Journal of Computer Science

Original Research Paper

# Fault Tolerance Grid Scheduling with Checkpoint Based on Ant Colony System

**[1]Saufi Bukhari, [2]Ku Ruhana Ku-Mahamud and [3]Hiroaki Morino**

*[1,2]School of Computing, Universiti Utara Malaysia, Malaysia*
*[3]Graduate School of Engineering and Science, Shibaura Institute of Technology, Japan*

*Article history*
Received: 15-05-2017
Revised: 25-08-2017
Accepted: 24-08-2017

Corresponding Author:
Saufi Bukhari
School of Computing,
Universiti Utara Malaysia,
Malaysia
Email: saufi@ahsgs.uum.edu.my

**Abstract:** Task resubmission and checkpoint are among several popular techniques used in providing fault tolerance in grid computing. However, due to the lack of side-by-side comparison, it is not certain of the best technique that would not degrade the system performance in addition to providing fault tolerance capability. This study proposed Dynamic ACS-based Fault Tolerance in grid computing using resubmission to new resource, checkpoint technique and utilization of resource execution history with the aim to reduce execution and task processing time and to increase the success rate in grid environment. The proposed algorithm is compared with other relevant algorithms to measure the performance in terms of execution time, success rate and average processing time. The results suggest that the proposed algorithm with improved task resubmission, checkpoint and extended pheromone update formula gives better performance in managing execution failure as well as resource selection during task assignment or resubmission.

**Keywords:** Grid Computing, Fault Tolerance, Task Resubmission, Task Checkpoint, Ant Colony System

## Introduction

Fault tolerance is essential in a distributed system, specifically grid computing, where systems are allowed to recover and continue to operate despite failure occurrence during runtime. Fault tolerance consists of three main strategies, which are fault detection or identification, fault prediction and fault recovery. Fault detection or identification generally is for detecting the type of fault when it occurs before mitigating it with the most suitable solution. On the other hand, fault prediction focuses on predicting the probability of fault to occur based on historical data and applies a suitable scheduling policy to reduce fault probability. Last but not least, fault recovery consists of several popular techniques such as job replication (space-sharing) and check-pointing (time-sharing) (Altameem, 2013). The advantage of job replication is that it does not require re-computation because each job has several simultaneous copies assigned to different resources and if one fails, the others can still be processed. However, this technique is not very effective because a copy of a job is considered as individual execution and may potentially congest the job queue. Checkpoint is another technique that requires the state of running tasks to be stored at a defined checkpoint and if the job fails, the execution will restart from the last saved state instead of from the beginning. This technique ensures that failed tasks will be reprocessed from the last saved state instead of from the beginning and eventually can greatly save the execution time.

Fault tolerance is often integrated with the typical scheduling algorithm. Keerthika and Kasthuri (2011) proposed fault tolerance time to release the scheduling algorithm, which is based on transmission time and fault rate. Time to Release (TTR) is calculated for each resource, while expected deadline is calculated for each job. The job will only be submitted to the resource that has TTR lower than expected deadline. The application checkpoint was proposed by Bawa and Singh (2012) to eliminate the need to restart the job execution from the beginning during failure. The checkpoint information is stored by the central point, or known as the manager and will be retrieved back in the occurrence of failure. There are also several fault tolerance algorithms based on bio-inspired algorithms, specifically Ant Colony Optimization (ACO) as proposed by Wenming *et al*. (2009), Modiri *et al*. (2011), Mandloi and Gupta (2013), Vedulla *et al*. (2013) and Prashar *et al*. (2014). In ACO, pheromone is employed to indicate the fitness of routing path or resource used during path or resource selection process.

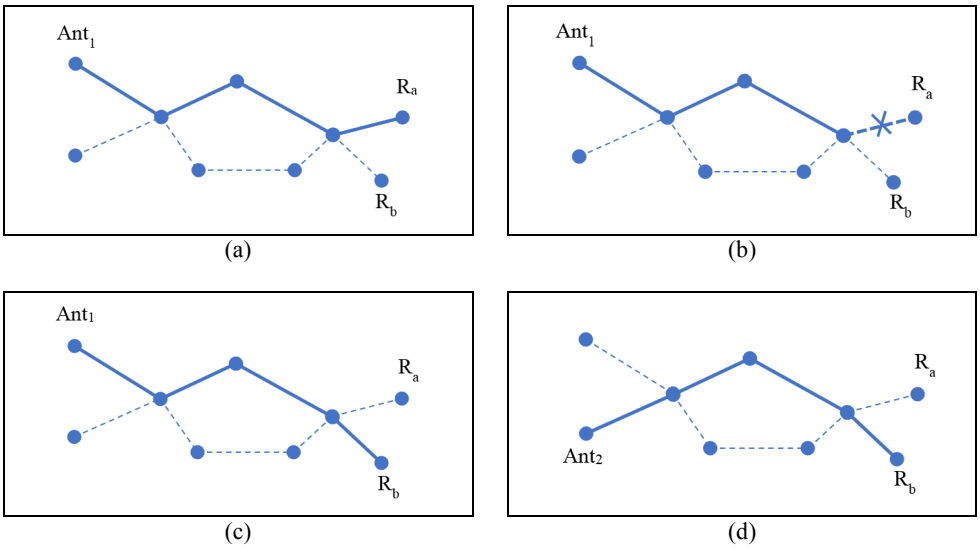


Fig. 1. Illustration of the way ants find alternative resource during failure (a) A path to optimal resource $R_a$ is established by $Ant_1$ and task is submitted (b) Resource $R_a$ fails to complete execution and resubmission flow is invoked (c) A new path to alternative resource $R_b$ is established by $Ant_1$ and task is submitted and processed completely (d) Following $Ant_2$ from different source selects the optimal path to $R_b$ constructed by previous $Ant_1$ to assign the next task

ACO is a biologically-inspired algorithm that provides an adaptive concept for solving optimization problems and designing metaheuristics algorithms (Dorigo and Stützle, 2004; Ferdaus *et al*., 2014). This algorithm is based on the evolutionary approach, where the best solution is searched by a group of ants that work together within the colony. The complete solution is built by combining all the individual solutions of each ant, which are also known as pheromone deposits on the chosen solution or path. The strength of the pheromone is used by other ants as a reference to choose the most optimized path. ACO is very effective when dealing with scheduling and load balancing, but the optimal path finding capability allows an alternative path to be constructed in the presence of fault as illustrated in Figure 1(a-d) respectively.

ACO consists of multiple variants such as Ant System (AS), MAX-MIN Ant System (MMAS) and Ant Colony System (ACS) (Dorigo and Stützle, 2004). The ACO algorithm is suitable to be extended to include the fault tolerance capability because it can be easily adapted to solve both static and dynamic combinatorial optimization problems (Lorpunmanee *et al*., 2007; Ku-Mahamud and Alobaedy, 2012; Goyal and Singh, 2012). ACO is flexible to be modified and combined with other nature-inspired swarm intelligence approaches such as Intelligent Water Drop (IWD) to speed up optimal scheduling, in addition to minimizing make span, balancing the load and utilizing resources efficiently (Mathiyalagan *et al*., 2013). Modiri *et al*. (2011) combined the ACS algorithm with the Directed Acyclic Graph (DAG) method to cater both load balancing and fault tolerance aspects whereby scheduling process adopted DAG method and fault tolerance adopted ACS algorithm.

Most of related works cited in this article focus on improving the task assignment process to reduce the possibility of failure during execution. In addition to that, some related works focus on task resubmission but without checkpoint and with checkpoint but lack of proper validation. Related works in fault tolerance are presented in Section 2, while the proposed algorithm is covered in Section 3. The analysis on experimental results is covered in Section 4 and finally, the conclusion is provided in Section 5.

## Related Work

Enhanced Ant Colony Optimization (EACO) was proposed by Ku-Mahamud and Nasir (2011) to improve the load balancing of grid scheduling by considering the capacity of each resource. Dynamic evaporation rate is introduced to determine the rate of pheromone evaporation based on the number of tasks and resources. This will ensure that whenever the number of tasks to be processed by each resource is high, the evaporation rate will be small, else, the evaporation rate will be high. Results showed that controlling the evaporation rate can effectively balance the load of all resources where the number of tasks assigned to each resource falls within very small margin from the expected number of tasks to be assigned.

Trust-based Ant Colony Optimization (TACO) for grid resource scheduling was proposed by Wenming *et al*. (2009) with the aim to minimize the completion time of jobs, balance all workloads on available resources and at the same time, to introduce the resource-oriented trust mechanism in handling the resource failure problem. Trustworthiness is considered during the resource selection process as it depicts the fitness level of a particular resource. High trust indicates that the resource is highly reliable and will lead to minimum probability of failure. In terms of the fault tolerance aspect, their proposed algorithm used resubmission to other available resources. Since the trust level is based on the execution history, it is possible that the recently failed resource may still be the most preferred resource in the next cycle. However, it would be effective if the checkpoint technique is adopted to eliminate the need to reprocess failed tasks from the beginning and reduce the processing time.

A study by Modiri *et al*. (2011) proposed a new algorithm to manage fault in grid computing by combining the ACS algorithm and Directed Acyclic Graph (DAG) method. By using the DAG method, all tasks are sorted by their dependency. Thus, an offspring task may not begin its work until the parent task is completely executed. All the sorted tasks will go through the resource allocation process using ACS, where ants will try to find the optimal combination of tasks and resource. Once the resource allocation is done, tasks will be executed according to their sorted order. To balance the system load, local pheromone update is applied to route between current task and selected processor while global pheromone update is applied to the optimal route once all solutions are constructed. The main focus of their algorithm is to provide effective resource allocation but without recovery action such as resubmission. The effectiveness of this algorithm can be further improved by integrating recovery actions to ensure all the failed tasks are executed completely in the end of the session.

Hybrid ACO with Genetic Algorithm (GA) was proposed by Mandloi and Gupta (2013) to overcome the uncontrolled nature of metaheuristics of AS, which could degrade the performance of grid allocation. GA is used to choose whether to increase or decrease pheromone update parameters in AS. At the start, ants will randomly select resources to be assigned into subsets. Then, each subset will be evaluated to find the lowest estimated error and will be sorted ascendingly. The best subset will be used to execute tasks in each iteration and the pheromone trail for the chosen subset will be updated. Resources within the best subset will have a high chance to be selected in the subsets of the next iteration. The algorithm can be further upgraded by considering the load balancing aspect and the way to handle job failure when it happens.

Fault Tolerance ACO (FTACO) using checkpoint in grid computing was proposed by Prashar *et al*. (2014) with the aim to solve fault and load balancing problems. The ant-based approach was adopted in the checkpoint mechanism to effectively utilize dynamic resources in grid computing. Ants will move to find an optimal path to process a job and detect the occurrence of failure during job execution. The fault index manager is used to store the list of failures that have occurred in resources during job execution as a reference to the next job iteration. Based on the information in the fault index manager, jobs are re-scheduled to other optimal resources by using the checkpoint mechanism, where the job is restarted from the last saved checkpoint instead of from the beginning. The re-scheduling process considers load balancing on each resource, where a resource with low workload has a high probability to be chosen. The pheromone update mechanism is also used to solve the load balancing problem. Additionally, adding the resource recovery technique may further enhance the functionality of the proposed algorithm. However, due to the lack of experimental results, it is not certain on how good is the proposed algorithm.

Job scheduling with fault tolerance in grid computing using ACO was proposed by Idris *et al*. (2017) to satisfy user's Quality of Services (QoS) which employs resource failure rate and checkpoint-based roll back recovery strategy. During task execution, fault index manager will continuously interact with checkpoint handler to record resource failure rate. When job completion message is received, success index will be incremented. The same case applies whereby failure index will also be incremented when job failure message is received. The resource failure rate stored in fault index manager is used to control the occurrence and interval of checkpoint during scheduling process. In addition to that, for each occurrence of failure, rollback recovery technique will be applied to save the execution time. It is claimed that controlling the occurrence and interval of checkpoint can enhance the performance of fault tolerance system based on checkpoint. The results showed that the proposed algorithm reduces makespan, increases throughput and average turnaround time. Despite the effectiveness of controlling the checkpoint interval to improve QoS, it is also essential to consider load balancing aspect to reduce possibility of stagnation.

ACO is considered as a potential algorithm in grid computing to solve fault problems. Several approaches have been identified to provide fault tolerance such as checkpoint, task resubmission and resource trustworthiness. Despite all these approaches, it is still unclear on how the recently failed resource will not be re-selected for a certain cycle count and how does the checkpoint approach can be effectively implemented using ACO with the structured validation process.

However, task resubmission to other resources and checkpoint can be further combined with the consideration of resource fitness to improve the fault tolerance aspect without disregarding the performance as well as to adapt with the dynamic grid environment.

## Proposed Dynamic ACS-based Fault Tolerance with Checkpoint

Dynamic ACS-based Fault Tolerance (DAFT) is inspired by the concept of ant searching for the optimal path to the most suitable resource to assign tasks. This basic concept is further extended for ants to have the ability to perform resource researching during the resubmission process to ensure that any failed task will be processed completely. In addition to that, the pheromone update technique is further improved as a mechanism to penalize unfit resources so that they become less attractive and eventually reduce the possibility of failure and properly control the task assignment based on resource fitness.

Figure 2 illustrates the high-level workflow of DAFT. For each task, an ant will be generated, which is responsible to perform resource selection based on pheromone values. The initial pheromone value will first be initiated to determine the state of all resources before the first task in queue can be submitted. The selection of resources will be based on the amount of pheromone values either from the initial pheromone calculation or pheromone update process. During the execution, each task will be divided into several checkpoints, which will be processed in sequence to preserve the authenticity of the output. If the execution is successful, the global pheromone update will be applied by the ants to the resource to increase the pheromone after execution. However, in case of any failure during execution, the last checkpoint will be resubmitted to another suitable resource and local pheromone update will be applied. In addition to that, local pheromone update will also be applied in each successful checkpoint. Lastly, the resource will be released with updated pheromone for the next task assignment.

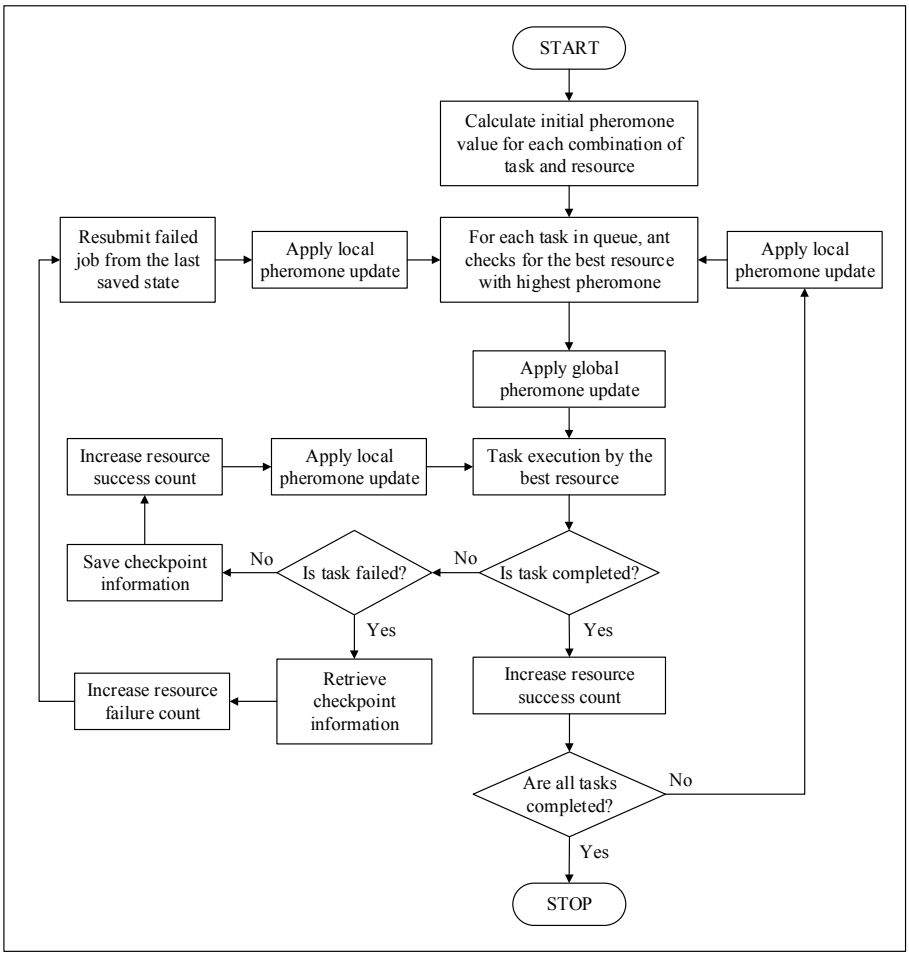


Fig. 2. High-level workflow of DAFT

During the initial task submission, each resource should have pre-defined parameters such as processor speed, current load and bandwidth and number of processing elements. All these parameters will be used to calculate the initial pheromone value, $PV_{ij}$ for each combination of resource $i$ and task $j$. The initial pheromone value formula is given by the formula (1):

$$PV_{ij} = \left[ \frac{S_j}{bandwidth_i} + \frac{C_j}{MIPS_i(1-load_i)} \right] \quad (1)$$

Where:

$S_j$ = The size of a given task $j$
$Bandwidth_i$ = The available bandwidth of resource $i$
$C_j$ = The CPU time needed of task $j$
$MIPS_i$ = The processor speed
$Load_i$ = The current load at resource $i$

Note that the initial pheromone value is assigned during initialization, but after that, it is considered as a resource pheromone value. Since the initial pheromone value is calculated for each combination of task and resource, this information is stored in a PV matrix (2) as follows:

$$PV_{matrix} = \begin{bmatrix} PV_{1,1} & PV_{1,2} & PV_{1,n-1} & PV_{1,n} \\ PV_{2,1} & PV_{2,2} & PV_{2,n-1} & PV_{2,n} \\ PV_{m-1,1} & PV_{m-1,2} & PV_{m-1,n-1} & PV_{m-1,n} \\ PV_{m,1} & PV_{m,2} & PV_{m,n-1} & PV_{m,n} \end{bmatrix} \quad (2)$$

Where:

$n$ = Total number of tasks
$m$ = Total number of resources
$PV_{matrix}$ = A logical form of ant topology whereby an ant would move from one index to another index to find the best resource for task assignment

It is assumed that all the resources are interconnected which means that if the task originates from a specific resource, it can be assigned to all other available resources. Each row in $PV_{matrix}$ presents the list of possible tasks for resource $i$ while each column represents the list of possible resources for task $j$.

The largest pheromone value in each column will be considered by the ant as the most fit resource and the task will be assigned to the resource referenced by the selected index for processing. As soon as the task is assigned, the pheromone value in $PV_{matrix}$ at respective row will be updated by the global pheromone update (3) to decrease the amount of pheromone assigned to the current resource, so that it becomes less attractive by the next ant and leads to the exploration of other resources:

$$\tau_{ij} = (1-\rho).\tau_{ij} + \rho.\Delta\tau_{ij} \quad (3)$$

$\tau_{ij}$ is the amount of pheromone on the resource, while $\Delta_{\tau ij}$ is $1/L_{best}$, where $L_{best}$ denotes the length of global best tour or otherwise (no global best tour found), $\Delta_{\tau ij}$=0. $\rho$ is the evaporation rate that is dynamically controlled by using the following formula (4) with $m$ and $n$ as the total number of resources and tasks respectively:

$$\rho = \left[ \left( \frac{n}{m} \right)^2 \right]^{-1} \quad (4)$$

The task assignment will continue while the previously assigned task is being executed. However, if the execution is not successful, the task will be resubmitted from the last saved checkpoint to another suitable resource. On the other hand, the checkpoint information will be recorded during the execution for each task being executed and this information is also used to update the execution history t able for each resource.

In every checkpoint, another round of local pheromone update (5) will be applied to reduce more pheromone value by considering the execution history to influence the reduction of pheromone; the success status would slightly reduce pheromone, while the failure status would reduce more pheromone:

$$\tau_{ij} = (1-\rho).\tau_{ij} + \rho.\tau_0(E_i) \quad (5)$$

$\tau_0$ is the initial pheromone value of resource $i$, while $E_i$ is the execution history of resource $i$ and calculated by dividing the number of successful checkpoints with the total number of checkpoint calls (failure + successful) at resource $i$.

1. Initialization
   - 1.1 All the parameters are set
   - 1.1 Calculate the initial pheromone value for each resource
   - 1.2 Spawn an individual ant for each task
   - 1.3 Identify the resource with the highest initial pheromone value for the first iteration
2. Main loop begins
   - 2.1 The ant finds the best the resource with the highest pheromone value
   - 2.2 Once found, the ant gives a signal for task submission and applies formula (3)
   - 2.3 Task submission and processing begins
   - 2.4 While task assignment status = false
     - 2.4.1 Repeat step 2.1
   - 2.5 While task execution status = false
     - 2.5.1 If task execution does not fail, save checkpoint, increase success count and apply formula (5)
     - 2.5.2 Else if task execution failed, retrieve the last checkpoint, increase failure count, apply formula (5) and repeat step 2.1
     - 2.5.3 Else if task execution status = true, increase success count and apply formula (5)
3. Terminate the execution when both task assignment status = true and task execution status = true

Fig. 3. High-level pseudocode of DAFT

The execution history $E_i$ (or defined as resource fitness) is extended to existing local pheromone update formula and will be used by ants to control the amount of pheromone reduction; the better the execution history is, the lower the pheromone is reduced. In contrast, low execution history leads to higher reduction of pheromone in each resource.

The high-level pseudocode of DAFT is further illustrated in Figure 3 in which every step that requires computation is properly mapped to respective formula.

## Experimental Results

To validate the performance of the proposed DAFT algorithm, the mean success rate is defined at 70% (0.7) with the error range represented by a standard deviation of ± 0% (0.0) up to ± 30% (0.3). The success rate is assigned using the pseudorandom algorithm with a standard deviation to define the range to each individual resource during the initialization process. Each resource has different success rate and this information is not known by the ants during the resource assignment. To ensure that the experiment on failure is reliable, each individual resource is set to have the same processing capability (except for success rate) as shown in Table 1. Grid resource characteristics such as PE rating, bandwidth, and number of machine per resource are defined based on parameters used by Idris *et al*. (2017). DAFT algorithm is compared with TACO proposed by Wenming *et al*. (2009) and fault tolerance FTACO proposed by Prashar *et al*. (2014) which are reimplemented in GridSim Toolkit based on published flowchart, pseudocode and formulation. Each scenario is executed 10 times where the average is taken for a more accurate measurement.

Figure 4 shows that the execution time slightly decreases when the error range is increased for both checkpoint scenarios (FTACO and DAFT), while it greatly decreases for scenario without checkpoint (TACO). Both FTACO and DAFT have relatively equivalent performance. However, as the range increases, the heuristic capability of the ant algorithm can make better decisions on resource selection during resubmission, whereby it will be influenced to assign to the resource with the highest pheromone value and ultimately shortens the execution time. This is aligned with the actual distributed scenario that has many non-homogeneous resources with different fitness.

Figure 5 shows that the success rate increases in parallel with the standard deviation, where DAFT has the highest success rate as compared to TACO and FTACO. Again, this supports the fact that better decision making during the resource assignment with consideration of execution history can reduce the possibility of failure. Furthermore, with the application of checkpoint technique, the need for the failed task to be restarted from the beginning is eliminated and eventually reduces the exposure of task execution to possible failure.

Table 1. Simulation parameters

| Parameter | Value |
|---|---|
| Number of tasks | 10000 |
| Number of resources | 100 |
| PE rating | 50 MIPS |
| Bandwidth | 5000 B/S |
| Number of machine per resource | 1 |
| PE per machine | 5 |
| Mean success rate | 0.7 |
| Range of resource success rate (Interval: 0.05) | 0-0.3 |

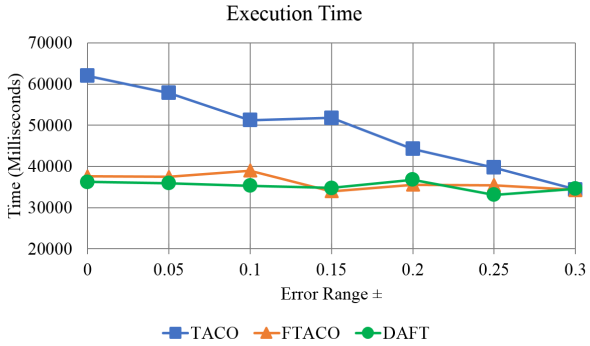


Fig. 4. Results of execution time

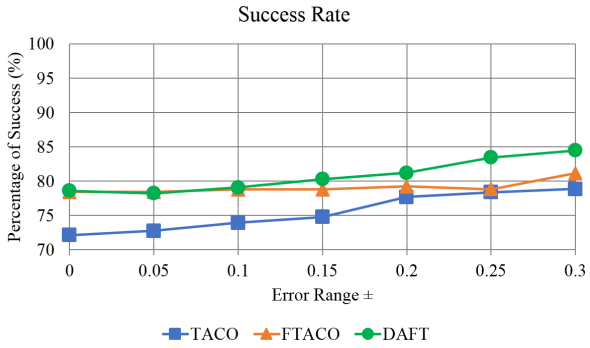


Fig. 5. Results of success rate

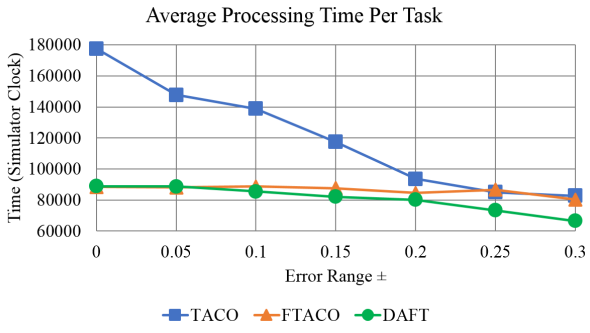


Fig. 6. Results of average processing time per task

Figure 6 shows that the average processing time per individual task decreases along with error range increment. DAFT has the best performance at a standard deviation of 0.3 (each resource having a possibility of success from 40% up to 100%). Another significant output from the results is that the heuristic capability of the ant algorithm during the resource assignment can select the best resource according to its fitness regardless of whether the checkpoint is applied or not. This fact ultimately reduces the processing time for each task.

In real grid environment, each available resource will have different fitness in addition to processing capability. In this case, the minimum and maximum fitness value can be used to form fitness range. The results suggest that the wider the range is, the better heuristic capability can improve task assignment process and eventually improve the performance in grid environment with high probability failure. Consideration on the resource execution history is proven to be effective as ants can make better decisions in selecting the most fit resource. In other words, as the execution goes on, the success and failure counts will be recorded and will eventually affect the evaporation of resource pheromone values and dynamically distribute the task according to resource fitness. For example, if the resource has 0% success rate (100% failure rate), it will have the least number of tasks assigned to it. On the other hand, if the resource has a very high success rate, it will be assigned with the most number of tasks.

In addition to consideration of resource fitness during scheduling or resubmission process, the checkpoint allows the failed task to be resubmitted from the last saved state. This greatly reduces the processing time as the task does not need to be restarted from the beginning. It is highly recommended that the checkpoint information is stored by independent components such as checkpoint manager placed outside of task or resource. Storing the checkpoint information at task or resource level is not reliable as failure may corrupt the checkpoint information.

## Conclusion

It can be concluded that DAFT gives better average execution time per task and success rate compared to TACO and FTACO. However, since the heuristic capability of the ant algorithm relies on historical record, it is still possible that a recently failed resource will be assigned with tasks right after failure (should the resource have high pheromone values) and this may still lead to a higher possibility of failure. To prevent this from happening, there should be a mechanism to suspend recently failed resources temporarily so that they will not be selected as soon as they fail. Thus, future work can focus on the application of temporary resource suspension should the resource fail to complete task execution in addition to checkpoint-based resubmission technique. This can potentially be extended in the resource selection algorithm, but the suitability of the amount of suspension should be further explored so that sufficient amount of suspension can be applied.

## Acknowledgement

The authors wish to thank the Ministry of Higher Education Malaysia in funding this study under the Trans Disciplinary Research Grant Scheme (TRGS), S/O code 13164 and RIMC, Universiti Utara Malaysia, Kedah for the administration of this study.

## Author's Contributions

All authors are equally contributed in this work and the article.

## 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

Altameem, T., 2013. Fault tolerance techniques in grid computing systems. Int. J. Comp. Sci. Inform. Technol., 4: 858-862.

Bawa, R.K. and R. Singh, 2012. Comparative analysis of fault tolerance techniques in grid environment. Int. J. Comput. Appli., 41: 21-25. DOI: 10.5120/5505-7520

Dorigo, M. and T. Stützle, 2004. Ant colony optimization. MIT Press, Cambridge.

Ferdaus, M.H., M. Murshed, R.N. Calheiros and R. Buyya, 2014. Virtual Machine Consolidation in Cloud Data Centers using ACO Metaheuristic. In: Euro-Par 2014 Parallel Processing, F. Silva, I. Dutra and V.S. Costa (Eds.), Springer International Publishing, Porto, pp: 306-317.

Goyal, S.K. and M. Singh, 2012. Adaptive and dynamic load balancing in grid using ant colony optimization. Int. J. Eng. Technol., 4: 167-174.

Idris, H., A.E. Ezugwu, S.B. Junaidu and A.O. Adewumi, 2017. An improved ant colony optimization algorithm with fault tolerance for job scheduling in grid computing systems. PLOS ONE, 12: 1-24. DOI: 10.1371/journal.pone.0177567

Keerthika, P. and N. Kasthuri, 2011. A new proactive fault tolerant approach for scheduling in computational grid. Proceedings of the International Conference on Web Services Computing (ICWSC'11), IJCA, pp: 55-59.

Ku-Mahamud, K.R. and H.J.A. Nasir, 2011. Enhancement of ant colony optimization for grid load balancing. Eur. J. Sci. Res., 61: 42-50.

Ku-Mahamud, K.R. and M.M. Alobaedy, 2012. New heuristic function in ant colony system for job scheduling in grid computing. In: Mathematical Methods for Information Science and Economics, Mastorakis, N., E. Zaitseva, D. Randjelovic, K.K.F. Yuen and C.G. Carstea *et al*. (Eds.), WSEAS, Montreux, pp: 47-52.

Lorpunmanee, S., M.N. Sap, A.H. Abdullah and C. Chompoo-inwai. 2007. An ant colony optimization for dynamic job scheduling in grid environment. World Academy Sci. Eng. Technol., 29: 314-321.

Mandloi, S. and H. Gupta, 2013. Adaptive job scheduling for computational grid based on ant colony optimization with genetic parameter selection. Int. J. Advanced Comp. Res., 3: 66-71.

Mathiyalagan, P., S.N. Sivanandam and K.S. Saranya, 2013. Hybridization of modified ant colony optimization and intelligent water drops algorithm for job scheduling in computational grid. ICTACT J. Soft Comput., 4: 651-655.
DOI: 10.21917/ijsc.2013.0093

Modiri, V., M. Analoui and S. Jabbehdari, 2011. Fault tolerance in grid using ant colony optimization and directed acyclic graph. Int. J. Grid Comp. Appli., 2: 14-26. DOI: 10.5121/ijgca.2011.2102

Prashar, T., Nancy and D. Kumar, 2014. Fault tolerant ACO using checkpoint in grid computing. Int. J. Comp. Appli., 98: 44-49. DOI: 10.5120/17223-7465

Vedulla, A., A. P. Reddy and B.K. Prasad, 2013. Fault recovery in optimal task scheduling and grid service reliability. Int. J. Comput. Sci. Inform. Technol., 4: 147-151.

Wenming, H., D. Zhenrong and W. Peizhi, 2009. Trust-based ant colony optimization for grid resource scheduling. Proceeding of the 3rd International Conference on Genetic and Evolutionary Computing, Oct. 14-17, IEEE, pp: 88-292.
DOI: 10.1109/WGEC.2009.180