Reliability of A Distributed Computing System Considering Task Abort

Rui Peng
School of Economics & Management
Beijing University of Technology
Beijing, China
pengruisubmit@163.com

Kaiye Gao*
School of Economics & Management
Beijing Information Science & Technology University
Beijing, China
kygao@foxmail.com

Di Wu
School of Management
Xi'an Jiaotong University
Xi'an, China
wd_0824@stu.xjtu.edu.cn

Qingqing Zhai
School of Management
Shanghai University
Shanghai, China
zhaiqing59@163.com

Abstract—**This paper studies the reliability of a distributed computing system where the task on a computer may be aborted if it is still not finished until a time threshold. The mission is regarded as successful if all these tasks are completed by at least one computer before a pre-specified time. An abort policy model is proposed and a multi-valued decision diagram approach is adopted to evaluate the system reliability for any given abort policy. A numerical example is provided to illustrate the applications.**

Keywords-distributed computing systems; abort policy; multi-valued decision diagram; reliability

I. INTRODUCTION

Some recent researches have studied the abort policy for different types of systems, motivated by the fact that sometimes the system survival should be given a priority compared with mission completion [1-4]. Levitin et al. studied the influence of failure propagation on mission abort policy in heterogeneous warm standby systems [1]. Cha et al. proposed an approach to obtain optimal mission abort policy for partially repairable heterogeneous systems [2]. Levitin et al. discussed how to balance the uncompleted mission penalty and system loss risk with the mission abort policy [3]. Peng applied the abort policy for cooperative unmanned aerial vehicles [4].

However, none of these works have considered the application of aborting in the context of distributed computing systems, which are the origin of cloud computing technique and are paramount for the processing of big data which cannot be solved on centralized equipment [5,6].

Thus, in this paper, a model for task abort policy in a distributed computing system is proposed. The remaining of this paper is organized as follows. Section 2 describes the system. Section 3 presents a specific example and evaluates its reliability with multivalued decision diagram (MDD). Section 4 concludes and points out some future works.

National Nature Science Foundation of China (71671016)
School Found of Beijing Information Science and Technology University (1935004)

II. SYSTEM DESCRIPTION

Consider a distributed computing system consisting of N computers. There are totally M tasks to be completed before time T. The M tasks are allocated to the N computers such that each task is assigned to at least one computer. It is assumed that all the tasks assigned to a computer must be executed sequentially and the same task cannot be executed twice on the same computer. Assume $h_i(k) = j$ represent that the k-th task to be executed for computer i is task j where $h_i(k) = 1,...,M$. Then we can obtain a set of tasks assigned for computer i, i.e, $H_i = \{h_i(k) \mid k = 1,...,|H_i|\}$ where $|H_i|$ is the number of tasks assigned to computer i and $h_i(1) \neq h_i(2) \neq ... \neq h_i(|H_i|)$.

It is assumed that the time needed to execute each task on each computer is random, and the progress of each task is unobservable before it is finished. As the focus of this paper is on the mission abort policy, for simplicity, the time needed to execute $h_i(j)$ is assumed to be exponential distributed with probability density function (pdf) denoted as $\lambda_{i,h_i(j)} e^{-\lambda_{i,h_i(j)} t}$.

In order to design the task abort policy, a few concerns need to be discussed. First, the tasks that are only assigned to one computer cannot be aborted, because aborting a task without redundancy simply makes the mission fail. Second, given that a task is executed by multiple computers, whether to abort a task on a computer it depends on not only the time spent by this task but also the time spent by the previous tasks on this computer. The user is less likely to abort the task on a specific computer if the previous tasks on this computer consume less time. Third, a task that is originally assigned to multiple computers may gradually run out of redundancy with the abort of the task on these computers. Lastly, a task does not need to be executed any more if the same task is already finished on other computers.

With these consideration, the aborting policy is designed as follows. A task assigned to multiple computers is defined as redundant task. For tasks executed on computer *i*, the time thresholds for the aborting of the tasks is $Q_i = \{t_{i,1}, t_{i,2}, \cdots, t_{i,|H_i|}\}$, where $t_{i,1} \le t_{i,2} \le \cdots \le t_{i,|H_i|} = T$ and the *j*-th task should be aborted if it is still not finished at $t_{i,j}$. Specifically, it should be noted that $t_{i,j} = +\infty$ if $h_i(j)$ is not a redundant task and $t_{i,|H_i|} = T$ as the whole mission should be completed at *T*. The thresholds are used to ensure that the whole process of mission execution is not influenced too much by a single task. A task is not allowed to abort if this task is already aborted on all the other computers assigned this task. In addition, whenever a task is completed on a computer, the same task is immediately aborted on all the other computers no matter the aborting time threshold has reached or not. Under this abort policy, the mission reliability is defined as the probability that all the *M* tasks are completed by at least one computer before time *T*. This paper studies the mission reliability for fixed aborting policy $Q_i, \ldots, Q_N$, which will be the basis for the study of optimal aborting policy in future.

III. ILLUSTRATIVE EXAMPLE

Consider an example where three tasks are distributed to two computers. The tasks assigned to the first computer and the second computer are denoted as $H_1 = \{3,1,2\}$ and $H_2 = \{2,1\}$ respectively. Because task 1 and task 2 are assigned to both computers, the abort of a task may be dependent on the abort and the completion of the same task on other computers. In order to handle such dependency, the MDD technique is adopted. The MDD technique has been adopted in many studies for evaluating the reliability of multistate systems with time dependency, such as warm standby systems and phased mission systems [7-10].

To apply the MDD in this study, the top node of the MDD is designed to contain all the tasks assigned to all the computers. The branches of the top node represent the first event that happens to the system, which can be the completion of any task on any computer, the abort of any task on any computer, or "nothing happens". For any branch representing completion of a task, check whether the path leading to it contains the completion of all tasks, which represents mission success and indicates that this path needs no more branching if it is true. Similarly, for any branch representing "nothing happens", it means that the mission fails and the path also needs no more branching. For any branch representing task abort, check whether there is any task after it. In the case of "Yes", continue to branch, otherwise it indicates mission failure and that the path leading to this branch needs no more branching. At the end of each branch locates a second level node which represents all the tasks except for the tasks that are already completed or aborted. Similarly, the branches for each second level node represent the second event that happens to the system, which can be the completion of any remaining task on any computer, the abort of any remaining task on any computer, or "nothing happens". The branching is continued level by level until all the paths represent either the success of the mission or the failure of the mission. The mission reliability can be obtained by adding up the probabilities of all the paths leading to system success.

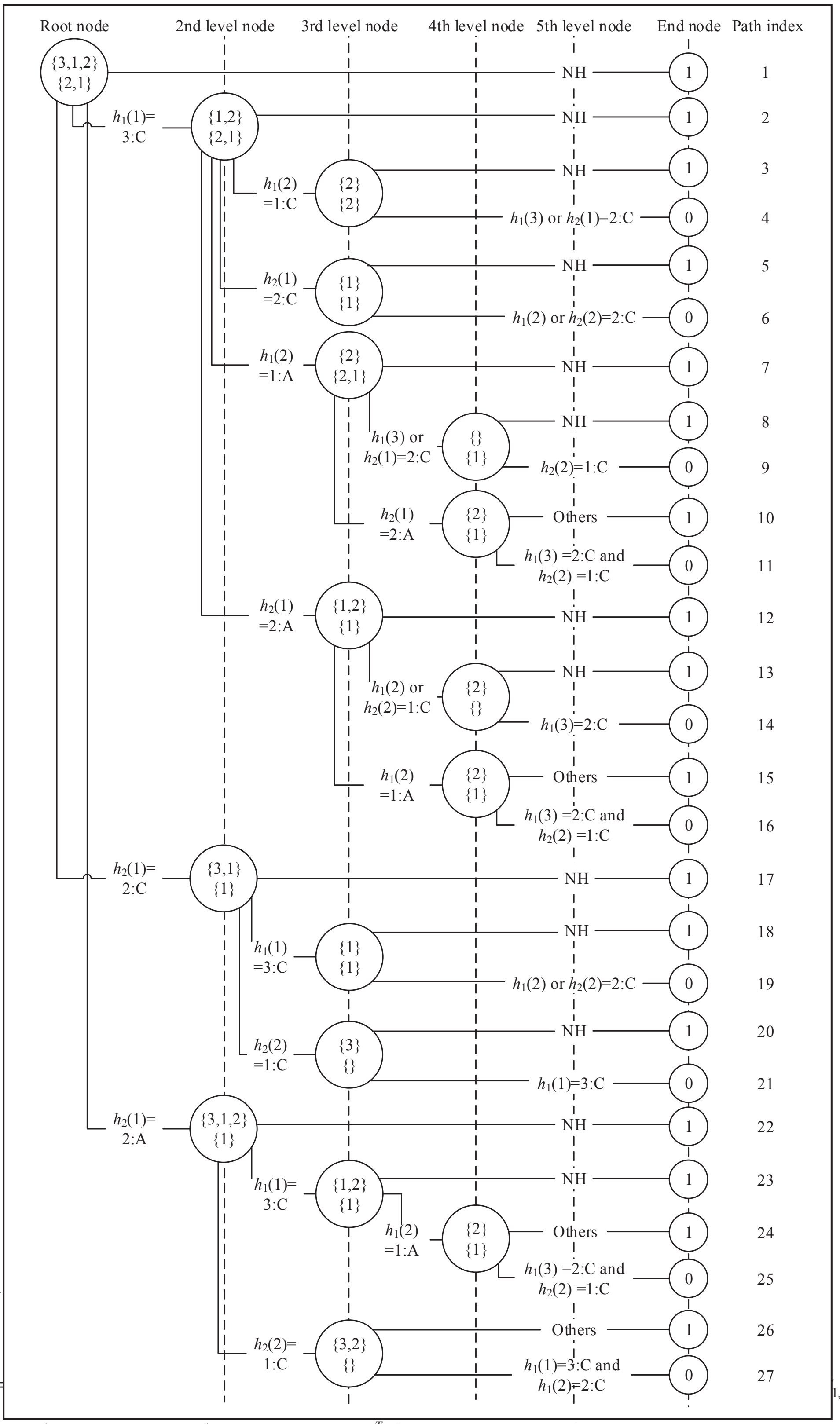
The MDD for the example system is as shown in Figure 1. In the figure, $h_i(j)=k$: *C* and $h_i(j)=k$: *A* denote the "Completion" and the "Abortion" of the task $h_i(j)=k$, and "NH" indicates that "Nothing Happens" before *T*. For simplicity, "Others" indicates all the possible situations except the one specified by the other branch. In the terminal nodes, "1" indicates there are still incomplete tasks and "0" indicates there is no task left. In addition, all the paths are orderly indexed, which is shown on the right of the terminal nodes.

In this example, we assume that the exponential parameters of executing time are $\{\lambda_{1,h_1(1)}, \lambda_{1,h_1(2)}, \lambda_{1,h_1(3)}\}$ and $\{\lambda_{2,h_2(1)}, \lambda_{2,h_2(2)}\}$ for computer 1 and computer 2 respectively. As there are only 2 tasks with redundancy in both two computers and the last task of each computer should be aborted only before *T*, we set that $Q_1 = \{-, t_{1,2}, T\}$, $Q_2 = \{t_{2,1}, T\}$ and $t_{1,2} < T \,\&\, t_{2,1} < T$. Without loss of generality, it is assumed that $t_{1,2} < t_{2,1}$. We use $P(kp)$ to denote the probability of the path *kp*. Then, we can obtain the mission reliability by adding up all the probabilities of the paths leading to system success, that is, those paths with end nodes of 0.

For example, the probability of path 4 is

$$\begin{aligned} P(4) = {} & \int_0^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_0^{t_{1,2}-\tau_{1,1}} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} \int_0^{T-\tau_{1,2}} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} \int_{\tau_{1,2}+\tau_{1,3}}^{+\infty} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} d\tau_{2,1} d\tau_{1,3} d\tau_{1,2} d\tau_{1,1} \\ & + \int_0^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_0^{t_{1,2}-\tau_{1,1}} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} \int_{\tau_{1,2}}^{T} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} d\tau_{2,1} \int_{\tau_{2,1}}^{+\infty} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} d\tau_{1,3} d\tau_{2,1} d\tau_{1,2} d\tau_{1,1}, \end{aligned} \tag{1}$$

where the first term indicates the situation that $h_1(1)$, $h_1(2)$ and $h_1(3)$ are completed one by one by computer 1 within *T*; and the second item indicates the situation that $h_1(1)$ and $h_1(2)$ are completed by computer 1 and then $h_2(1)$ is completed by computer 2 within *T*.



Similarly, we can

$P(6) =$ ${}_{1,2}d\tau_{2,1}d\tau_{1,1}$ (2)

$$+\int_0^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{\tau_{1,1}}^{t_{1,2}} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_0^{T-\tau_{2,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} \int_{\tau_{2,1}+\tau_{2,2}}^{+\infty} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} d\tau_{1,2}d\tau_{2,2}d\tau_{2,1}d\tau_{1,1},$$

$$
\begin{aligned}
P(9) = & \int_0^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{t_{1,2}-\tau_{1,1}}^{+\infty} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} \int_0^{t_{2,1}-t_{1,2}} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} \int_{t_{1,2}+\tau_{1,3}}^{+\infty} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \\
& \times \int_0^{T-t_{1,2}-\tau_{1,3}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} d\tau_{2,2} d\tau_{2,1} d\tau_{1,3} d\tau_{1,2} d\tau_{1,1} \\
& + \int_0^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{t_{1,2}-\tau_{1,1}}^{+\infty} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} \int_{t_{1,2}}^{t_{2,1}} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \\
& \times \int_{\tau_{2,1}-t_{1,2}}^{+\infty} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} \int_0^{T-\tau_{2,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} d\tau_{2,2} d\tau_{1,3} d\tau_{2,1} d\tau_{1,2} d\tau_{1,1},
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
P(11) = & \int_0^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{t_{1,2}-\tau_{1,1}}^{+\infty} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} \int_{t_{2,1}}^{+\infty} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_{\tau_{2,1}-t_{1,2}}^{T-t_{1,2}} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} \\
& \times \int_0^{T-t_{2,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} d\tau_{2,2} d\tau_{1,3} d\tau_{2,1} d\tau_{1,2} d\tau_{1,1},
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
P(19) = & \int_0^{t_{1,2}} \lambda_{2,h_2(1)} e^{-\lambda_{2,1}\tau_{2,1}} \int_{\tau_{2,1}}^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_0^{t_{1,2}-\tau_{1,1}} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} \int_{\tau_{1,1}+\tau_{1,2}-\tau_{2,1}}^{+\infty} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} d\tau_{2,2} d\tau_{1,2} d\tau_{1,1} d\tau_{2,1} \\
& + \int_0^{t_{1,2}} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_{\tau_{2,1}}^{t_{1,2}} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{\tau_{1,1}-\tau_{2,1}}^{t_{1,2}-\tau_{1,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} \int_{t_{2,1}+t_{2,2}-\tau_{1,1}}^{+\infty} \lambda_{1,h_1(2)} e^{-\lambda_{1,h_1(2)}\tau_{1,2}} d\tau_{1,2} d\tau_{2,2} d\tau_{1,1} d\tau_{2,1} \\
& + \int_0^{t_{1,2}} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_{t_{1,2}}^{T} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{\tau_{1,1}-\tau_{2,1}}^{T-\tau_{1,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} d\tau_{2,2} d\tau_{1,1} d\tau_{2,1} \\
& + \int_{t_{1,2}}^{t_{2,1}} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_{\tau_{2,1}}^{T} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{\tau_{1,1}-\tau_{2,1}}^{T-\tau_{1,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} d\tau_{2,2} d\tau_{1,1} d\tau_{2,1},
\end{aligned} \tag{5}
$$

$$
P(21) = \int_0^{t_{2,1}} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_0^{T-\tau_{2,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} \int_{\tau_{2,1}+\tau_{2,2}}^{T} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} d\tau_{1,1} d\tau_{2,2} d\tau_{2,1}, \tag{6}
$$

$$
P(25) = \int_{t_{2,1}}^{T} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_{t_{2,1}}^{T} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_{\tau_{1,1}-t_{2,1}}^{T-t_{2,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} \int_0^{T-\tau_{1,1}} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} d\tau_{1,3} d\tau_{2,2} d\tau_{1,1} d\tau_{2,1}, \tag{7}
$$

$$
P(27) = \int_{t_{2,1}}^{+\infty} \lambda_{2,h_2(1)} e^{-\lambda_{2,h_2(1)}\tau_{2,1}} \int_0^{T-t_{2,1}} \lambda_{2,h_2(2)} e^{-\lambda_{2,h_2(2)}\tau_{2,2}} \int_{t_{2,1}+\tau_{2,2}}^{T} \lambda_{1,h_1(1)} e^{-\lambda_{1,h_1(1)}\tau_{1,1}} \int_0^{T-\tau_{1,1}} \lambda_{1,h_1(3)} e^{-\lambda_{1,h_1(3)}\tau_{1,3}} d\tau_{1,3} d\tau_{1,1} d\tau_{2,2} d\tau_{2,1}, \tag{8}
$$

Besides, the probabilities of path 14 and 16, $P(14)$ and $P(16)$, are zero. Actually, since $t_{2,1} > t_{1,2}$, the situations represented by paths 14 and 16 cannot happen.

Finally, by adding up these probabilities, the mission reliability is

$$
\begin{aligned}
P = & P(4) + P(6) + P(9) + P(11) \\
& + P(14) + P(16) + P(19) \\
& + P(21) + P(25) + P(27)
\end{aligned} . \tag{9}
$$

Then we give the specific values of parameters to illustrate the feasibility of the proposed model. To facilitate the calculation, we assume that the execution time of every assigned task is unified to follow a standard exponential distribution e^{-t}. The time thresholds of tasks aborting are set as $Q_1 = \{-, 4, 7\}$ and $Q_2 = \{6, 7\}$. Then according to the equations listed above, we can calculate all the probabilities of the paths leading to system success, which are shown in Table 1. Finally, we have the mission reliability according to equation (9), which is 0.988933.

TABLE I. PROBABILITIES OF THE PATHS LEADING TO SYSTEM SUCCESS

Path Index	Probability
4	0.407497
6	0.281100
9	6.115801×10^{-4}
11	9.078672×10^{-7}

Path Index	**Probability**
19	0.123162
21	0.249548
25	1.968020×10^{-6}
27	2.301930×10^{-6}

IV. CONCLUSIONS

In this paper, we proposed an abort policy model to distributed computing systems and evaluate the system reliability based on the multi-valued decision diagram approach. This model considers the case where the task may be aborted on a computer if it is still not finished until a time threshold. The mission success is defined as the completion of all the tasks by at least one computer before a pre-specified time. A specific numerical example is presented to illustrate the model and its applications.

REFERENCES

[1] G. Levitin, L. D. Xing, and L. Luo, "Influence of failure propagation on mission abort policy in heterogeneous warm standby systems," Reliability Engineering & System Safety, vol. 183, pp. 29-38, 2019.

[2] J. H. Cha, M. Finkelstein, and G. Levitin, "Optimal mission abort policy for partially repairable heterogeneous systems," European Journal of Operational Research, vol. 271, no. 3, pp. 818-825, 2018.

[3] G. Levitin, M. Finkelstein, and Y. S. Dai, "Mission abort policy balancing the uncompleted mission penalty and system loss risk," Reliability Engineering & System Safety, vol. 176, pp. 194-201, 2018.

[4] R. Peng, "Joint routing and aborting optimization of cooperative unmanned aerial vehicles," Reliability Engineering & System Safety vol. 177, pp. 131-137, 2018.

[5] S. Bagchi, "Self-adaptive and reconfigurable distributed computing systems," Applied Soft Computing, vol. 12, no. 9, pp. 3023-3033, 2012.

[6] I. Grasso, S. Pellegrini, B. Cosenza, and T. Fahringer, "A uniform approach for programming distributed heterogeneous computing systems," Journal of Parallel and Distributed Computing, vol. 74, no. 12, pp. 3228-3239, 2014.

[7] R. Peng, Q. Q. Zhai, L. D. Xing, and J. Yang, "Reliability of demand-based phased-mission systems subject to fault level coverage," Reliability Engineering and System Safety, vol. 121, pp. 18-25, 2014.

[8] Zhai, Q., Xing, L., Peng, R., Yang, J., 2018. Aggregated Combinatorial Reliability Model for Non-Repairable Parallel Phased-Mission Systems. Reliability Engineering and System Safety 176, 242-250.

[9] Y. C. Mo, Y. Liu, and L. Cui, "Performability analysis of multi-state series-parallel systems with heterogeneous components," Reliability Engineering & System Safety, vol. 171, pp. 48-56, 2018.

[10] Y. C. Mo, L. Xing, L. Cui, and S. Si, "MDD-based performability analysis of multi-state linear consecutive-k-out-of-n: F systems," Reliability Engineering & System Safety, vol. 166, pp. 124-131, 2017.