without any predecessor, i.e., $Pd(t_{entry}) = ø$.
- t_{exit} denotes the end task which does not have any successors, i.e., $S(t_{exit}) = ø$.
- The *t-level* of a task t_i is the length of a longest path from the entry task t_{entry} to t_i. Therefore, *t-level* of t_{entry} is zero.

D. Problem definition

The task scheduling problem can be defined as, given a set of tasks need to be scheduled on the target processors so that all the tasks can be completed within a minimum time period, i.e., minimization of the makespan. The makespan of the schedule

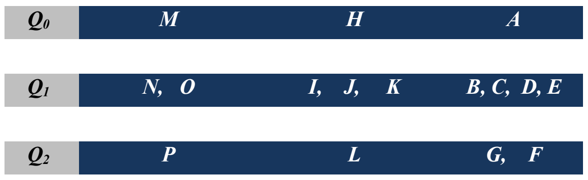


Fig. 3: MLQ formation

is considered as the completion time of $C(t_{exit})$. While assigning a task to a suitable processor, we have to taken care the dependency, i.e., we can not assign a task until unless all of its parent tasks finishes.

IV. Proposed Work

Our proposed algorithm schedules a set of tasks on a given number of heterogeneous processors. The proposed work consists of three fundamental operations, Multi-Level Priority Queue (MLQ) formation, Selection of processor and Queue shifting operation. The operations are as follows:

Formation of MLQ: At first, a multi-level queue of the tasks is formed based on the *t-level* of the tasks. The queue of level i, i.e., Q_i holds all the tasks with *t-level* i. Therefore, initially all the t_{entry} tasks are in Q_0. Similarly Q_1 holds all the tasks with *t-level 1* and so on as shown in Fig. 3.

Selection of Processor: Now, the task (say t_i) with minimum execution time is selected from the Q_0.and assigned to a processor in the set of processors that minimizes its finish execution time i.e., $EFT(t_i)$.

Queue Shifting Operation: After the assignment of t_i, it is removed from the Q_0 and all the child tasks of t_i those are currently in Q_1 are shifted to Q_0. Subsequently, the child tasks of the shifted tasks are moved from Q_2 to Q_1. The process is continued for all the queues. Now, the whole process is continued until the Q_0 becomes empty as shown in Fig. 4.

A. An illustration

Let us consider three different application DAGs as shown in Fig. 2 and the corresponding execution time of each tasks is given in Table I. As per the algorithm, initially, Q_0 has all the parent tasks, i.e., A, H and M. The minimum execution time of the tasks A, H and M is 3, 2 and 2 respectively. Now the task with minimum execution time is selected from Q_0 to be assigned. Let us consider M is selected (between H and M, as both has the minimum execution time). M has to be assigned to the processor for $E(M, p_1)$. It can be seen from Table II that in processor p_1, M is finished earlier. Therefore, M is assigned to p_1 which is reflected in line *1*, Table II. After assigning M,all the child tasks of M from Q_1, i.e., N and O are moved to Q_0 (refer to line *2*, Table II) and consequently the child of N and O are also moved accordingly. Now, the next minimum execution time task from Q_0, i.e., H is a scheduled to p_4 for$EST(H)$ as it is reflected in line *2*, Table II. The child and sub- child tasks of H are moved at their respective queue. The whole scheduling process is continued until Q_0 is empty. The final scheduling can be seen in Table II and the algorithm can be seen in Fig. 5.

TABLE I: GIVEN EXECUTION TIME AND RANK CALCULATION OF HEFT AS IN [12].

Tasks	Execution Time $E(t_i, p_j)$				$Avg_{E(ti)}$	$Rank_{ti}$
	P_1	P_2	P_3	P_4		
A	9	3	5	7	6	16.25
B	4	6	7	2	4.75	10.25
C	9	8	10	4	7.75	7.75
D	2	6	5	4	4.25	9.25
E	6	5	2	3	4	9
F	5	2	7	6	5	5
G	9	4	6	3	5.5	5.5
H	5	4	7	2	4.5	19.5
I	2	5	6	3	4	13.25
J	8	4	9	2	5.75	15
K	6	3	7	4	5	5
L	10	8	7	11	9.25	9.25
M	2	3	4	8	4.25	11
N	8	10	2	7	6.75	13.25
O	3	4	5	6	4.5	4.5
P	6	7	9	4	6.5	6.5

TABLE II: Schedule of tasks in each iteration of our proposed algorithm

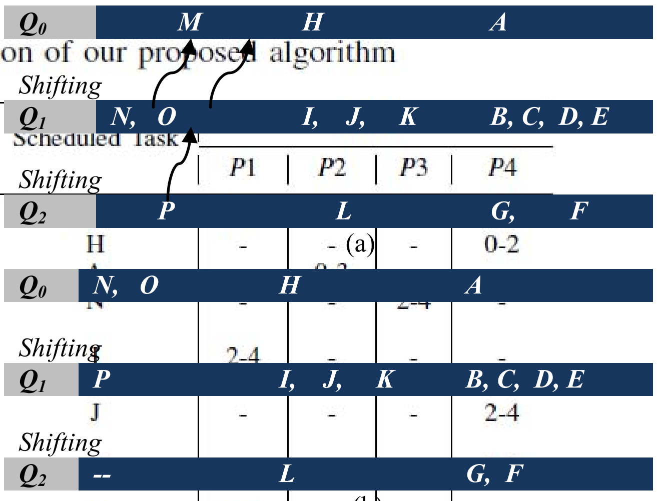
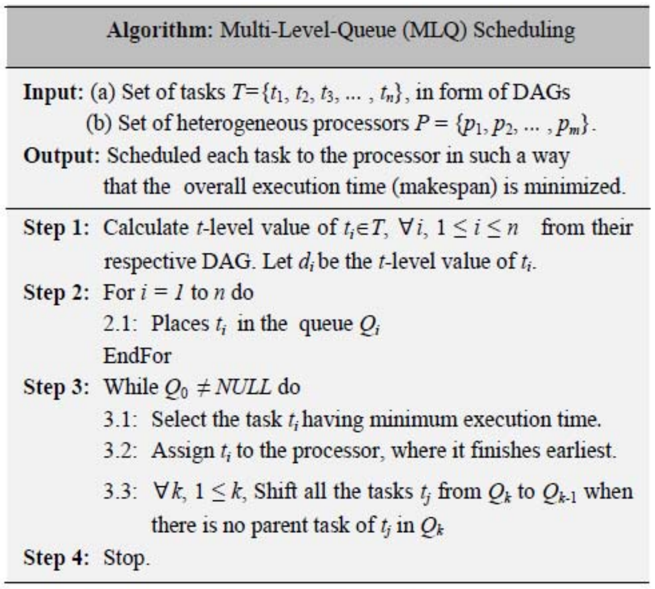


Fig. 4: (a) Initial position of the *MLQ* and (b) after assignment of *M*, the child tasks, i.e., *N*,*O* and *P* is shifted to on upper respective queues.

Step	Q_0	Q_1	Q_2	Scheduled Task	P1	P2	P3	P4
1	*M*, *H*, *A*,	N ,O, I, J, K, B, C, D, E	G, F ,L, P	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	H, A, N ,O,	I, J, K, B, C, D, E, P	G, F, L	H	-	-	-	0-2
3	A, N ,O,I, J, K,	B, C, D, E, P, L	G, F,	A	[illegible]	0-3	[illegible]	[illegible]
4	N ,O, I, J, K, B, C, D, E,	P, L, G, F,	-	N	-	-	2-4	-
5	O, I, J, K, B, C, D, E, P	L, G, F,	-	[illegible]	2-4	-	-	-
6	O, J, K, B, C, D, E, P	L, G, F	-	J	-	-	-	2-4
7	O, K, B, C, D, E, P, L	G, F	-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
8	O, K, C, D, E, P, L, G,	F,	-	D	4-6	-	-	
9	O, K, C, E, P, L, G,	F	-	E	-	-	4-6	-
10	O, K, C, P, L, G, F,	-	-	O	6-9	-	-	-
11	K, C, P, L, G, F,	-	-	K	-	3-6	-	
12	C, P, L, G, F,	-	-	F	-	6-8	-	-
13	C, P, L, G,	-	-	C	-	-		6-10
14	P, L, G,	-	-	G	-	-	-	10-13
15	L, P,	-	-	P	9-15	-	-	-
15	L,	-	-	L	-	6-13	-	-



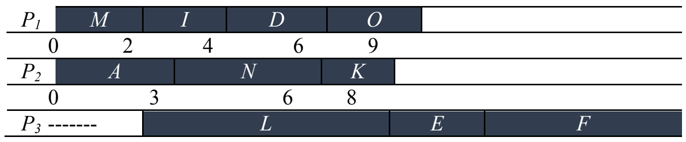
```
Algorithm: Multi-Level-Queue (MLQ) Scheduling

Input: (a) Set of tasks $T=\{t_1, t_2, t_3, \dots, t_n\}$, in form of DAGs
       (b) Set of heterogeneous processors $P=\{p_1, p_2, \dots, p_m\}$.
Output: Scheduled each task to the processor in such a way
        that the overall execution time (makespan) is minimized.

Step 1: Calculate t-level value of $t_i \in T$, $\forall i$, $1 \le i \le n$ from their
        respective DAG. Let $d_i$ be the t-level value of $t_i$.
Step 2: For $i = 1$ to $n$ do
            2.1: Places $t_i$ in the queue $Q_i$
        EndFor
Step 3: While $Q_0 \ne NULL$ do
            3.1: Select the task $t_i$ having minimum execution time.
            3.2: Assign $t_i$ to the processor, where it finishes earliest.
            3.3: $\forall k$, $1 \le k$, Shift all the tasks $t_j$ from $Q_k$ to $Q_{k-1}$ when
                 there is no parent task of $t_j$ in $Q_k$
Step 4: Stop.
```

Fig. 5: Multi-Level-Queue (MLQ) Scheduling Algorithm

Remark 1: The proposed work satisfies the dynamism property. The execution time of tasks and data dependencies between the tasks is not known in advance in the proposed algorithm. Here, tasks are allocated to processors according to their arrival and scheduled at run time. Scheduling decisions are based on dynamic parameters that may change during run time. Let at random instant of the a new task arrived, as per the algorithm t_{entry} where, $Pd(t_i)$ = ø are queued at *t-level* of t_{entry} according as in line number *1* in Table II.

V. Results and Discussion

Here, we have considered three different DAGs which may arrive at different instant of time as shown in Fig. 2 to implement the proposed work. As it has been shown the computed makespan for our taken DAGs is 15 as in Fig. 6. According to heterogeneous earliest finish time (*HEFT*) [12]. The rank is computed as shown in Table I. rank of the different task are computed based on the paper [12]. The execution sequence of the tasks according to the computed rank is follows- (*H, A, J, I, M, N, B, D, L, E, C, P, F, K, G, O*). The computed makespan of taken DAGs, is 18 as shown in Table III. The improvements of our proposed algorithm over existing algorithms are presented in Table III. In *meta-task* (*MT*) heuristic based task scheduling algorithm, where tasks are scheduled by two well renowned algorithms known as Min-Min and Max-Min the makespan and the complexity is highlighted in the Table III. We have implemented our algorithms on two different sets of DAG as shown in Fig. 2. The performance evaluation in terms of makespan is shown in Fig. 6 and Fig. 7.



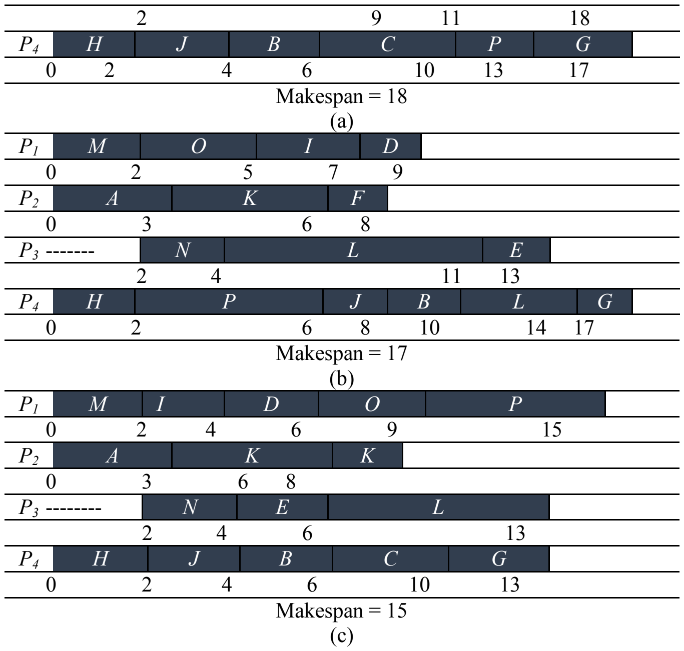


Fig. 6: Comparison of different makespan in (a) HEFT-B, (b) MT-based Min-Min and (c) Proposed

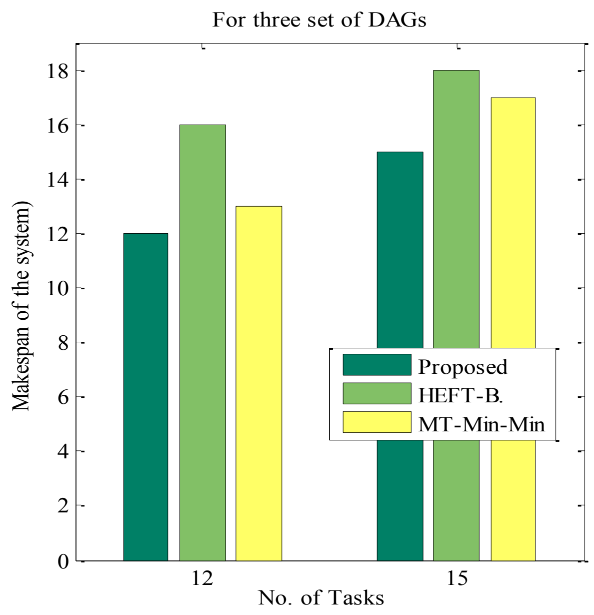


Fig. 7: Comparative status of Makespan

TABLE III: PERFORMANCE EVOLUTION METRICS OF DIFFERENT TASK SCHEDULING ALGORITHMS WITH THE PROPOSED WORK

Features	***Task Scheduling Algorithms***		
	HEFT	*MT*	*MLQ*
Makespan	18	17	15

Complexity	$O(T^2 \times P)$	$O(T^2 \times P)$	$O(T \times P)$
Resource Utilization	Less	Less	More
System Throughput	Less	Less	More
Dynamic	Na	Na	Yes

VI. Conclusions

In this paper, we have presented a novel heuristic approach for task scheduling in heterogeneous distributed computing system. The proposed algorithm (*MLQ*) considers dependency number into account and gives priority to those tasks have possibility to be completed in earliest finish time. The approach shows considerable improvement over HEFT and MT in terms of makespan, complexity and average processor utilization. In future, our attempt will be on two directions. Firstly, to apply some energy minimization method this is a great concern today. Secondly, developing a new scheme for scheduling in dynamic heterogeneous environment where tasks behavior changes after every execution.

References

[1] D. Ye, M. Zhang, and D. Sutanto, "Self-adaptation-based dynamic coalition formation in a distributed agent network: a mechanism and a brief survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 5, pp. 1042–1051, 2013

[2] J. Liu, X. Jin, and Y. Wang, "Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization," *IEEE Transactions on parallel and distributed systems*, vol. 16, no. 7, pp. 586–598, 2005.

[3] W. Rao, L. Chen, A. W.-C. Fu, and G. Wang, "Optimal resource placement in structured peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 1011–1026, 2010.

[4] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: A price-based approach," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 347–364, 2006.

[5] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? exploiting dynamic task migration in mobile cloud computing systems," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 24–32, 2013.

[6] Y. Jiang and J. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2743–2759, 2014.

[7] G. Attiya and Y. Hamam, "Task allocation for maximizing reliability of distributed systems: a simulated annealing approach," *Journal of parallel and Distributed Computing*, vol. 66, no. 10, pp. 1259–1266, 2006.

[8] Y. Jiang and J. Jiang, "Contextual resource negotiation-based task allocation and load balancing in complex software systems," IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 5, pp. 641–653, 2009.

[9] Q. Kang, H. He, and J. Wei, "An e_ective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems," Journal of Parallel and Distributed Computing, vol. 73, no. 8, pp. 1106–1115, 2013.

[10] S. Sharma and P. Kuila, "Design of dependable task scheduling algorithm in cloud environment," *in Proceedings of the Third International Symposium on Women in Computing and Informatics*. ACM, 2015, pp.516–521.

[11] B. Hong and V. Prasanna, "Adaptive allocation of independent tasks to maximize throughput," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1420–1435, 2007.

[12] Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Information Sciences*, vol. 270, pp. 255–287, 2014.

[13] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy efficient task scheduling algorithm in dvfs-enabled cloud environment," *Journal of Grid Computing*, vol. 14, no. 1, pp. 55–74, 2016.

[14] S. Parsa and R. Entezari-Maleki, "Rasa: A new task scheduling algorithm in grid environment*," World Applied sciences journal*, vol. 7, pp. 152–160, 2009.