Journal of Computer Science

Original Research Paper

A Great Deluge Algorithm with Bi-Decay Rate for Efficient Task Scheduling in Grid Computing

[1]KaiLun Eng, [1]Abdullah Muhammed, [1]Sazlinah Hasan and [2]Mohamad Afendee Mohamed

[1]Department of Communication Technology and Networking,
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia
[2]School of Computer Science, Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut, Malaysia

Article history
Received: 06-07-2017
Revised: 17-08-2017
Accepted: 01-03-2019

Corresponding Author:
Abdullah Muhammed
Department of Communication Technology and Networking,
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia,
Serdang, Malaysia
Email: abdullah@upm.edu.my

Abstract: To realise the utmost idea of global collaborative resource sharing with Grid computing, the fundamental scheduling process is playing a critical role. However, scheduling in Grid computing environment is a well-known NP-complete problem. In this study, we propose a new extension of Great Deluge algorithm with an effective diversification strategy for the Grid scheduling problem. The proposed approach, namely BiGD, exploits two different decay rates (a linear and a non-linear decay rate of water level) to provide a better diversification strategy for exploring the solution space. The performance of the proposed algorithm has been evaluated and compared with the standard Great Deluge and Extended Great Deluge algorithm, through the GridSim simulation toolkit. Four different scheduling scenarios or cases which comprise different combination of task heterogeneity and resource heterogeneity are considered for the performance evaluation. Moreover, we have adapted all the algorithms to have same total number of evaluation for solution searching in order to ensure a fair comparison is established in the performance evaluation. The experimental simulation results show that the proposed algorithm is superior and able to produce good quality solutions compared to the other algorithms in all the problem instances.

Keywords: Heuristic, Great Deluge, Extended Great Deluge, Diversification, Grid Scheduling Problem

Introduction

As new generation of information technologies and applications demand more and more computing power, Grid computing has become one of the most popular computing infrastructures to satisfy this ever-increasing demand of computing power. Grid computing is a form of computing infrastructure that allow users from anywhere to work together for obtaining the capability of executing computational intensive and data-intensive applications with the ideas of resource sharing and virtual organisation. The key concept of Grid computing is the ability to negotiate resource-sharing arrangements among the participating parties and then to use the resulting resource pool in a flexible, secure and coordinated fashion for some purpose (Lindner, 2002).

To fully realise the utmost concept of global collaborative resource sharing of Grid computing, the underlying resource management system which consists of techniques and mechanisms for resource discovery, negotiation, allocation, scheduling and monitoring is significant and critical. One of the most impactful functions in Grid resource management system is the scheduling process, which involves the assignment of application tasks to resources.

Scheduling in Grid computing which consists of various phases and steps is viewed as a whole family of problems (Xhafa and Abraham, 2008). The detailed description of the phases and steps of scheduling process that commonly defined by Grid researchers can be found in (Dong and Akl, 2006; Schopf, 2004; Yanmin and Ni, 2003). Among all the phases, the most important phase which has significant impact on the overall efficiency of Grid scheduling process is mapping. Basically, mapping can be considered as a function that produces a schedule from a given set of tasks and a set of appropriate resources. There are two types of mapping modes (Maheswaran *et al*., 1999), which are on-line mode (dynamic) and batch-mode (static). In the dynamic mode of mapping, a task is mapped onto a resource as soon as it arrives at the scheduler. In this mode of mapping, upcoming tasks or future tasks are not considered in the mapping decision. Whereas in the static mode of mapping, tasks are grouped in batches first and each task is considered at each mapping decision. In this study, the proposed scheduling algorithm is designed for global and static scheduling in Grid computing environment.

Although scheduling is a problem that has previously been studied in traditional distributed computing environments, scheduling in Grid computing environment is significantly challenging and demands a re-examination due to the unique characteristics of Grid which induce additional challenges in each phase and step of scheduling. Moreover, the task of mapping in Grid Computing environment is very large-scale due to the enormous amount of tasks and resources that need to be considered in each mapping decision. To solve such large-scale NP-Complete problem, exact optimisation method or exhaustive search is not viable. Therefore, many studies have been focused on developing a powerful heuristic algorithm that can solve this problem effectively and efficiently. However, developing a search algorithm to find an optimal or near-optimal solution effectively and efficiently for Grid scheduling problem is still a challenge. One of main challenges is to effectively prevent the search process from being trapped in local optima of search space. Therefore, diversification strategy is playing a significant role in the design of an efficient and effective heuristic search algorithm. In this study, the Great Deluge algorithm is extended with two different decay rates to provide a better diversification strategy for escaping from local optima and allowing the search process to examine wider regions of solution space.

This paper is organised as follows. Section 2 gives the concept of Great Deluge algorithm metaheuristic and provides a brief overview of the existing research on Great Deluge and scheduling problem in Grid computing environment. Section 3 presents the problem formulation and describes the proposed algorithm. Section 4 contains the detailed information of experiments setup for performance evaluation and results. Finally, Section 5 offers conclusion and future work.

Literature Review

The Great Deluge (GD) is a global search metaheuristic, which first proposed by (Dueck, 1993) as an alternative to Simulated Annealing (SA) metaheuristic. In contrast to SA, GD uses a deterministic acceptance function rather than a probability measure to accept worse neighboring solutions. GD employs a boundary level (water level) as the diversification strategy to guide the search. The water level is constantly decrease in a linear fashion throughout the search. In brief, Great Deluge metaheuristic always accepts a solution that is not worse than the current best solution and allows worse neighboring solution to be accepted only if its value of quality evaluation is same as or lower than the water level. This diversification strategy of GD algorithm is far less dependent upon parameters as compared to Simulated Annealing algorithm, because it needs just two parameters: The "rain speed" and the initial water level. The "rain speed" parameter is used to control the amount of computational time of the algorithm while the initial water level parameter is an estimate of the quality of initial solution. This is an advantage over SA since the effectiveness of a metaheuristic technique is often dependent upon parameter tuning.

To improve the standard GD, (McMullan, 2007) has proposed an extended version of the GD for the course timetabling problem. The standard GD has been extended by adding re-heat mechanism to widen the boundary condition when a lack of improvement has been observed for a specified amount of time. The motivation for this extension is to improve the convergence speed (obtain solutions in a relatively short amount of time) while avoiding the problem of getting trapped in local optima.

Implementation of this extended version of the GD algorithm for dynamic job scheduling problem in Grid environment has been presented by (McMullan and McCollum, 2007). Simple neighborhood structures such as adding, removing and swapping the tasks and resources, were employed in the algorithm to avoid the scheduler repeatedly getting stuck in local optima. According to the authors, the proposed extended GD algorithm can provide the means of achieving the balance between solution quality and speed in searching a solution by producing good quality schedules quickly. Experiments were carried out and the results showed that, given the limited time, the Extended Great Deluge generally performs better over Simulated Annealing in terms of computation time.

Landa-Silva and Obit (2008) made another extension to the original Great Deluge algorithm by proposing a non-linear decay rate for the boundary value (water level) in order to find a better quality neighborhood solution. The proposed method produced superior results in four out of the eleven course timetabling problem instances. Moreover, the results shown the proposed method is effective on medium problem instances whilst is not effective on the small instances.

A hyper-heuristic based scheduling algorithm which adopts Great Deluge as heuristic acceptance and Tabu Search as heuristic selection method is proposed by (Aron *et al.*, 2013). The proposed algorithm aims to minimise the makespan and cost (in Grid dollars) for scheduling of jobs in Grid environment. The performance of the proposed algorithm is evaluated using GridSim toolkit and (Braun *et al.*, 2001) simulation model with different test cases which consist of different heterogeneity of jobs and resources. From the results obtained, the authors conclude that the proposed algorithm is able to outperform Simulated Annealing (SA), Genetic Algorithm (GA) and GA-TS algorithm in all cases with respect to makespan and cost.

Noticeably, there has been a growing interest in attempting to improve the efficiency of search process by enhancing the diversification strategy in metaheuristic algorithms. The aim of this study is to develop a global search algorithm that offers a better strategy for diversification in such a manner that a good quality of solution can be obtained while avoiding from being trapped in the local optimum. Therefore, a novel heuristic based algorithm, namely BiGD, which extends from Great Deluge metaheuristic, is proposed with the hope to improve the ability of escaping from local optima by exploiting two different decay rates as the diversification strategy.

Problem Formulation

In this study, the Grid system considered is consists of a set of heterogeneous resources, which located at different sites and they are coordinated by a Grid scheduler. Given n independent tasks T_1, T_2, T_3,…,T_n, which are submitted by users to Grid scheduler in a particular interval of time for scheduling and m heterogeneous resources R_1, R_2, R_3,…,R_m, which located by GIS, the problem is to generate an optimal mapping or schedule that can optimise the objective function. In this study, this problem is formulated based on the Expected Time to Compute (ETC) Matrix Model (Ali *et al.*, 2000) and the following constraints are considered:

1. The computing load or length (in millions of instructions, MI) of each task is known
2. A task can only be executed in one Grid resource in each interval
3. Task migration and pre-emptive process are not allowed
4. Tasks which submitted to Grid scheduler are scheduled in batch mode
5. A Grid resource may consist of multiple machines that may have one or more than one processor
6. The processing power of each resource (in millions of instructions per second, MIPS) is known
7. A Grid resource cannot remain idle when it has tasks in queue and free processors
8. A processor can only process one task at a time

Solution Representation

In this study, direct-based representation is used to encode the schedules or solutions. The size of the vector is equal to the number of tasks and the index number of the element in the vector denotes the ID of task. The element in the vector indicates the resource that assigned for the corresponding task. All the elements are integers, which in the range of [1, m], where m is the total number of resources. These values in the vector can be repetitive, which means that different tasks can be scheduled to a same resource. For example, given six tasks and three resources (with ID: 1, 2, 3), let assume that the first task and the last task are assigned to Resource 3, the second task to Resource 1, the third and fifth task to Resource 2 and the fourth task to Resource 3. With the direct representation, let *schedule* be the vector denoting a solution, the solution can be represented as *schedule* = [3, 1, 2, 3, 2, 3].

Solution Evaluation

In order to describe the quality of each solution and guide the search process over the solution space, an evaluation function is used to associates a real value to every solution. As makespan is one of the most popular metric used for representing the quality of a schedule for Grid scheduling problem, the evaluation function in this study is defined as a function that gives a makespan value from any given candidate schedule. Under the ETC matrix model, the evaluation function can be expressed as:

$$makespan = \max\left\{completionTime\left[r\right] \mid r \in Resources\right\} \quad (1)$$

where, *completionTime*[*r*] is the time when resource r has finished executing all the tasks that assigned to it. Meanwhile, the completion time of resource r can be expressed as:

$$completionTime\left[r\right] = readyTime\left[r\right] + \sum_{\left\{t \in Tasks \mid S_i\ \left[t\right] = r\right\}} ETC\left[t\right]\left[r\right] \quad (2)$$

where, *readyTime*[*r*] is the time when resource r has finished executing all the previously assigned tasks.

Objective Function

The objective function is the function that needs to be satisfied in order to achieve our goal. In this study, we have considered the most well-studied optimisation criteria, i.e., the minimisation of the makespan (Rajni and Chana, 2013; Xhafa and Abraham, 2010) and it is formulated by defining the objective function as the evaluation function.

Accordingly, let $f(S)$ denotes the evaluation function or objective function and *Schedules* denotes the set of all possible schedules, the Grid Scheduling problem is formulated as:

$$\min_{S_i \in Schedule} f(S) = \max\left\{completionTime[r] \mid r \in \mathrm{Re}sources\right\} \quad (3)$$

Proposed Algorithm

It has been observed that in the standard Great Deluge algorithm (Dueck, 1993; McMullan and McCollum, 2007), the decay rate at which the boundary value *B* decreases, is determined by a linear function ($B = B - \Delta B$, where ΔB is a constant). On the other hand, (Landa-Silva and Obit, 2008) proposed a non-linear decay rate for the water level in order to find a better quality of solution. In this study, we propose a new extension of Great Deluge algorithm with bi-decay rate, which uses a linear and a non-linear decay rate of water level to efficiently guide the search in finding a better quality of solution meanwhile preventing it from getting stuck in the local optima.

In the first half of iterations, our proposed algorithm will exploit the strength of non-linear decay rate of water level (Landa-Silva and Obit, 2008) (shown in Equation 4) to guide the diversification of search. Furthermore, we allow the water level to decrease immediately to the same level of current best makespan when a significant improvement is found. Alternatively, we decelerate the decay rate of water level when the water level has dropped too much below the current best makespan. Two parameters, Fbottom and Ftop are used to control this floating strategy (in line 12 of Algorithm 1). This floating water level strategy, together with the exponential function of decay rate are aim to further improve the diversification of search by allowing wider search of solution in hoping to get a new better quality of solution.

$$B = B \times \left(\exp^{-\delta(rnd[\min, \max])}\right) + \beta \tag{4}$$

In the second half of iterations, we however, use a linear (steeper) decay rate to help algorithm to speed up the search of optimal solution. The initial decay rate of boundary in the second phase is reset using reheat mechanism (McMullan, 2007) at the quarter of the initial solution penalty cost. The boundary value of this phase is constantly decreased in a linear fashion based on the remaining iteration. To avoid wasting much computation time in generating neighboring solutions, two simple neighborhood structures which based on swap move and insertion move (Xhafa, 2007; Xhafa and Abraham, 2010) have been implemented in this study. Pseudo-code of the BiGD algorithm is given in Algorithm 1.

Algorithm 1: BiGD algorithm for minimization

```
Input: $x$, $rainSpeed$, $totalIter$, $min$, $max$, $\delta$, $\beta$
Output: $x$
1: $waterLevel \leftarrow f(x)$
2: $decayRate \leftarrow f(x) \times rainSpeed \div totalIter$
3: $halfIter \leftarrow (totalIter \div 2)$
4: $reheated \leftarrow$ false
5: for $iter = 1$ to $totalIter$ do
6:    $x' \leftarrow$ generateNeighbor(x)
7:    if ($f(x') \le f(x)$ or $f(x') \le waterLevel$) then
8:       $x \leftarrow x'$
9:    endif
10:   if $iter \le halfIter$ then
11:      $diff \leftarrow waterLevel - f(x)$
12.      if $diff < F_{bottom}$ or $diff > F_{top}$ then
13:         $waterLevel \leftarrow f(x)$
14:      else
15:         $waterLevel \leftarrow waterLevel \times (\exp^{-\delta(rnd[min, max])}) + \beta$
16:      endif
17:   else
18:      if $reheated$ then
19:         $waterLevel \leftarrow waterLevel - decayRate$
20:      else
21:         $waterLevel \leftarrow f(x)$
22:         $reheated \leftarrow$ true
23:      endif
24:   endif
25: endfor
```

Illustration of Algorithm

A small-scale task scheduling problem, which involves 5 tasks and 3 resources, is used to illustrate mathematical calculation related to the algorithm process. Let us consider the task lengths of 5 tasks are 3, 1, 2, 6 and 5 MI, respectively, whereas the processing power of 3 resources are 1, 2 and 3 MIPS, respectively. Assume that all the resources are available and ready time for each resource is 0:

Initialisation

Given that initial solution, $x = [1, 1, 1, 1, 1]$, $rainSpeed = 0.3$, $totalIter = 6$, $min = 1000$, $max = 1500$, $\delta = 0.00005$, $\beta = 0$, $F_{top} = 10$ and $F_{bottom} = -5$:

$f(x) = \max(17, 0, 0) = 17$
$waterLevel = 17$
$decayRate = 17 \times 0.3 \div 6 = 0.85$
$halfIter = 3$
$reheated =$ false

First Iteration

Assume that a neighboring solution of x, which denoted as x', is generated by inserting 4th task into Resource 2 and a random number, 1022 is generated from:

$rnd[1000, 1500]$
$x' = [1, 1, 1, 2, 1]$
$f(x') = \max(11, 3, 0) = 11$
$x = x'$
$diff = 17 - 11 = 6$
$waterLevel = 17 \times (\exp^{-0.00005(1022)}) + 0 = 16.153122$

Second Iteration

Assume that a neighboring solution of *x*, which denoted as *x'*, is generated by inserting 3rd task into

Resource 2 and a random number, 1373 is generated from *rnd*[1000, 1500]:

x' = [1, 1, 2, 2, 1]
f(x') = max(9, 4, 0) = 9
x = *x'*
diff =16.153122 – 9 = 7.153122
waterLevel = 16.153122 × ($\exp^{-0.00005(1373)}$) + 0 = 15.081417

Third Iteration

Assume that a neighboring solution of *x*, which denoted as *x'*, is generated by inserting 1st task into Resource 3 and a random number, 1279 is generated from *rnd*[1000, 1500]:

x' = [3, 1, 2, 2, 1]
f(x') = max(6, 4, 1) = 6
x = *x'*
diff = 15.081417 – 6 = 9.081417
waterLevel = 15.081417 × ($\exp^{-0.00005(1279)}$) + 0 = 14.147152

Fourth Iteration

Assume that a neighboring solution of *x*, which denoted as *x'*, is generated by swapping 1st task which assigned to Resource 3 with 4th task which assigned to Resource 2 and a random number, 1154 is generated from *rnd*[1000, 1500]:

x' = [2, 1, 2, 3, 1]
f(x') = max(6, 2.5, 2) = 6
x = *x'*
waterLevel = 6
reheated = true

Fifth Iteration

Assume that a neighboring solution of *x*, which denoted as *x'*, is generated by swapping 5th task which assigned to Resource 1 with 3rd task which assigned to Resource 2 and a random number, 1304 is generated from *rnd*[1000, 1500]:

x' = [2, 1, 1, 3, 2]
f(x') = max(3, 4, 2) = 4
x = *x'*
waterLevel = 6 – 0.85 = 5.15

Sixth Iteration

Assume that a neighboring solution of *x*, which denoted as *x'*, is generated by inserting 3rd task into Resource 2 and a random number, 1379 is generated from *rnd*[1000, 1500]:

x' = [2, 1, 2, 3, 2]
f(x') = max(1, 4.5, 2) = 4.5
x = *x'*
waterLevel = 5.15 – 0.85 = 4.3

Performance Evaluation

Simulation experiments have been carried out by using the GridSim simulator (Buyya and Murshed, 2002) to test and evaluate the proposed algorithm. GridSim is a very popular simulator which has been widely used by Grid researchers to evaluate the performance of their proposed algorithms (Anderson *et al.*, 2012; Aron *et al.*, 2013; Naik and Satyanarayana, 2013; Rajni and Chana, 2013; Maipan-Uku *et al.*, 2016). The simulation was coded using JAVA programming language and implemented on Eclipse IDE. All the experiments were performed in the environment of Window 10 Pro with 64-bit and run on a PC with Intel Core i5-3470 CPU 3.20 GHz and 8 GB RAM. The proposed algorithm, BiGD was evaluated and compared with the standard GD and EGD in order to investigate the performance of search by introducing the bi-decay rate diversification strategy into GD algorithm. Four different scheduling scenarios or cases which comprise different combination of task heterogeneity and resource heterogeneity are considered for the performance evaluation. Every algorithm was repeated run for 30 times to obtain the average and best results of makespan. We have adapted all the algorithms to have same total number of evaluation. The settings of the parameters of GridSim environment for all the experiments are shown in Table 1 while the parameter settings of the algorithms are shown in Table 2.

Performance Metrics

In this study, two performance metrics, namely makespan and computation time, are selected to evaluate the performance of our proposed algorithm. Makespan is one of the most popular performance measures for evaluating scheduling algorithm in heterogeneous computing environment (Xhafa and Abraham, 2010). Makespan can be an indicator of the productivity of a Grid scheduler. A small value of makespan indicates the scheduling algorithm which adopted by the Grid scheduler is effective as it is able to produce a good quality of solution. In GridSim simulation, the makespan of a schedule can be defined as the time when the last task is finished and can be expressed as:

$$makespan = \max\{finishTime[i] \mid i \in Tasks\} \quad (5)$$

Second performance metric is computation time. Computation time is the time taken for the scheduling algorithm in Grid Scheduler to produce a solution. It can be used to determine the efficiency of a scheduling algorithm in finding a good quality of schedule. Small values of computation time indicate that the scheduling algorithm is efficient as it does not required to spend huge amount of search time in obtaining a good quality of solution.

Table 1: Simulation parameters of GridSim

Parameters	Values
Number of tasks	512
Input file size of task	100+(10%-40%)B
Output file size of task	100+(10%-40%)B
Task length for LoLo	10-1000 MI
Task length for LoHi	1000-100000 MI
Task length for HiLo	10-30000 MI
Task length for HiHi	1000-3000000 MI
Number of resources	16
Allocation policy of resources	Space-Shared
Number of machines per resource	1-16
Number of PE per machine	1-4
PE rating for low resource heterogeneity	1-10 MIPS
PE rating for high resource heterogeneity	1-1000 MIPS
Baudrate of resources and scheduler	10,000,000,000.0
Propagation delay	0.0005 sec

Table 2: Parameters of algorithms

Algorithms	Parameters	Values
GD	rainSpeed	0.5
	totalIter	50000
EGD	rainSpeed1	0.5
	rainSpeed2	0.75
	wait	15
	totalIter	50000
BiGD	rainSpeed	0.3
	min	1000
	max	1500
	δ	0.000000005
	β	0
	Ftop	100
	Fbottom	-50
	totalIter	50000

Results and Discussion

In this section, the results of best makespan and best average makespan among all the algorithms are presented. In addition, the values of computation time and number of evaluation for each algorithm are also reported. It should be mentioned again that in the experiments for evaluating the performance of BiGD algorithm, all the algorithms were configured to have same total number of evaluation in order to have a fair comparison among them.

The resulting makespan values of the schedule produced by the algorithms for low task and low resource heterogeneity (LoLo) scheduling case are illustrated in Table 3. In terms of the best makespan obtained, both EGD and BiGD algorithms are able to obtain the best schedule with makespan value of 219.45 sec, whereas in terms of average makespan of 30 runs of experiments, BiGD algorithm is performing better than GD and EGD algorithms, with the lowest makespan value of 222.55 sec been generated.

The comparison results for low task and high resource heterogeneity (LoHi) scheduling case are presented in Table 4. In this case, BiGD is again able to produce the most optimal schedule with makespan value of 258.32 sec. In terms of average makespan, BiGD is also able to achieve the lowest among all the algorithms with makespan value of 273.33 sec.

Table 5 presents the results of all the algorithms for high task and low resource heterogeneity (HiLo) scheduling case. From the makespan results, although it is showed that the best schedule obtained by BiGD (5936.34 sec) is slightly worse than EGD (5936.34 sec), but BiGD is able to outperform EGD and also GD in terms of average makespan. In average, BiGD achieved the lowest makespan (6162.03 sec) among all the algorithms.

As shown in Table 6, it is observed that BiGD is still able to provide excellent performance in high task and high resource heterogeneity (HiHi) scheduling case. BiGD obtained the most optimal schedule with makespan value of 7357.29 sec. In addition, BiGD obtained the best average makespan result with 7665.77 sec.

Overall, from the results of all the scheduling cases, it reveals that the linear decay rate diversification strategy of the GD algorithm is not effective in helping the search to jump out from local optima as the GD algorithm performed worst among all the algorithms. On the other hand, the reheat mechanism of EGD algorithm appeared to be a good diversification strategy as the EGD algorithm is able to obtain the best results of best makespan in two out of four scheduling cases (LoLo and HiLo), whereas the proposed BiGD algorithm obtained the best results in most of the scheduling cases except HiLo. These findings indicate that the bi-decay rate diversification strategy is able to help the GD algorithm to jump out from local optima effectively and at the same time widen the search in order to obtain new solution with better makespan. The average makespan results showed that the BiGD is able to consistently perform better than the GD and EGD algorithms in guiding the search to obtain a better solution while escaping from local optima.

Table 3: Comparison results for the LoLo scheduling case

Algorithm	Makespan (s) Best	Makespan (s) Average	Computation time (s) Min	Computation time (s) Average	Number of evaluation
GD	225.39	228.01	13.62	15.71	50000
EGD	**219.45**	222.98	14.55	15.68	50000
BiGD	**219.45**	**222.55**	14.39	15.16	50000

Table 4: Comparison results for the LoHi scheduling case

Algorithm	Makespan (s) Best	Makespan (s) Average	Computation time (s) Min	Computation time (s) Average	Number of evaluation
GD	289.46	292.12	13.09	15.00	50000
EGD	261.52	275.56	13.04	13.79	50000
BiGD	**258.32**	**273.33**	13.93	13.93	50000

Table 5: Comparison results for the HiLo scheduling case

Algorithm	Makespan (s) Best	Makespan (s) Average	Computation n time (s) Min	Computation n time (s) Average	Number of evaluation
GD	6319.29	6386.94	13.36	16.36	50000
EGD	**5902.54**	6166.56	13.07	14.51	50000
BiGD	5936.34	**6162.03**	13.16	13.26	50000

Table 6: Comparison results for the HiHi scheduling case.

Algorithm	Makespan (s) Best	Makespan (s) Average	Computation n time (s) Min	Computation n time (s) Average	Number of evaluation
GD	8214.33	8287.38	12.19	15.08	50000
EGD	7364.45	7867.04	13.85	14.50	50000
BiGD	**7357.29**	**7665.77**	13.08	13.33	50000

Conclusion

In this study, a new extension of Great Deluge algorithm with two different decay rates (a linear and a non-linear decay rate of water level) is proposed to provide a better diversification strategy for exploring the solution space of Grid scheduling problem. Simulation experiments have been carried out to test and evaluate the performance of the proposed algorithm. Computation time and number of evaluation are also reported to show that a fair comparison of performance evaluation is established in this study. Four different scheduling scenarios or cases which comprise different combination of task heterogeneity and resource heterogeneity are considered for the performance evaluation. From the experimental results, it is concluded that the novel bi-decay rate diversification strategy is effective in preventing premature convergence to local optima and hence beneficial to improve the performance of search.

Funding Information

This research was supported by the Malaysian Ministry of Higher Education [Grant No: FRGS/1/2014/ICT03/UPM/03/1].

Author's Contributions

All authors are equally contributed in this work and this paper.

KaiLun Eng: Designed the study, developed the methodology, collected the data, performed the analysis, and wrote the manuscript.

Abdullah Muhammed: Invented the idea of GD bi-decay rate for his FRGS research proposal and been implemented in this work, helped shape the research, assisted in data analysis and interpretation, and provided critical feedback and comments on writing.

Sazlinah Hasan: Helped shape the research, assisted in research design and methodology, and provided critical feedback and comments on writing.

Mohamad Afendee Mohamed: Helped shape the research, assisted in data collection and interpretation, and provided critical feedback and comments on writing.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Ali, S., H.J. Siegel, M. Maheswaran, D. Hensgen and S. Ali, 2000. Task execution time modeling for heterogeneous computing systems. Proceeding of the 9th Heterogeneous Computing Workshop, May 1-1, IEEE Xplore Press, Cancun, Mexico, pp: 185-199. DOI: 10.1109/HCW.2000.843743

Anderson, D., C. Zhao, C. Hauser, V. Venkatasubramanian and D. Bakken *et al*., 2012. Intelligent design" Real-time simulation for smart grid control and communications design. IEEE Power Energy Magazine, 10: 49-57. DOI: 10.1109/MPE.2011.943205

Aron, R., I. Chana and A. Abraham, 2013. Hyper-heuristic based resource scheduling in grid environment. Proceedings of the International Conference on Systems, Man and Cybernetics, Oct. 13-16, IEEE Xplore Press, Manchester, UK, pp: 1075-1080. DOI: 10.1109/SMC.2013.187

Braun, T.D., H.J. Siegel, N. Beck, L.L. Bölöni and M. Maheswaran *et al*., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distributed Comput., 61: 810-837. DOI: 10.1006/jpdc.2000.1714

Buyya, R. and M. Murshed, 2002. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency Comput. Pract. Exp., 14: 1175-1220. DOI: 10.1002/cpe.710

Dong, F. and S.G. Akl, 2006. Scheduling algorithms for grid computing: State of the art and open problems.

Dueck, G., 1993. New optimisation heuristics: The great deluge algorithm and the record-to-record travel. J. Comput. Phys., 104: 86-92. DOI: 10.1006/jcph.1993.1010

Landa-Silva, D. and J.H. Obit, 2008. Great deluge with non-linear decay rate for solving course timetabling problems. Proceeding of the 4th International IEEE Conference Intelligent Systems, Sept. 6-8, IEEE Xplore Press, Varna, Bulgaria. DOI: 10.1109/IS.2008.4670447

Lindner, B., L. Schimansky-Geier and A. Longtin, 2002. Maximizing spike train coherence or incoherence in the leaky integrate-and-fire model. Phys. Rev. E., DOI: 10.1103/PhysRevE.66.031916

Maheswaran, M., S. Ali, H.J. Siegal, D. Hensgen and R.F. Freund, 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings of the 8th Heterogeneous Computing Workshop, Apr. 12-12, IEEE Xplore Press, San Juan, pp: 30-44. DOI: 10.1109/HCW.1999.765094

Maipan-Uku, J.Y., A. Muhammed, A. Abdullah and M. Hussin, 2016. Max-Average: An extended max-min scheduling algorithm for grid computing environtment. J. Telecommunication, Electronic Comput. Eng., 8: 43-47.

McMullan, P. and B. McCollum, 2007. Dynamic job scheduling on the grid environment using the great deluge algorithm. Proceedings of the International Conference on Parallel Computing Technologies, (PCT'07), Springer International Publishing AG, pp: 283-292. DOI: 10.1007/978-3-540-73940-1_29

McMullan, P., 2007. An extended implementation of the great deluge algorithm for course timetabling. Proceeding of the International Conference on Computational Science (CCS‘2007), Springer Berlin Heidelberg, pp: 538-545. DOI: 10.1007/978-3-540-72584-8_71

Naik, K.J. and N. Satyanarayana, 2013. A novel fault-tolerant task scheduling algorithm for computational grids. Proceedings of the 15th International Conference on Advanced Computing Technologies, Sep. 21-22, IEEE Xplore Press, Rajampet, India, pp: 1-6. DOI: 10.1109/ICACT.2013.6710529

Rajni and I. Chana, 2013. Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. Future Generation Comput. Syst., 29: 751-762. DOI: 10.1016/j.future.2012.09.005

Schopf, J.M., 2004. Ten actions when grid scheduling. Proceedings of the Grid Resource Management, (GRM'04), Springer US, pp: 15-23. DOI: 10.1007/978-1-4615-0509-9_2

Xhafa, F. and A. Abraham, 2008. Meta-heuristics for grid scheduling problems. Proceedings of the Metaheuristics for Scheduling in Distributed Computing Environments, (DCE'08), Springer International Publishing AG, pp: 1-37. DOI: 10.1007/978-3-540-69277-5_1

Xhafa, F. and A. Abraham, 2010. Computational models and heuristic methods for grid scheduling problems. Future Generation Comput. Syst., 26: 608-621. DOI: 10.1016/j.future.2009.11.005

Xhafa, F., 2007. A Hybrid Evolutionary Heuristic for Job Scheduling on Computational Grids. In: Hybrid Evolutionary Algorithms. Studies in Computational Intelligence, Abraham, A., C. Grosan and H. Ishibuchi (Eds.), Springer Berlin Heidelberg, ISBN-10: 978-3-540-73296-9, pp: 269-311.

Yanmin, Z. and L.M. Ni, 2003. A survey on grid scheduling systems. Department of Computer Science, Hong Kong University of Science and Technology.