New Batch Mode Scheduling Strategy for Grid Computing System

J.Y Maipan-uku[#1], J. Kok Konjaang[*2], Ayannor Issaka Baba[#3]

[#]Department of Mathematics and Computer Science, Faculty of Natural Sciences
Ibrahim Badamasi Babangida University Lapai, P.M.B 11, Niger State, Nigeria
jymaipounds@yahoo.com

[2,3]Department of Liberal Studies, Bolgatanga Polytechnic, Ghana
[2]larisco10@yahoo.com, [3]babscom002@yahoo.com

Abstract—**Effective scheduling algorithm to reduce total completion time and promote resource utilization with load balancing in a grid computing environment is required. Scheduling tasks on heterogeneous machines distributed over a grid system proves to be an NP complete problem. Many algorithms have been developed to counter this problem by researchers. However, it is obvious that, task selection is a key challenge to these heuristics. For this reason, a substantial enhancement in the computational efficacy of the algorithm might be welcome. In this paper, a new batch mode scheduling algorithm (MinExt) is proposed. The intent is to reduce the total completion time (makespan), utilization of idle resources and load balance. To achieve this, the proposed algorithm made an initial task queue, we collects the Average Completion Time (Act) of all tasks, then for all tasks greater than Act is scheduled first and follow by the set of tasks less than or equal to the Act. Our simulation results indicate that the algorithm minimizes total completion time and utilizes the idle resources effectively with load balancing in comparing to other algorithms.**

Keywords: Grid Scheduling, Proposed Scheduling Algorithm, Makespan, Resource Utilization, Load Balance

I. INTRODUCTION

This Grid computing system [1] are novel technology for building high-speed computing environment in which heterogeneous, homogeneous, distributed and dynamically resources integrated across the world through networks**.** A computational grid is a group of heterogeneous processors, and machines feast through several administrative fields with the intention of providing managers easy contact to these machines. It allows virtualization of dispersed computing and data machines such as processors, network bandwidth and storage volume to make a particular system [2]. Figure 1 depicts key steps in grid scheduling.

Grid Task scheduling had turned into major research aims, seeing as direct influences for performance of grid applications. It's described as the course of choosing the best resource for a suitable task. Grid task scheduling is a joint module of computing that efficiently uses the idle time of machines [3, 4]. Allocation strategy [5] is done in two categories; immediate and batch mode heuristics. In immediate mode, task is represented on a resource as quickly as it reaches the scheduler. While in Batch mode heuristics, tasks are not allocated on the resources as they reach; instead, they are collected into a set that is inspected for allocation at prescribe periods called mapping events**.** This paper considers a static batch mode scheduling algorithm.

The major contribution of this paperwork is to devise a new batch mode scheduling algorithm that is efficient for mapping independent tasks with the intensification of minimizing makespan, maximizing average resource utilization rate and loads balanced. The concept of the algorithm relies on Average completion time. A condition for mapping tasks to their paramount resources is considered, this enhances the efficiency of grid computing system environment.

The rest of this paper is as follows: Section 2 presents the related works along with several well- known scheduling policies. In Section 3, a new batch mode scheduling algorithm (MinExt) is proposed. Section 4 describes experimental setup, results and discussion. Finally, a Section 5 presents conclusions while Section 6 gave references.

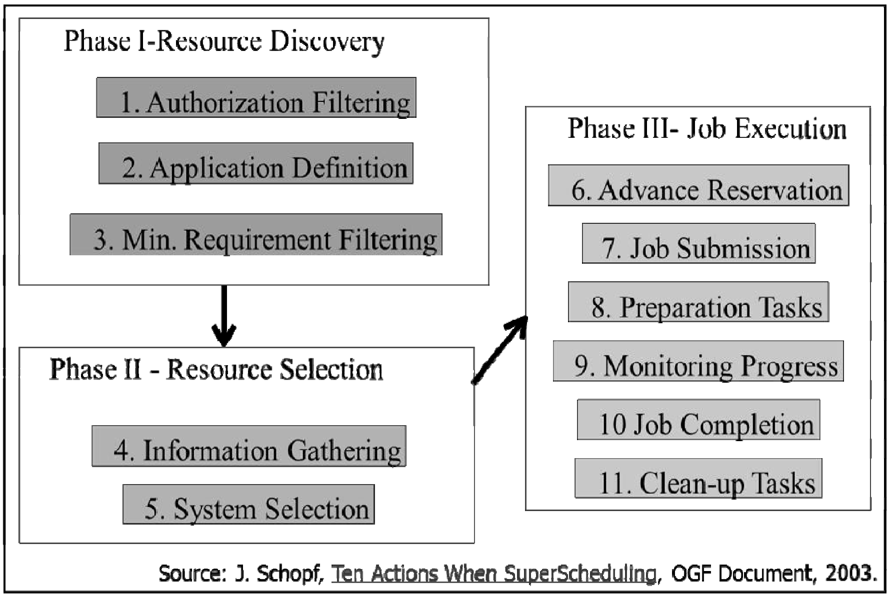

FIGURE I: key steps in grid scheduling

II. RELATED WORKS

Minimum execution time (MET) algorithm [6] assigns each task in arbitrary order to the machine with the minimum execution time without considering resource availability. However, minimum completion time (MCT) algorithm [6] assigned each task in arbitrary order to the machine with the earliest completion time. On the other hand, Min-Min algorithm discovered task with a minimum expected time and assign to the machine that yields its minimum completion time. The ready time of the resource is updated. This procedure is repeated executed until the entire task has been mapped [7]. While, Max-Min algorithm [8, 5] is the inverse of min-min policy which select task with the maximum completion time and mapped to resource with a minimum expected completion time of it.

Because of it widely used for mapping independent tasks in the heterogeneous computing system among other several heuristic developed by different researchers, min-min undergone a series of change and involve in multiple comparisons among other's heuristics, we present very few that are closest to our proposed heuristics.

Earlier work on static heterogeneous computing, scheduling in [6] was introduced by Ibarra and Chul, five special algorithms were computed, Min-Min inclusive. They also studied two other strategies: i.e., when tasks need to be scheduled on just two resources, and when the resources are of the same attributes.

Due to its ability in making it likely to gain high-quality solutions in a suitable runtime, large number of researchers have worked and revealed the benefits of MinMin algorithm for heterogeneous computing scheduling. Some of these works include the following. Maheswaran et al., [7] review four heuristics for dynamic mapping of a Class of Independent Tasks to Heterogeneous Computing Systems include Min-Min, Braun et al., [9] considered experimentally eleven algorithms for static scheduling in heterogeneous computing environments, this includes an extensive series of simple greedy constructive heuristic approaches and MinMin. Furthermore, Fujimota et al., in [10] compared scheduling algorithms for independent coarse-grain tasks; among them include MinMin. Then, Xhafa et al. [11] have also evaluated several static scheduling strategies for allocations of jobs on resources using the batch mode method, including MinMin. Similarly, Luo et al. [12] analysed and compared a set of twenty greedy heuristics under different conditions.

However, researchers have also proposed several extensions to Min-Min or new algorithms with several points of contact with this heuristic. Wu et al. [13] introduced Segmented Min-Min that secretly related to Min-Min. In this algorithm, tasks are sorted according to some score function of the expected time to compute in all machines (it could be the maximum, minimum or average expected time to compute among all machines). Then, the ordered sequence is segmented into groups, and finally MinMin is applied to schedule the group of larger tasks, followed by the other group. Furthermore, Hesam et al. [14] improve the efficiency of Min-Min in two phases: phase 1, Applied Min-Min for scheduling tasks, phase 2, task with minimum execution time on the fastest machine is divided by its execution time on the choice machine (in phase 1) has the maximum value is selected for mapping. In the same line of work: Kamalam and Bhaskaran [15], Soheil & Mahmoud in [16], Kfatheen and Banu [17], Bansal et al. [18] have contributed among others.

On the other hand, researchers presented interesting extensions to traditional Min-Min, known as Quality of Service (QoS). In [19] He et al., being the first that proposes a QoS guided MinMin heuristic, which guarantees QoS requirements by certain tasks while minimizing the makespan at the same time. The aim of this algorithm is that, some tasks may need more network bandwidth to exchange a large amount of data among processors, while others can satisfy with low network bandwidth. Hence, the task that required a high bandwidth will be assigned to resource that produces more network bandwidth. Singh & Suri [20] present QoS based Min-Min and Max-Min switcher algorithm for scheduling in Grid system. As Sharma & Bansal considered QoS in two forms; Computational based, and Communication based in [4] for details. Devipriya et al in [21] for cloud computing among others.

Meanwhile, other researchers put more effort on load balancing. Load Balanced Min-Min (LBMM) Algorithm for Static Meta-Task scheduling that applied Min-Min is the first step and rescheduled tasks from the most loaded machines to the idle or fewer loaded machines whose makespan is less in comparison to the loaded machines by Kokilavani et al. [22], Alharbi in [3] considered average completion time to find the greatest loaded machine and reschedule some of its tasks to fewer loaded machines. Then, Minal et al. [23] with Kfatheen et al. [24] present similar concept.

Moreover, some researchers believed that hybridizing Min-Min to Max-Min, which considers tasks with greatest execution time for mapping at first, contrary to Min-Min will yield a reasonable benefit to overcome the drawback of Min-Min. Etminani and Naghibzadeh presented selective algorithm in [25], Parsa et al. [26] introduced Resource Aware Scheduling Algorithm (RASA). In this algorithm, Min-Min is applied when the number of available machines is odd; otherwise Max-min is applied. Then, Gupta & Singh [27] have also proposed switcher algorithm that chooses between the two algorithms under a prescribed condition.

From all their viewed efforts, MinMin heuristic is commonly used by the community to solve scheduling problems. On the other hand, their solution shows that, mapping tasks to their best resources is an important challenge to this heuristic. For these reasons, a significant improvement in the computational efficiency of the algorithm could be welcome.

NOTATIONS	NOTATION DEFINITION
MinExt	Min-Min Scheduling Algorithm Extension
Act	Average completion time
MinECT	Minimum Execution Completion Time
Ti	Meta-task Id of meta-task i
Rj	Resource Id of resource j
Ci,j	Completion time for meta-task i on resource j
Xi,j	Execution time for meta-task i on resource j
Rj	Ready time of j
RU	Resource Utilization
MT	Meta-Tasks
Avgru	Average resource utilization
LB	Load Balancing
FIG	figure

ASSUMPTIONS

The following assumptions have taken, in this paper:

1) The experiments are carried out in a heterogeneous environment.

2) There is no priority among the meta-tasks / resources. Meanwhile, the meta-tasks / resources are independent of each other.

3) There is no deadline for the meta-tasks/resource.

4) Data sets are known prior.

III. PROPOSED ALGORITHM

The performance of Min-Min Grid scheduling algorithm gave the worst result in the case where a number of light tasks are more than the heavy ones, that is, a situation where a number of lighter tasks exceeded that of the heavier tasks, in this case max-min does better by executing lighter tasks concurrently with the heavy tasks. Moreover, since Min-Min algorithm attempts to assign the lighter tasks before heavy ones, it gives best makespan compare to Max-Min on a case where heavy tasks are much more than lighter ones. We present our proposed algorithm in FIGURE II. Firstly, all the tasks are sorted in non-decreasing order. This means tasks with minimum completion time are in the front and task with maximum completion time in the rear of the queue. Secondly, like traditional Min-Min, computes the completion time of all tasks on available resources and obtain their average. After that, the resource is chosen according to the proper condition. For all tasks greater than Act is scheduled first and follow by the set of tasks less than or equal to the Act.

Proposed Scheduling Algorithm

1. Sort all tasks in MT in non-decreasing order
2. While there are tasks in MT
3. For all submitted tasks in the set; T_i
4. For all resources; R_j
5. Calculate completion time (CT_{ij}) = xt_{ij} + rt_j; (for each task in all resources)
6. Find the Average CTij (Act)
7. For all tasks completion time
8. Schedule tasks greater than the Act. First
9. Schedule tasks less than or equal to the Act.
10. Remove the task from the set
11. Update ready time (rt_j) of the selected resource R_j
12. Update ct_{ij} for all T_i
13. End While

FIGURE II Pseudo code of the proposed algorithm (MinExt)

IV. SIMULATION, RESULTS AND DISCUSSIONS

To compare and assesses the efficiency of the proposed algorithm with existing algorithms in a heterogeneous environment as a grid, a simulation program in Java developed by [28] running on Intel(R) core(TM) i5-3470 CPU @ 3. 20GHz, 3.20GHz has been implemented. Our experimental testing performed using expected time to compute (ETC) matrix of 512 tasks x 16 resources as proposed in [9]. Table 3 displays the different scenario applied.

TABLE I. Tasks and resources Heterogeneity

Scenarios			Data set
1	HiHi	Heavy tasks along with high capacity resources	512 x 16 & 1024 x 32
2	HiLo	Heavy tasks along with low capacity resources	
3	LoHi	Light tasks along with high capacity resources	
4	LoLo	Light tasks along with low capacity resources	

There exist numerous performance measures to assess the quality of a scheduling algorithm. In this paper, the metrics used includes:

1. *Makespan*

It is the period taken by heuristic to finish a batch of jobs. It is a measure of efficiency and throughput of a grid computing system. Makespan is estimated by equation 1 & 2 as:

$$Makespan = \max \{completion\ [j] \mid j \text{ in Resource}\} \quad \text{Eq. 1}$$

$$\text{Completion Time } (CT_{ij}) = \text{Execution Time } (ET_i) + \text{Resource Ready Time } (R_j) \quad \text{Eq. 2}$$

TABLE IIA illustrates the values of makespan produces by different heuristics for four scheduling instances assumed in this study using 512 x 16 combinations. From the result, we figure out that, MinExt algorithm substantially outperforms all the existing algorithms in scenario one, two and three (HiHi, HiLo & LoHi). Similarly, Max-Min outperforms Min-Min, MCT and MET in the same three scenarios as proposed algorithm. However, MCT outperforms MET in scenarios one and two (HiHi & HiLo) while MET performs worst in scenario one and two (HiHi & HiLo) and gives the best result in scenario three (LoLo). Meanwhile, we display a comparative sketched of makespan between Min-Min and MinExt algorithms in FIGURE IIA**.**

Moreover, observations have shown from the values of makespan performances by different algorithms for 1024 x 32 combinations in TABLE V(a) that, MinExt gives better performance in all instances with significant differences in scenario one & two, slightly different in scenario three and a little different in the fourth instance. However, FIGURE III (a) demonstrates a comparison analysis of makespan between MinExt and Min-Min algorithm.

2. Resource Utilization

Maximizing resource utilization rate is one of the objective functions of a grid computing system. Resource utilization is achieved by minimizing the idle time of a resource. In Coa et al [29], resource utilizations *ru* of each resource can be calculated as in equation (3). In this paper, We collect average resource utilization of all algorithms which can be calculated using the equation (4)

$$ru = \sum \forall j, R_{ij=1} \frac{(R_{rt} - R_{it})}{makespan} * 100$$

Eq. 3

Where R_{rt} and R_{it} are resource running time and resource idle time

$$Avgru = \frac{\sum_{i=1}^{n}(ru)}{n}$$

{n = number of resources}

Eq. 4

TABLES II(b) and III(b) illustrate the values of *Avgru* for five heuristics. MinExt performed better than other algorithms in all instances achieving the best algorithm for resource utilization. Then followed by Min-Min, Max-Min, MCT and MET. While, FIG. II(b) & III(b) gave the comparison of Min-Min and the Proposed Algorithm (MinExt)

TABLE II. Makespan performance of different algorithms using 512 x 16

	MET	MCT	Min-	Max-	MinExt
HiHi	151.413	124.954	125.401	133.652	117.464
HiLo	207.188	179.933	173.687	162.572	153.781
LoHi	59.985	67.99	70.807	84.946	58.809
LoLo	56.991	94.707	92.61	90.341	75.293

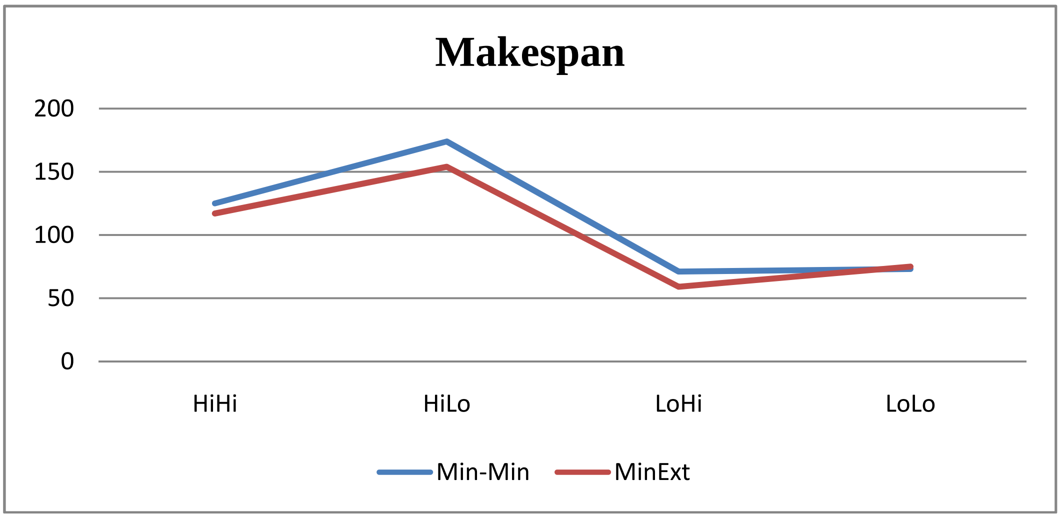

FIGURE IIA Makespan Comparison

TABLE II(B). Resource Utilization of different algorithms using 512 x 16

	MET	MCT	Min-Min	Max-Min	MinExt
HiHi	89	88	90	85	98
HiLo	81	79	83	91	97
LoHi	60	80	79	65	95
LoLo	47	74	76	80	97

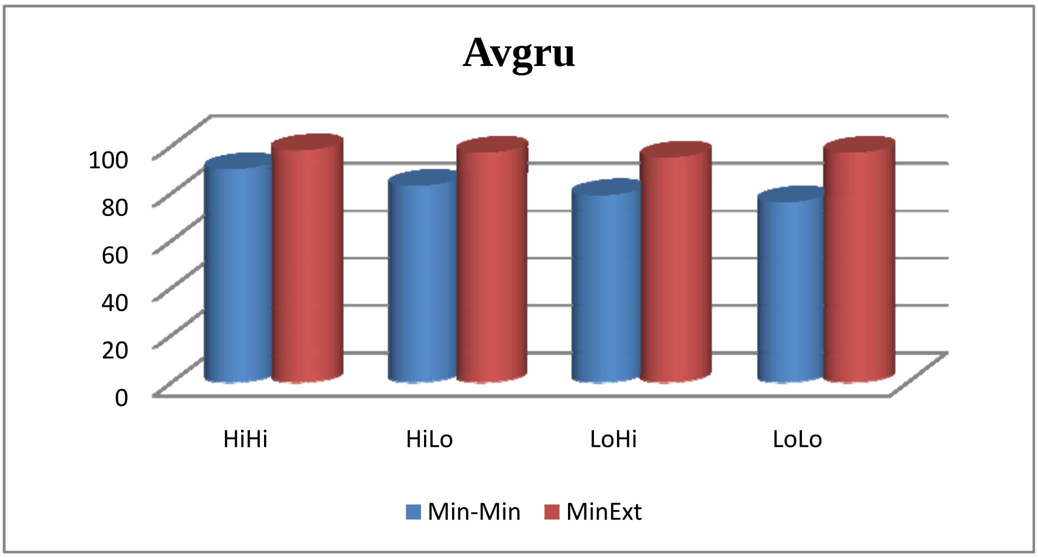

FIGURE IIB Average resource utilisation Comparison

TABLE III. Makespan performance of different algorithms using 1024 x 32

	MET	MCT	Min-Min	Max-Min	MinExt
HiHi	183.599	141.308	145.167	152.88	134.643
HiLo	237.134	205.166	200.979	204.333	157.733
LoHi	103.215	103.215	103.215	103.215	105.636
LoLo	113.215	114.502	120.578	147.449	108.075

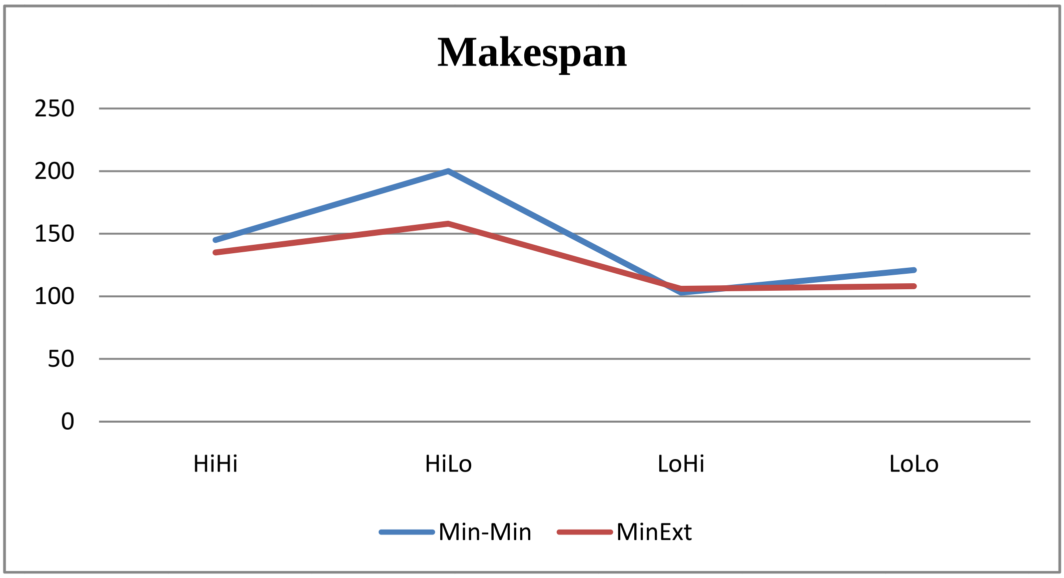

FIGURE IIIA Makespan Comparison

TABLE III(B). Resource Utilization of different algorithms using 1024 x 32

	MET	MCT	Min-Min	Max-Min	MinExt
HiHi	70	77	77	75	85
HiLo	68	66	69	73	94
LoHi	34	50	53	51	51
LoLo	28	59	56	49	67

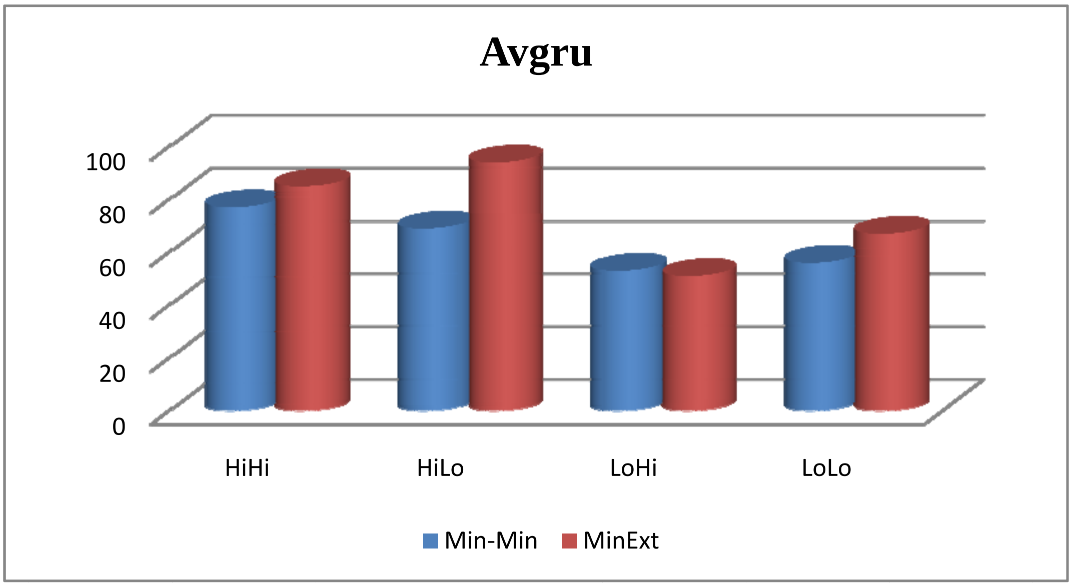

FIGURE IIIB Average resource utilisation Comparison

3. Load Balancing

Distributing load evenly to resources available on the grid system environment is one of the major objectives of grid computing systems. In [29] load balancing is calculated as in equation (5) below;

$$\beta = \left(1 - \frac{\mathrm{d}}{\mathrm{Avgru}}\right) * 100$$

Eq. 5

Where mean deviation (d) is:

$$d = \sqrt{\frac{\sum_{i=1}^{n}(Avgru - ru_i)^2}{n}}$$

TABLE IV illustrates the values of load balancing in percentage by different algorithms and FIGURE IV displays the Gantt chart of load balancing produces by the algorithms using 512 x 16 combinations. The outcomes show the consistent execution of the proposed algorithm by providing above 93% in all scenarios. Meanwhile, the existing algorithm (Min-Min) made a lower distribution of scores across the resources by performing below 85% in all scenarios except for combination of heavy tasks along with high capacity resources (HiHi) scenario that gets up to 90%. However, Max-Min performed well by utilizing the resources from 96% below, follow by MET in two scenarios and MCT in two scenarios.

Furthermore, TABLE V and FIGURE V demonstrate the values of load balancing is performed by different algorithms using 1024 x 32 combinations. From the results, we find out that the proposed algorithm outperforms Min-Min in scenario one, two and four while produced the same result with Min-Min in scenario three. This is because; the proposed algorithm was able to considerably distribute the jobs/tasks evenly among the resources in three scenarios.

TABLE IV. Load Balancing of different algorithms using 512 x 16

	MET	MCT	Min-Min	Max-Min	MinExt
HiHi	86	81	91	96	98
HiLo	85	82	83	95	99
LoHi	73	82	79	90	94
LoLo	50	83	76	93	96

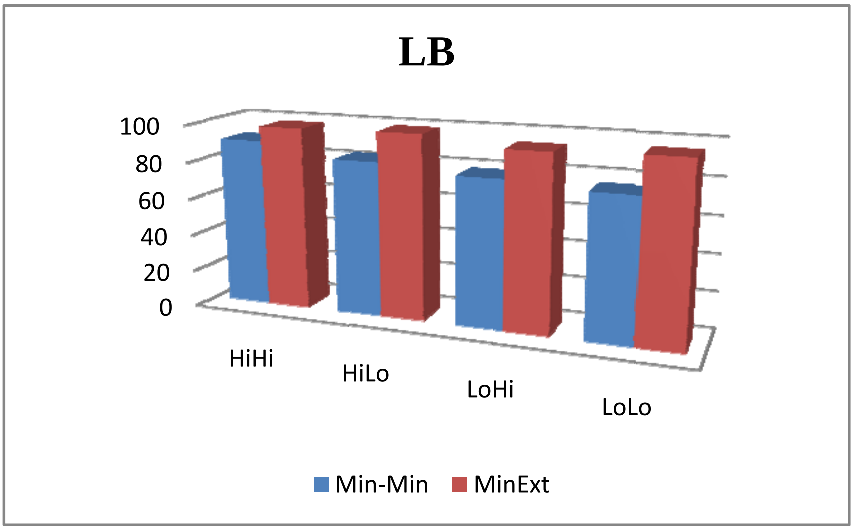

FIGURE IV Load Balancing Comparison

TABLE V. Load Balancing of different algorithms using 1023 x 32

	MET	MCT	Min-Min	Max-Min	MinExt
HiHi	75	67	77	82	88
HiLo	76	66	73	93	97
LoHi	33	47	70	55	70
LoLo	15	70	71	90	93

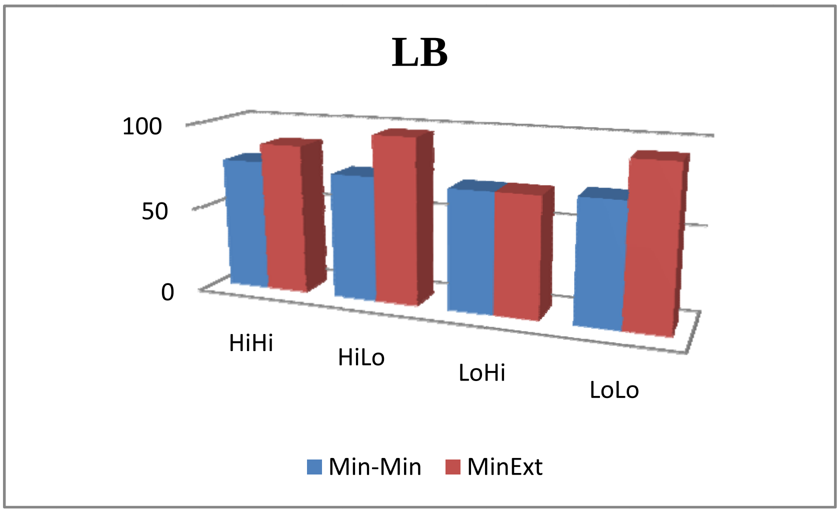

FIGURE V Load Balancing Comparison

V. CONCLUSIONS

Selecting the suitable machine for a specific task is a challenging attribute in the computational grid environment. This work introduces a new batch mode scheduling strategy called MinExt. MinExt and various existing algorithms are tested using developed grid simulation environment. Min-Min is a straightforward and regular scheduling algorithm for grid computing. However, it works inadequately when the number of small tasks is less than the number of heavy tasks. Furthermore, the computed makespan and grid's resource utilization by Min-Min in this case is not good. To circumvent the drawback of this grid's resource utilization and makespan with load imbalance by this algorithm, MinExt was introduced to minimize the makespan and to maximize grid's resource utilization with proper load balance. This algorithm conquers the liking of huge differs of task execution times. A comparison of makespan values and grid's resource utilization with load balancing between our designed algorithm and other four heuristics has been accomplished. Observably, the result of MinExt is better than all algorithms in four underlying instances. However, MinExt is the best for all instances. In conclusion, the rank of the proposed MinExt algorithm on both makespan, resource utilization, and Load balancing is excellent.

REFERENCES

[1] Singla, M.K, (2013).Task Scheduling Algorithms for Grid Computing with Static Jobs: A Review. International Journal of Computer Science Engineering (IJCSE). 2(5), pp.218.

[2] Kumar, R., Khan I. A. & Gupta V. D., (2013). Literature review on grid computing. African Journal of Mathematics and Computer Science Research. 6(7), pp.144.

[3] Alharbi, F, (2012). Simple Scheduling Algorithm with Load Balancing for Grid Computing. Asian Transactions on Computers. 2, pp. 8-9.

[4] Sharma, M.G., Bansal, P., (2012). Min-Min Approach f or Scheduling In Grid Environment. International Journal of Latest Trends in Engineering and Technology (IJLTET). 1, pp.26.

[5] Soheil, A., & Mahmoud, A., (2013). An Improved Min-Min Task Scheduling Algorithm in Grid Computing. © Springer-Verlag Berlin Heidelberg. 32, pp.103 – 106.

[6] Freund RF, Gherrity M, Ambrosius S, Campbell M, Halderman M, Hensgen D, et al. Scheduling resources in multi -user, heterogeneous, computing environments with SmartNet. HCW '98, Orlando, FL; 1998. p. 184–99.

[7] Maheswaran, M., Ali, S. and Siegel, H. J., Hensgen, D., & Freund, F. R., (1999). Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems1. Journal of Parallel and Distributed Computing. 59, pp.107 – 118.

[8] Vijayalakshmi, R., & Vasudevan, V. (2015). Static Batch Mode Heuristic Algorithm for Mapping Independent Tasks in Computational Grid. Journal of Computer Science, 11(1), pp. 224.

[9] Braun, T. D., Siegel, H. J. and Beck, N., (2001). A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing. 61, pp.823 – 831.

[10] Fujimoto, N., & Hagihara, K. (2004). A Comparison among Grid Scheduling Algorithms for Independent Coarse- Grained Tasks. 2(4), pp. 7-7.

[11] Xhafa, F., Barolli, L., & Durresi, A. (2007). Batch mode scheduling in grid systems. International Journal of Web and Grid Services, 3(1), pp. 19-19.

[12] Luo, P., & Shi, Z. (2007). A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems (K. Lü, Ed.). Journal of Parallel and Distributed Computing, 67, pp. 695-714.

[13] Min-You Wu, M.U., Shu, W. & Zhang, H., (2000). A Static Mapping Algorithm for Meta-tasks on Heterogeneous Computing Systems. IEEE . 1, pp.1 – 6.

[14] Hesam, I., Ajith, A., Senior member, IEEE, VS., (2009). Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environment. International Joint Conference on Computational Sciences and Optimization., pp.8 – 10.

[15] G.K, K., & Bhaskaran.V, M. (2010). A New Heuristic Approach:Min-mean Algorithm For Scheduling Meta-Tasks On Heterogeneous Computing Systems. IJCSNS International Journal of Computer Science and Network Security. 10(1).

[16] Soheil, A., & Mahmoud, A., (2013). An Improved Min-Min Task Scheduling Algorithm in Grid Computing. © Springer-Verlag Berlin Heidelberg. 32, pp.103 – 106.

[17] Kfatheen, S.V., Banu, M.N., & Selvi, S.K., (2014). TLLB: Two-Level Load Balanced Algorithm for Static Meta-Task Scheduling in Grid Computing. International Journal of Computer Applications. 105, pp.38 – 41.

[18] Bansal, J., Rani, S., & Singh, P., (2016) A Novel Heuristic for Scheduling of Independent jobs on Grid Resources. International Journal of Engineering and Technology (IJET). 7(6), pp. 2122 – 2129.

[19] HE. X, X-He Sun, and Laszewski. G.V., (2003). QoS Guided Min- Min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology. 18, pp. 442-451.

[20] Singh. M and Suri. P.K, QPS., (2008). A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid. Information Technology Journal. 7, pp. 1176-80.

[21] Devipriya, S., Magesh, B., & Ramesh, K.C., (2014).Enhanced Heuristic Model for Effective Resource Utilization in Cloud. International Journal of Computer Applications., pp. 4-9.

[22] Kokilavani, T. & Amalarethinam, D.I.G., (2011). Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. International Journal of Computer Applications. 20(2), pp. 43-47.

[23] Minal, S., Prof. Rajesh, B., & Prof. Rupesh, M., (2013). Task Assignment in Heterogeneous Environment with Advanced Scheduling Algorithms. International Journal of Advanced Research in Computer Science and Software Engineering. 3, pp.1079 – 1082.

[24] Etminani, K., Naghibzadeh, M., and Yanehsari, N.R., (2007). A Hybrid Min-Min Max-Min Algorithm with Improved Performance.Department of Computer Engineering, Ferdowsi University of Mashad, Iran.32 (1 - 8), pp.1 – 3.

[25] Parsa, S and Reza, E. M., (2009). RASA: A New Task Scheduling Algorithm in Grid Environment. World Applied Sciences Journal., pp.152-155.

[26] Gupta, K., & Singh, M., (2012). Heuristic Based Task Scheduling In Grid. International Journal of Engineering and Technology (IJET). 4 (254 - 260), pp.254 – 258.

[27] H. Nagda and D. Visariya, (2013). Distributed Project Final Report Task Scheduling in Grid Computing. Department of Computer Science Rochester Institute of Technology (RIT) Team Coda.

[28] Cao, J., Spooner, D., Jarvis, S., & Nudd, G. (2005). Grid load balancing using intelligent agents. Future Generation Computer Systems, 21(1), pp. 135-149.

Jamilu Yahaya Maipan-uku (Mr.) B.Sc. Computer Science @ Ibrahim Badamasi Babangida University Lapai (IBBUL), Nigeria. M.Sc. Computer Networks @ Universiti Putra Malaysia (UPM). Malaysia.

James Kok Konjaang received his HND in Marketing from Bolgatanga Polytechnic and BSc degree in Management with Computing from Regent University College of Science and Technology, both in Ghana in 2006 and 2012 respectively. He is currently pursuing a Master degree program in Distributed Computing from University Putra Malaysia. He is a senior instructor at Bolgatanga Polytechnic with more than five years teaching experience. His research interest includes Cloud Computing, Distributed Computing and Grid Computing.

Ayannor Issaka Baba received his B.ED in ICT from University of Education Winnerba - Kumasi, Ghana in 2010. He is currently pursuing a Master degree program in Information and Technology from Kwame Nkrumah University of Science and Technology, Ghana. He is working as a senior instructor, Department of Liberal Studies, Bolgatanga Polytechnic. His research interest includes Distributed Computing and Grid Computing.