Reliability Engineering and System Safety 169 (2018) 63–75

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Determining the inspection intervals for one-shot systems with support equipment

Qian Qian Zhao, Won Young Yun*

Department of Industrial Engineering, Pusan National University, 30 Jangjeon-dong, Geumjeong-gu, Busan, 609-735, South Korea

ARTICLE INFO

Keywords:
One-shot device
Support equipment
Interval availability
Estimation of distribution algorithm

ABSTRACT

This paper considers systems that comprise one-shot devices and support equipment. One-shot devices are stored for long periods of time, and failures are detected only upon inspection. The support equipment needed to operate one-shot devices is maintained immediately upon failure. This paper addresses the inspection schedule problem for such systems with limited maintenance resources. The interval availability and life cycle cost are used as optimization criteria. The aim is to determine near-optimal inspection intervals for one-shot systems to minimize the expected life cycle cost and satisfy the target interval availability between inspection periods. An estimation of distribution algorithm (EDA) and a heuristic method are proposed to find the near-optimal solutions, and numerical examples are given to demonstrate the effects of the various model parameters to the near-optimal inspection intervals.

1. Introduction

One-shot systems such as the man-portable air-defense system (MANPADS) are complex and involve one-shot devices and support equipment. The one-shot devices in MANPADS are missiles, which are kept in storage for long periods of time. The support equipment is the launchers, which are including a battery coolant unit (BCU), grip stock, and launch tube. This equipment does not need a specific environment for storage, and their failures can be known immediately without inspection. Furthermore, one launcher can be used with several missiles.

One-shot devices carry out their functions only once at most during their life span, and usually need high reliability to ensure successful operation. The reliability of one-shot devices deteriorates over time during storage, and the exact failure times of a system cannot be predicted accurately in most instances. The conditions of such systems may only be determined upon operation. Thus, inspections are carried out periodically to detect system failures and maintain high system reliability. However, it is difficult to determine suitable inspection intervals for most one-shot systems due to the trade-off between inspection frequency and maintenance costs. More frequent inspections reduce the mean down time of systems or the time between failure and detection, but they incur higher maintenance costs. Additionally, the process of testing may degrade specific units of the system. Appropriate inspection intervals for one-shot systems are therefore needed.

Many researchers have proposed various inspection policies for such systems. Nakagawa and Mizutani [1] reviewed three inspection models over a finite time span: periodic inspection, sequential inspection, and asymptotic inspection. Nakagawa et al. [2] proposed periodic and sequential inspection policies that involve inspecting a system periodically. They determined the optimal inspection numbers to minimize the expected total cost. Hariga [3] developed a mathematical model for a single-unit system to determine inspection intervals that would maximize the expected profit per unit time. Chelbi and Ait-Kadi [4] also developed a mathematical model to obtain optimal inspection intervals for a system. They assumed that the system would be replaced with a new one if the inspection revealed a failure. Alternatively, a preventive replacement would be scheduled if the system does not fail but the measured values of the control parameter exceed predetermined threshold levels. The optimal inspection intervals were determined to minimize the expected total cost per unit time over an infinite time span.

Huynh et al. [5] considered periodic inspection/replacement ($P-I/R$) and block replacement ($B-R$) policies for a single-unit system. Under $P-I/R$ policy, the system is inspected with period T, and is restored to as-good-as new after repair. When degradation reaches a preventive maintenance threshold M, the system is replaced. They determined optimal inspection interval and a preventive maintenance threshold that minimize the expected maintenance cost per unit over an infinite time span. The system is always replaced at interval T, and corrective replacement cost is considered higher than preventive replacement cost in the $(B-R)$ policy, they also found the optimal values of regular time interval T which minimized the cost criterion. Van der Weide and Pandey [6] presented a stochastic alternating renewal process model for a single-unit system. Failures are detected only by periodic inspection, the system is renewed at each inspection time point. It can also be renewed by preventive maintenance (PM) once it reaches a

* Corresponding author.
E-mail addresses: zhaoyangxue@gmail.com (Q.Q. Zhao), wonyun@pusan.ac.kr (W.Y. Yun).

http://dx.doi.org/10.1016/j.ress.2017.08.007
Received 29 November 2016; Received in revised form 19 July 2017; Accepted 3 August 2017
Available online 5 August 2017
0951-8320/

predetermined age. Renewal function, point unavailability and time average unavailability, and the effect of age based on *PM* policy were evaluated. They concluded that the point unavailability can be reduced by *PM*. Cui et al. [7] obtained the instantaneous availability and the limiting average availability under periodic inspections for a single-unit storage system. The instantaneous availability is obtained by using the virtual age concept, and they assumed that the virtual age during the failure time is the same as at the moment before the system fails. Inspection is assumed to be perfect, and if failure is detected, there are two possible maintenance actions: minimal repair for regular failures and perfect repair at *N*th inspection time point.

Ito and Nakagawa [8] considered optimal inspection policies for a system with two units in storage, one of which is maintained upon inspection and the other degrading over time. To maintain a higher degree of system reliability, the system is inspected and maintained periodically, and it is overhauled if the reliability becomes less than or equal to a specific value. The optimal inspection times were determined to minimize the average cost, including inspection and overhaul costs. In a later study, Ito and Nakagawa [9] assumed that the system would be replaced upon the detection of failure or when its reliability decreases beyond a specific value. Ito and Nakagawa [10] also considered a system that contains a component that degrades over time, and they determined the optimal inspection intervals that minimize the expected total cost, including inspection and loss costs. They later determined the optimal inspection times that would minimize the mean down time and average cost until overhaul [11]. They also considered three types of units: Unit 1 is inspected and maintained at time interval *T*, Unit 2 is partially replaced at time interval *NT*, and Unit 3 is only overhauled if the reliability is less than or equal to a specific value. The optimal inspection and replacement times were determined to minimize the expected total cost until overhaul [12].

Badia et al. [13] proposed an inspection policy for a single-unit system that is renewed upon the observation of failure through periodic inspection. They assumed that the inspection might not be perfect, and the optimal inspection intervals were determined to minimize the average cost per unit of time over an infinite time span. Wolde and Ghobbar [14] considered reliability, availability, and cost as optimization criteria. Availability and reliability were used to evaluate the system performance. They showed how these optimization criteria are related to each other and that improved reliability can impact availability. They suggested a mathematical model to determine the optimal inspection intervals to improve reliability and availability while reducing cost.

Periodic inspection is the most commonly used inspection policy. However, the information gathered during inspection is used to decide when the next inspection will take place. Hence, non-periodic inspection might be more appropriate, especially when the aging rate of a unit is unknown and must be estimated with information gathered by inspection. Zhao et al. [15] developed a mathematical model to evaluate the reliability of a single-unit system and optimize the inspection schedule. If a defect is detected by non-periodic inspection, the system is repaired immediately but minimally. They assumed that a defect can be detected by inspection with a specific probability. The probability of defect detection had a significant effect on reliability. The optimal inspection intervals were determined by maximizing the reliability of the unit rather than merely meeting the required reliability within a given period $[0, t]$.

Yun et al. [16,17] considered optimization problems to determine inspection intervals for a one-shot device with two types of units. Type 1 units fail at random times and are maintained at inspection times, while Type 2 units do not fail and are replaced at pre-determined times. Yun et al. [18] assumed that Type 2 units in a one-shot system degrade over time, and they described the degradation using a compound Poisson process. Simulation was used to determine the optimal inspection intervals and the preventive maintenance thresholds of Type 2 units based on a genetic algorithm. Age-based preventive maintenance policing was considered for Type 1 units, and the optimal preventive replacement age was obtained by minimizing the life cycle cost [19].

To maintain one-shot systems, a certain amount of resources is required at the maintenance site. Therefore, the maintenance can be delayed when the number of resources is not enough to maintain a great number of one-shot devices at the same time. Hence, an efficient inspection schedule for one-shot systems should be established to reduce maintenance delay. Yun et al. [20] studied inspection schedules for many one-shot devices. They used simulation and a genetic algorithm to determine the inspection intervals and first inspection points of each one-shot device, as well as the preventive maintenance threshold of Type 2 units. An inspection schedule problem was also considered for multiple one-shot devices with limited maintenance resources [21]. Gamma processes were simulated using the gamma bridge sampling method and applied to the degradation of Type 2 units [21]. Some one-shot devices require support equipment to ensure successful operation, which previous studies have not considered. This paper deals with the inspection schedule problem for such systems with limited maintenance resources. Section 2 describes the inspection schedule model for one-shot systems with support equipment, and Section 3 explains a simulation-based optimization procedure using a hybrid EDA-based algorithm. Numerical examples are presented in Section 4, and conclusions are presented in Section 5.

2. Inspection schedule of one-shot systems

This section introduces an inspection policy and proposes an inspection schedule model for one-shot systems with support equipment. We also explain the performance measures of interval availability and the expected life cycle cost which are used as optimization criteria in this paper. The following notation is used for modeling the inspection schedule:

f	Index of periods ($f = 1, 2, 3, \ldots, F$)
R	Number of one-shot devices handled by one support equipment
TA	Target interval availability
C_{FI}	Fixed inspection cost
C_{VI}	Variable inspection cost
C_R^p	Repair cost of Type 1 unit p of one-shot device
C_R^q	Repair cost of unit q in support equipment
$E[LC]$	Expected life cycle cost
TI	Total number of inspections
TI_i	Total number of inspections of one-shot device i
AI_f	Interval availability in the fth period
$N_o(t)$	Number of functioning one-shot devices at time t
$N_s(t)$	Number of functioning support equipment at time t
$E[NR_i^p]$	Expected number of repairs of unit p in one-shot device i
$E[NR_j^q]$	Expected number of repairs of unit q in support equipment j
N_{tot}	Total number of one-shot devices

The following assumptions are made:

1) The life cycle of the one-shot devices and support equipment is finite and given.
2) Inspection is performed perfectly and any failure of one-shot devices can be identified.
3) Failures of support equipment are detected immediately.
4) Replacement times of Type 2 units in the one-shot device are given.
5) Inspection is also performed at the times of replacement for Type 2 units.
6) The number of inspection equipment is limited.
7) Repair is perfect and the state of units after repair is same as new ones.

2.1. Inspection policy of one-shot systems

There are two types of independent units in the one-shot devices. An example of Type 1 units is guidance units in missile systems. This type fails at random times, and such failures can be detected by inspection. Type 2 units such as the powder in missile systems degrade over time, and degradation beyond specific limits makes the system incapable of normal operation. Destructive inspection is commonly used to detect the failure of Type 2 units. In practice, the functional life span of each Type 2 unit in a system is determined in the system development phase, and the units are replaced after this time point is reached. Support equipment units such as the grip stock or BCU in the launchers fail randomly, and the failure can be detected immediately. Therefore, the support equipment is assumed to consist of many units with a series structure that can be repaired upon failure. Multiple one-shot devices are deployed together and stored for a long time until usage or inspection. After inspection, functional units are sent back to storage, but the failed systems are repaired. Repaired systems are also sent back to storage. The one-shot devices are also inspected at the replacement time points of Type 2 units. Support equipment is also deployed at the same place as the one-shot device prior to usage.

2.2. Inspection schedule model for one-shot systems with support equipment

In this sub-section, we consider inspection schedules of several one-shot devices deployed together. Availability is defined as the ability of a unit or system to perform its required function at a particular instant or over period of time [22]. It is usually expressed as the availability probability. One-shot device operation is usually sudden and occurs at most once during the life cycle. For such systems, it is important to maintain the required availability between inspection periods. This makes interval availability the most appropriate optimization criterion for the performance measures of one-shot devices [16–21]. We assume that the support equipment may affect the one-shot device operation and that one support equipment can handle several one-shot devices simultaneously. Hence, the performance of support equipment must be considered in calculating the interval availability.

The maximum number of functioning one-shot devices $M_o(t)$ handled by the functioning support equipment at time t can be calculated using Eq. (1):

$$M_o(t) = Min\{N_o(t), N_s(t) \times R\} \tag{1}$$

The interval availability of one-shot systems $A_{av}(t_1, t_2)$ is defined as the mean proportion of the number of operating one-shot devices in the time interval (t_1, t_2) and can estimated using Eq. (2):

$$A_{av}(t_1, t_2) = \frac{\int_{t_1}^{t_2} E[M_o(t)]dt}{N_{\text{tot}} \times (t_2 - t_1)} \tag{2}$$

As an example, we consider 12 support equipment and 120 one-shot devices. One unit of support equipment can handle at most 10 one-shot devices simultaneously. We assume that all one-shot devices are functioning, but two units of support equipment do not work at time t. Without considering the support equipment, the point availability of the one-shot systems at time t is 1.00 (=120/120), although only 10 support equipment are available [20,21]. However, based on the support equipment, only 100 one-shot devices can operate at time t, even though all devices are available. Hence, the point availability is 0.83 (=100/120) and the interval availability integrates the point availability in the interval (t_1, t_2) as Eq. (2).

The inspection can be delayed due to limited resources when many one-shot devices are scheduled for inspection at the same time and the maintenance resources are not replenished. As a result, the mean down time increases and the interval availability decreases. Therefore, the entire one-shot system should be divided into several groups with a rotational order of inspection to reduce the possibility of delays. As an example, we assume that 500 one-shot devices are each inspected at years 6, 8, and 17. The inspection of just 250 systems may occur at the same time due to limited inspection equipment. Thus, we need to divide the 500 systems into two groups and must inspect 250 systems at years 3, 6, 8, and so on. We determine the inspection intervals of each group with the same number of one-shot systems, as shown in Fig. 1.

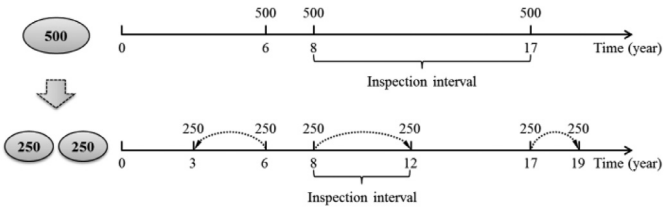


Fig. 1. Example of rotational inspection points of one-shot systems.

The expected life cycle cost is considered as an optimization criterion because inspection incurs maintenance costs, which include the setup cost (fixed inspection cost), inspection cost (variable inspection cost), and repair cost for Type 1 units of one-shot devices. The repair cost for support equipment units is also considered. The aim is to determine the near-optimal inspection intervals to minimize the expected life cycle cost and satisfy interval availability requirements. The optimization model is formulated using Eqs. (3) and (4):

$$Min\ E[LC] = TI \times C_{FI} + \left(\sum_{i=1}^{I} TI_i \times C_{VI}\right) + \sum_{i=1}^{I}\sum_{p=1}^{P}\left(C_R^p \times E\left[NR_i^p\right]\right) + \sum_{j=1}^{J}\sum_{q=1}^{Q}\left(C_R^q \times E\left[NR_j^q\right]\right) \tag{3}$$

$$Subject\ to\ AI_f \geq TA, \forall f \tag{4}$$

3. Optimal inspection schedule of one-shot systems with support equipment

A simulation-based optimization procedure is proposed to determine the near-optimal solution to minimize the expected life cycle cost and satisfy the target interval availability. The EDA, a hybrid EDA-based algorithm, and a heuristic algorithm are applied to generate alternatives. The notations for the heuristic method and the hybrid EDA-based algorithm are as follows:

ε	Allowable gap
R_u	Unit time
$\Delta\bar{l}$	Solutions at the lth generation
P_s	Selection probability
P_g	Sampling probability
ΔA_{sys}	Increment in system availability
$\Delta Cost$	Increment in expected life cycle cost

3.1. Simulation-based optimization procedure

It is difficult to obtain the accurate interval availability and life cycle cost using the model in Section 2.2. We consider a number of one-shot systems with limited maintenance resources and units that may have different failure distributions. The age of the units may also differ depending on the individual repair history. Multiple one-shot systems are deployed at operation sites and inspected at maintenance sites together, where maintenance resources such as equipment and engineers are limited. Hence, the system performance measures are not easy to obtain analytically. Thus, we use a simulation to estimate the life cycle cost and interval availability and propose an algorithm to search the solution space efficiently.

3.1.1. Discrete event simulation process

A discrete-event simulation model is considered in the simulation system. The system is modeled in terms of its state at each point in time

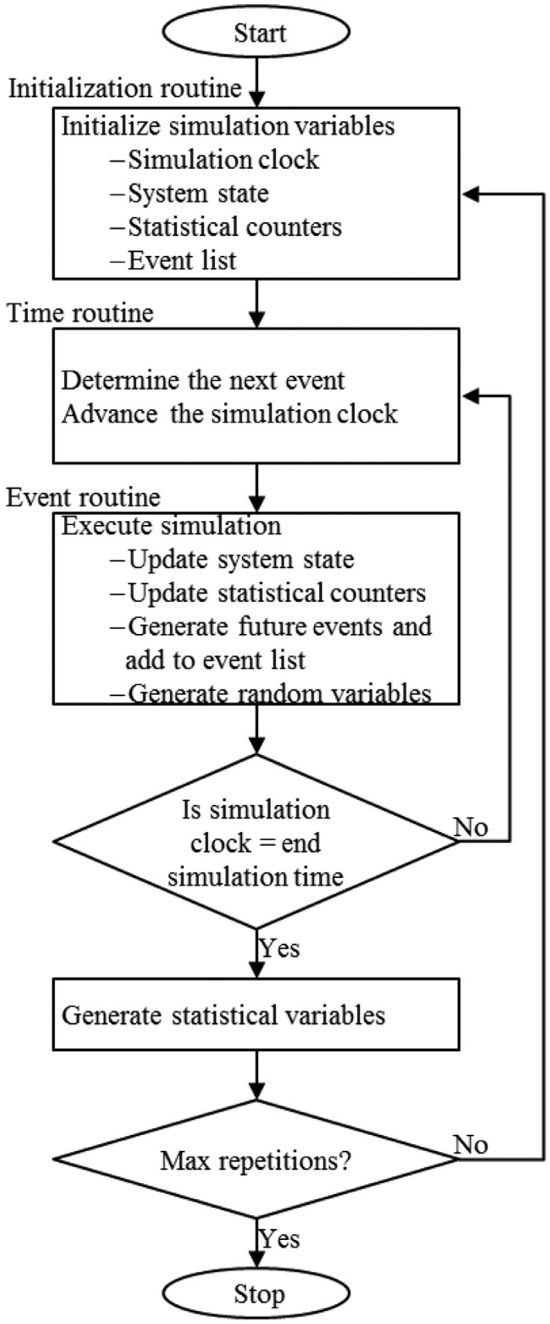


Fig. 2. Flowchart for simulation procedure.

and various events that cause changes of states when they occur [23]. The model was written in Microsoft Visual C++ and shares a number of common components introduced by Law [24], such as the system state, simulation clock, event list, initialization routine, time routine, event routine, and statistical counters. A flowchart of the simulation procedure is given in Fig. 2.

Based on the object-oriented design approach, we define objects for the simulation, function (one-shot device, support equipment), element, and maintenance. The states of the objects are defined as '*RUNNING*', '*IDLE*', and '*DOWN*' [25–27]. However, according to the nature of the maintenance, there is no '*DOWN*' state.

RUNNING: objects operate normally
IDLE: objects do not operate due to failure of another object (not its own failure)
DOWN: object fails to perform as required

Table 1
Events of objects.

Object	Event type	Event description
Simulation	START	Start the simulation
	END	Stop the simulation when the simulation clock equals the length of the simulation
Function (one-shot devices, support equipment)	EMPTY	No event
	functionActivated	Activate the function to operate
	functionDeactivated	Deactivate the function
	eleDown	Notice the function that the element has failed
	eleFixed	Notice the function that the element repair has finished
	IsFail	Check whether the function fails or not
Element	EMPTY	No event
	Activated	Activate the elements to operate
	deActivated	Deactivate the element
	BREAKDOWN	Make the element fail
	ENDFIX	Repair the failed element to ensure it can work normally
Maintenance	PM	Start the system inspection
	ENDPM	Finish the inspection for one system

Table 2
Statistics about system performance.

Name	Description
totDownTime	Total time of element or function in the failure state over a certain period of time
totRunTime	Total time of element or function in the working state over a certain period of time
numBreak	Mean failure times of element or function
numRepair ($NR^p{}_i$ and $NR^q{}_j$)	Mean repair times of element
numDoPM (TI and TI_i)	Total number of inspection times during system life cycle
aviFireSys	Number of functioning systems

In the discrete event simulation model, the system is modeled as sequential events. An event occurs at a given event time and activates the model to change the state. Therefore, we define the events that can change the state of objects in this simulation in Table 1. The variables used for storing statistical information about the system performance are defined in Table 2. Some of the statistics are used to calculate the expected life cycle cost using Eq. (3).

We invoke an initialization routine where the simulation begins at simulation clock 0. The defined system states are initialized to '*IDLE*', the statistical counters are cleared to zero, the events for the simulation object are initialized to '*START*', and the events for function and element objects are initialized to '*EMPTY*'. The events for maintenance objects are initialized to '*PM*', and the maintenance time for '*PM*' events is added to the event list. The time routine will then check the event time to determine the upcoming event type. A '*START*' event with its information for simulation is added to the event list, and the event occurrence time is 0. Therefore, the '*START*' event occurs, and the simulation clock is advanced to the time of its occurrence.

A flowchart for event simulation is given in Fig. 3. The simulation state changes from '*IDLE*' to '*RUNNING*' when the '*START*' event occurs. '*END*' is the next type of simulation event generated, and the occurrence time is the length of simulation, which is also the life cycle of the system. '*functionActivated*' is called to activate all functions that are required for simulation.

The '*functionActivate*' event changes the state of functions from '*IDLE*' to '*RUNNING*'. The '*Activated*' event for the element is called to support the function operation [26]. A flowchart of '*functionActivate*' is also

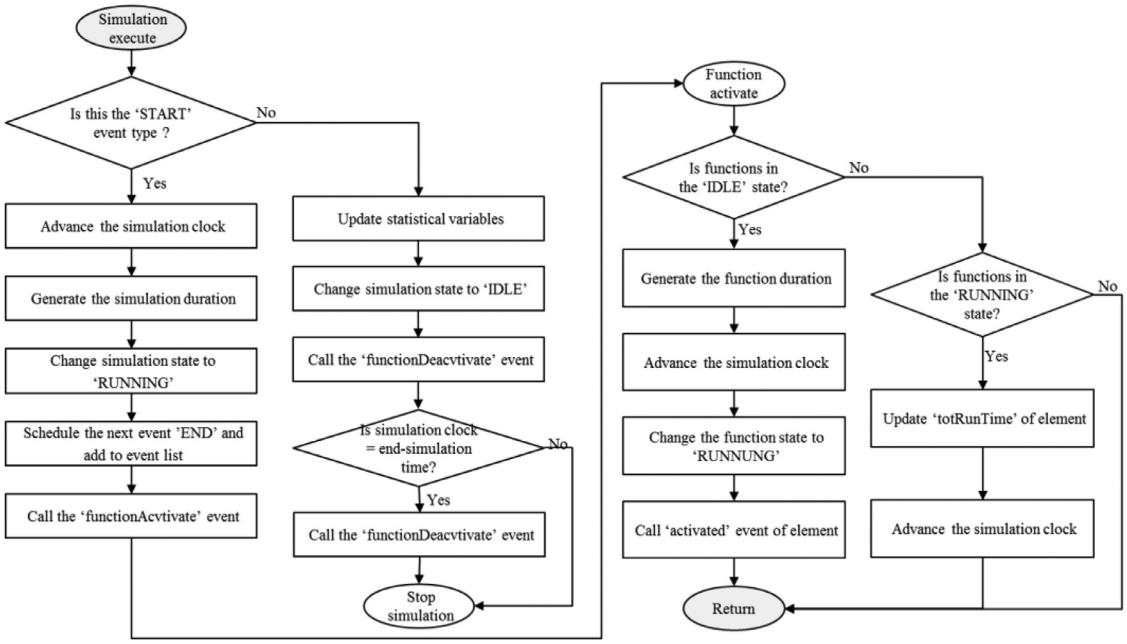


Fig. 3. Flowchart for simulation 'execute' and 'functionActivate' procedure.

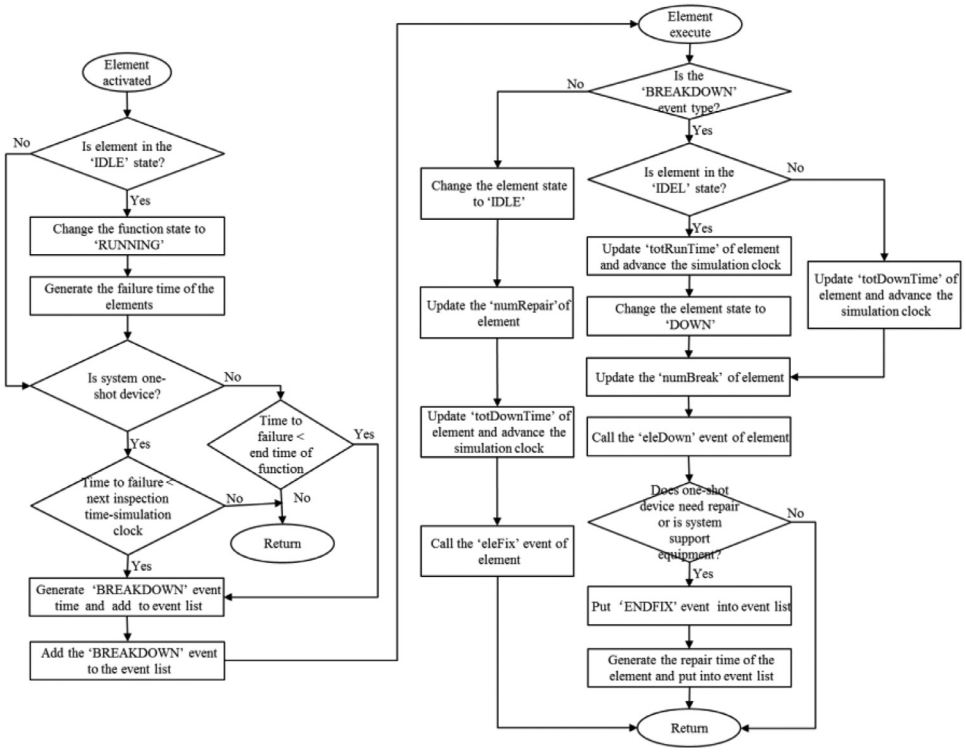


Fig. 4. Flowchart for element 'activated' and 'execute' procedure.

shown in Fig. 3. Each element is activated by the '*activated*' event, and then we check whether each element state is '*IDLE*'. If so, the state changes to '*RUNNING*'. The time to failure for the two types of functions is generated differently. In the case of support equipment, if the random value of time to failure is less than the time difference between the simulation clock and the end of the simulation time, then the time of occurrence of a '*BREAKDOWN*' event for elements is generated as shown in Fig. 4. The information for a '*BREAKDOWN*' event is added to the event list. In the case of one-shot devices, the '*BREAKDOWN*' event will be added to the event list when the random time to failure is less than the time difference between the simulation clock and the next inspection time because the failure of one-shot devices can be detected by inspection. The random time to failure of elements is generated from a predetermined probability distribution [24].

After an event routine has ended, the current simulation clock is checked. If the simulation time is earlier than the simulation termination time, the process returns to the time routine, the most imminent type of event '*BREAKDOWN*' is determined, and the simulation clock is advanced. When a '*BREAKDOWN*' event occurs for an element, the state of the element changes from '*RUNNING*' to '*DOWN*', and the statistical

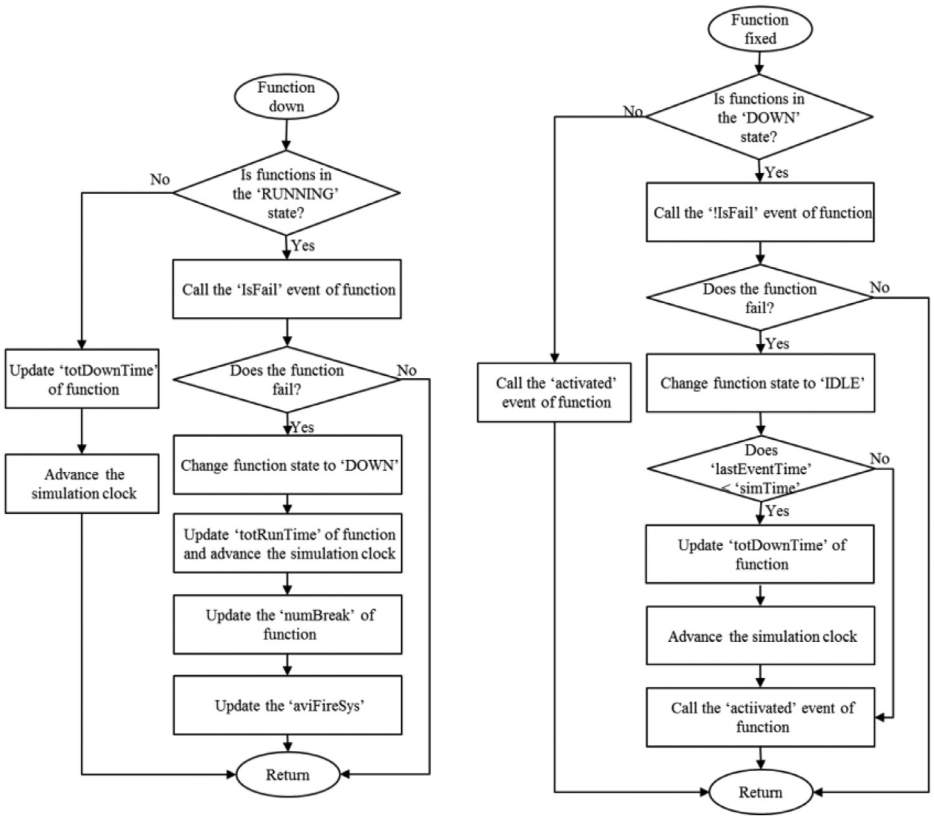


Fig. 5. Flowchart for function 'eleDown' and 'elefixed' procedure.

counters '*numBreak*', '*totDownTime*', or '*totRunTime*' are updated for the elements. An '*eleDown*' event is called to indicate failure to perform a related function (see Figs. 4 and 5). Using the event '*IsFail*', we can check the reliability structures among elements to determine whether there are some failures of a function caused by any element failure. Next, the total number and total run time of the operational one-shot devices are calculated.

If the failure of a function occurs by failed elements, the state is changed to '*DOWN*' by the '*eleDown*' event. '*numBreak*' and '*totRunTime*' are also updated for the function, as are the statistical counters '*aviFireSys*' and '*totRunTime*'. Next, we check whether the failure function comes from the support equipment or whether it is the time to repair the one-shot devices because of the different maintenance policies for the support equipment and one-shot devices. For the support equipment, an element is repaired as soon as the failure is detected. Then, the '*ENDFIX*' event for the element and the end-time of the repair of the failed element are generated and added to the event list. The occurrence time of the '*ENDFIX*' event is equal to the sum of the current simulation time and the repair time, which is a random variate from a probability distribution. The element state changes to '*IDLE*', and '*numRepair* (NR^q_j)' and '*totDownTime*' are updated when the '*ENDFIX*' event occurs.

The '*eleFixed*' event is shown in Fig. 5. This event informs the function that the failed elements have finished repair. The event '!*IsFail*' is used to check whether or not the failed elements have been repaired. If there is no failure of elements, the function state is changed to '*RUNNING*' by the '*eleFixed*' event, and '*totDownTime'* for the function is updated. The end time of repair, '*aviFireSys*', and '*totRunTime'* are also updated. All element states are changed to '*RUNNING*' by an '*activated*' event.

The repair action of one-shot device can only happen after inspection. The inspection information (time) of every one-shot device for a '*PM*' event is added to the event list in the initialization phase. Therefore, inspection will be performed when the simulation clock is advanced to a predetermined inspection time. The flowchart for maintenance execution is shown in Fig. 6. When a '*PM*' event occurs, the maintenance state changes from '*IDLE*' to '*RUNNING*', while '*numDoPM* (*TI* and TI_i)' and '*aviFireSys*' are updated. Meanwhile, the element state changes from '*RUNNING'* to '*DOWN',* even though the element is not in a failure state because it cannot operate during inspection. The function state is also changed from '*RUNNING'* to '*DOWN'* by an '*eleDown*' event, and '*totDownTime*' is also updated.

The next event, '*ENDPM',* and its occurrence time are added to the event list if the inspection equipment is available. Otherwise, the functions keep their previous state. The states of maintenance, element, and function change to '*IDLE*', and the '*totDownTime*' is updated through an '*ENDPM*' event. If all related elements are in the '*IDLE*' state, the states of the function and element are changed to '*RUNNING*' through '*eleFixed*' and '*activated*' events, and the '*totDownTime*' and information for the interval are updated. However, if the inspected element fails before inspection, the '*ENDFIX'* event time is generated for the element and added to the event list. Next, the states of elements after repair are changed to '*IDLE*' through '*eleFixed*', while '*numRepair* (NR^p_i)' and '*totDownTime*' are updated. The states of a function and element are changed to '*RUNNING*' through an '*Activated*' event.

After the event routine ends, the current simulation clock is checked. If the simulation clock is equal to the length of the simulation, the '*END*' event of the simulation occurs. Next, the number of functioning one-shot devices and the other available statistics are updated, and the state of simulation changes to '*IDLE*'. The states of functions and elements return to '*IDLE'* through '*functionDeactivated*' and '*deActivated*'. Finally, these processes are repeated until the simulation termination, the simulation process ends, and the statistics defined in Table 2 are collected.

3.1.2. Estimation of distribution algorithm

Mühlenbein et al. [28] first reported the EDA stochastic optimization algorithm. They proposed a new mode of evolution, which has become

Fig. 6. Flowchart for maintenance 'execute' procedure.

an important research topic in the field of evolutionary computation. The method is also an effective means of solving a range of engineering problems. The basic concept of EDA is the combination of a genetic algorithm (GA) and statistical learning, but without crossover or mutation operators. EDA uses statistical learning methods to establish a probability model that describes the candidate solution in the distribution of the space and then samples new solutions through a probability model. The principle advantage of EDA over a genetic algorithm is the absence of multiple parameters to adjust.

Su and Chow [29] showed that EDA is less time-consuming than a genetic algorithm since it does not perform crossover or mutation operations. A genetic algorithm requires several parameters to be tuned. Hauschild and Pelikan [30] summarized some important advantages of EDA over other metaheuristics. The first is the ability to adapt operators to the structure of a problem. Most metaheuristics use fixed operators to explore the potential solution space. While problem-specific operators may be developed and are often used in practice, EDA can also tune the operators to the problem. Secondly, EDA not only provides the solution to the problem, but also shows how it solves the problem. Finally, incremental EDA reduces memory requirements by replacing the population of candidate solutions with a probabilistic model [31,32]. This enables solutions to extremely large problems that could not be solved with other techniques. Larranaga and Lozano [33] presented the general procedure of EDA.

The three main steps of EDA are (i) selecting solutions from a population, (ii) establishing a probabilistic model to estimate the probability distribution, and (iii) generating a new population by sampling the probability distribution. These steps are repeated until a stopping condition occurs. An example of the stopping condition is when a fixed number of generations are achieved.

Alternatives for the inspection schedule are generated through a proposed hybrid EDA-based algorithm that combines general EDA and a heuristic method. This algorithm is used to search the solution space efficiently, and the simulation is used to estimate the interval availability and life cycle cost. Fig. 7 shows the simulation-based optimization procedure, which has the following steps:

Step 1. Input simulation data, such as the failure and repair distributions of units, maintenance cost, and target interval availability
Step 2. Generate alternative inspection intervals using the hybrid EDA-based algorithm
Step 3. Use the simulation to estimate the interval availability and life cycle cost
Step 4. If $TA \geq AI_f$, the current best solution is the global best solution; otherwise, return to Step 2
Step 5. Extract the statistical information from the global best solution

3.2. A hybrid EDA-based algorithm for the inspection scheduling problem

In the hybrid EDA-based algorithm, the solution to the problem is represented as a chromosome. The unit of time is half a year, and the total number of genes in the chromosome is the number of groups multiplied by twice the number of life cycles. The value of each gene in a chromosome is either '0' or '1'. If the value of the hth gene in the chromosome is '1', then the inspection for the system is performed at year $h \times 0.5$. For example, if the value of the second gene in the chromosome is '1', the system inspection is carried out in year 1. Fig. 8 shows an example of solution representation where the systems of group 1 and 2 are inspected within the first year.

The principal steps of applying the EDA to an inspection scheduling problem can be summarized as follows:

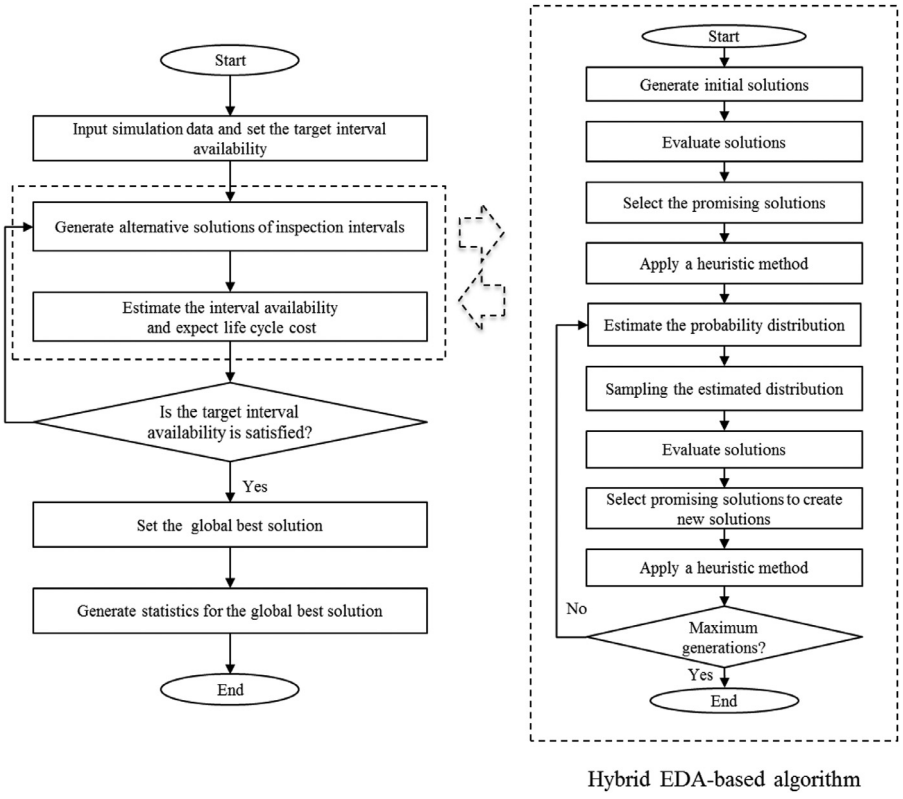


Fig. 7. Simulation-based optimization procedure and hybrid EDA-based algorithm.

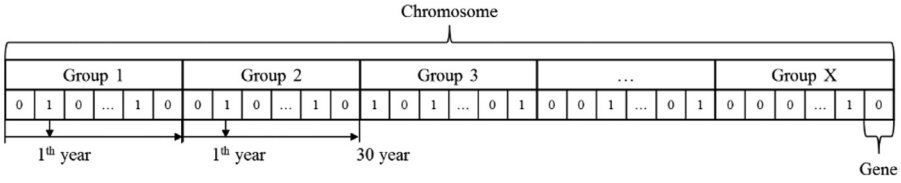


Fig. 8. Example of solution representation.

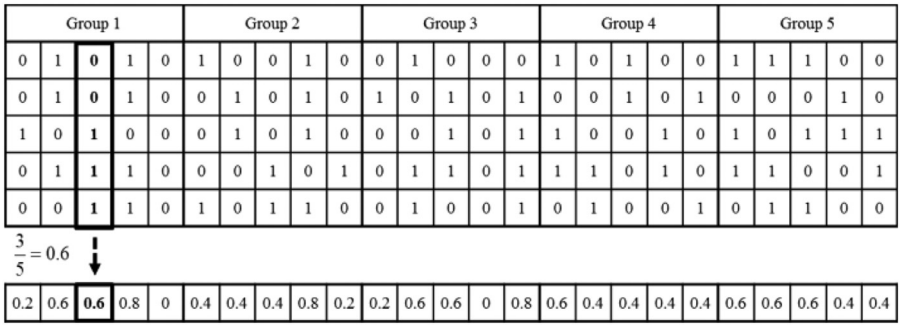


Fig. 9. Example of probability distribution estimation process.

Step 1. Generate inspection intervals of one-shot devices randomly with a uniform distribution

Step 2. Evaluate the initial solutions Δ_0 by estimating the interval availability and expected life cycle cost by simulation

Step 3. Select the Δ_{l-1} P_s best solutions with the best objective values

Step 4. Apply the heuristic method

Step 5. Estimate the probability distribution with the solutions obtained in Step 4, based on the initial solutions, which are generated randomly from uniform distribution. The total number of chromosomes that have the value '1' is used to estimate the probability distribution in Fig. 9 (refer [30]).

Step 6. Generate random values (0, 1) for all genes in the chromosomes and sample the probability distribution for Δ_{l-1} P_g best new solutions. We generate a random value between 0 and 1 for all genes in the chromosomes. If the generated value is larger than the probability value, we assign '0'. Otherwise, we assign '1'. For example, to generate new solutions for the first position in the second chromosome, the two values in the first and second tables in Fig. 10 are checked and the generated value 0.7 is bigger

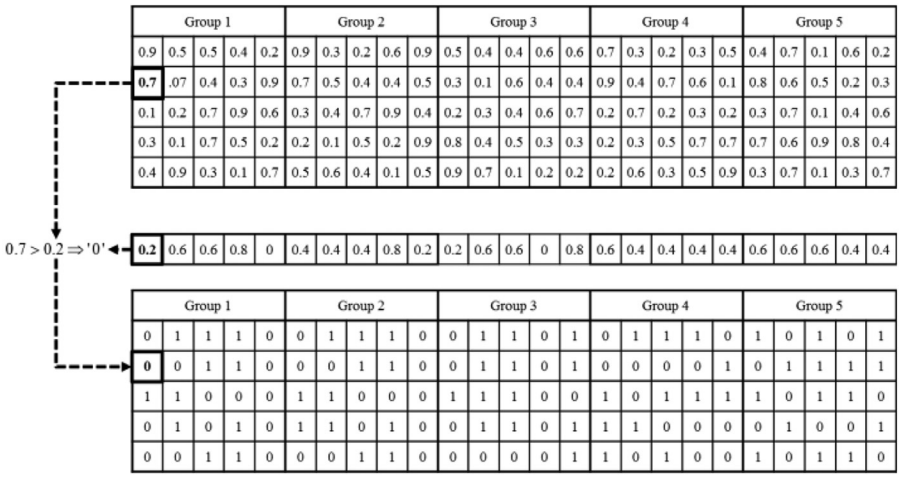


Fig. 10. Example of the process of sampling the probability distribution.

than the probability value 0.2. Thus, value 0 is assigned in this position.

Step 7. Evaluate new the Δ_{l-1} P_g solutions by estimating the interval availability and expected life cycle cost by simulation (repeat Step 2)

Step 8. Select the same number of solutions as the population size (repeat Step 3)

Step 9. Apply the heuristic method

Step 10. If the number of generations G_{Max} has been produced, then terminate. Otherwise, return to Step 4.

3.2.1. *Heuristic method in the hybrid-EDA*

A heuristic method is proposed to improve the alternatives for satisfying the allowable range of target interval availability. The heuristic procedure is terminated if the allowable range is satisfied for all intervals. The hybrid EDA-based algorithm uses the same steps as those in the EDA, and the heuristic method is performed after the selection operation in Step 4 and Step 9 (Section 3.2). The details of the heuristic procedure are as follows:

Step 1. Select a chromosome that has not yet been chosen.

Step 2. Estimate the interval availability for all intervals and the expected life cycle cost of the systems.

2.1. If $TA \le AI_f \le TA + \varepsilon$ (for all f), terminate the heuristic.

2.2. Select the first period that does not satisfy the target interval availability and let the period be the *fth* period. If $AI_f \langle TA$, proceed to Step 3. Otherwise, if $AI_f \rangle TA + \varepsilon$, proceed to Step 4.

Step 3. Check whether the inspection interval at the *fth* period can be changed or not. If the inspection interval of the *fth* period cannot be changed, proceed to Step 3.1. Otherwise, proceed to Step 3.2.

3.1. Add a new inspection point to the middle of the inspection interval of the *fth* period and return to Step 2.

3.2. Reduce the inspection interval of the *fth* period by R_u.

3.2.1. Estimate AI_f and the expected life cycle cost of the systems by simulation.

3.2.2. If $TA \le AI_f \le TA + \varepsilon$, return to Step 2.1.

3.2.3. If $AI_f < TA$, return to Step 3.2.

3.2.4. If $AI_f > TA + \varepsilon$, obtain $\Delta A_{sys}/\Delta$Cost of each group by simulation when each group moves to the next inspection point.

3.2.5. Move the group with the lowest score to the next inspection point.

3.2.5.1. Estimate AI_f and the expected life cycle cost of systems by simulation.

3.2.5.2. If $TA \le AI_f \le TA + \varepsilon$, return to Step 2.1.

3.2.5.3. If $AI_f < TA$, send back the group which has moved to the original inspection point, return to Step 2.1.

3.2.5.4. If $AI_f > TA + \varepsilon$, return to Step 3.2.4.

Step 4. Check whether the inspection interval at the *fth* period can be changed or not. If the inspection interval of the *fth* period cannot be changed, return to 2.1. Otherwise, proceed to Step 4.1.

4.1. Increase the inspection interval of the *fth* period by R_u.

4.1.1. Estimate AI_f and the expected life cycle cost of systems by simulation.

4.1.2. If $TA \le AI_f \le TA + \varepsilon$, return to Step 2.1.

4.1.3. If $AI_f > TA + \varepsilon$, return to Step 4.1.

4.1.4. If $AI_f < TA$, obtain $\Delta A_{sys}/\Delta$Cost of each group by simulation when each group moves to the previous inspection point.

4.1.5. Move the group with the lowest score to the previous inspection point.

4.1.5.1. Estimate AI_f and life cycle cost of systems by simulation.

4.1.5.2. If $TA \le AI_f \le TA + \varepsilon$, return to Step 2.1.

4.1.5.3. If $AI_f < TA$, return to Step 4.1.4.

4.1.5.4. If $AI_f > TA + \varepsilon$, send back the group which has moved to the original inspection point, return to Step 2.1.

3.3. *Heuristic algorithm for the inspection scheduling problem*

We also propose a heuristic algorithm and compare its performance to that of the general EDA and hybrid EDA-based algorithm. The inspection schedule problem is solved in two phases:

1) In Phase 1, we consider a single one-shot device and use a previous method [19] to determine the inspection intervals that minimize the expected life cycle cost and satisfy the target interval availability.
2) In Phase 2, we divide all the one-shot devices into several groups for inspection with limited maintenance resources. The heuristic method explained in Section 3.3.1 is used to determine the inspection intervals for groups of one-shot devices that minimize the expected life cycle cost and satisfy the target interval availability.

3.3.1. *Heuristic method in phase 1*

The procedure of the heuristic method in Phase 1 is as follows:

Step 1 Input data for the reliability and maintainability of units, maintenance costs, replacement intervals of Type 2 units, and target system availability.

Step 2 Determine the initial inspection schedule for only the replacement times of Type 2 units.

Table 3
Failure and repair distributions and repair costs of Type 1 units of one-shot devices.

Unit	Failure distribution	Scale parameter	Shape parameter	Repair distribution	Scale parameter	Shape parameter	Repair cost (KRW)
A	Exponential	300,000	–	Exponential	12	–	4000
B	Exponential	250,000	–	Exponential	12	–	3500
C	Exponential	300,000	–	Exponential	10	–	3000
D	Weibull	250,000	1.5	Exponential	10	–	3500
E	Weibull	300,000	1.5	Exponential	10	–	3500

Step 3 Estimate the interval availability for all intervals and the expected life cycle cost of systems.
- 3.1 If $TA \le AI_f \le TA + \varepsilon$, (for all f), terminate the heuristic.
- 3.2 Select the first period that does not satisfy the target interval availability as the *fth* period. If $AI_f < TA$, proceed to Step 4. Otherwise, proceed to Step 5.

Step 4 Check whether the inspection interval at the *fth* period can be changed or not. If the inspection interval cannot be changed, proceed to Step 4.1. Otherwise, proceed to Step 4.2.
- 4.1 If the inspection interval can be changed, reduce the inspection interval of the *fth* period by Ru.
  - 4.1.1 Estimate AI_f and expected life cycle cost of systems by simulation.
  - 4.1.2 If $TA \le AI_f \le TA + \varepsilon$, return to Step 3.1.
  - 4.1.3 If $AI_f < TA$, return to Step 4.1.
  - 4.1.4 If $AI_f > TA + \varepsilon$, increase the inspection interval of the *fth* period by $R_u/2$ and estimate AI_f by simulation.
    - 4.1.4.1 If $AI_f < TA$, reduce the inspection interval of the *fth* period by $R_u/2$ and proceed to Step 3. Otherwise, return to Step 3.
- 4.2 If the inspection interval cannot be changed, set a new inspection point at the middle of the inspection interval of the *fth* period and return to Step 3.

Step 5 Check whether the inspection interval at the *fth* period can be changed or not. If the inspection point cannot be changed, return to Step 3.1. Otherwise, proceed to Step 5.1.
- 5.1 Increase the inspection interval of the *fth* period by R_u.
  - 5.1.1 Estimate AI_f and the expected life cycle cost of the systems by simulation.
  - 5.1.2 If $TA \le AI_f \le TA + \varepsilon$, return to Step 3.1.
  - 5.1.3 If $AI_f < TA$, reduce the inspection interval of the *fth* period by $R_u/2$ and estimate AI_f by simulation.
    - 5.1.3.1 If $AI_f < TA$, reduce the inspection interval of the *fth* period by $R_u/2$ and return to Step 3. Otherwise, return to Step 3.
  - 5.1.4 If $AI_f > TA + \varepsilon$, return to Step 5.1.

3.3.2. Heuristic method in phase 2

The heuristic method for Phase 2 is that used in the hybrid EDA-based algorithm in Section 3.2.1.

4. Numerical examples

Consider 500 one-shot devices and 50 units of support equipment, each of which can support 10 one-shot devices. We define the system rate as the number of one-shot devices that can be supported by one support equipment (i.e., the system rate is 10). Three units of inspection equipment are used. The inspection time per system is 10 h. The inspection cost per one-shot device is 1,100,000, and the setup cost per inspection is 770,000. There are seven one-shot units (*A* to *G*) with a series structure. Table 3 shows the failure and repair distributions and the repair costs of Type 1 units. We consider the same distributions as those in previous studies [16–21]. We assume that the time to failures of Type 1 units follows an exponential distribution or Weibull distribution. Table 4 shows the replacement distribution, lifetimes, and the replacement costs of Type 2 units [16–21].

Table 4
Replacement distributions, lifetimes, and replacement costs of Type 2 units of one-shot devices.

Unit	Life cycle (*year*)	Replacement distribution	Scale parameter	Replacement cost (KRW)
F	10	Exponential	9	4000
G	14	Exponential	9	4000

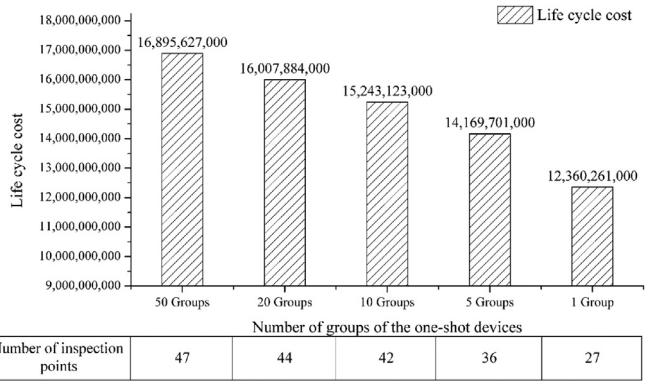


Fig. 11. Life cycle costs and number of inspection points for different numbers of groups.

The support equipment consists of five units (*H* to *L*) with a series structure, as shown in Table 5. The time to failures of units of support equipment also follows an exponential distribution or Weibull distribution. The parameters for EDA are a population size of 50, generation size of 50, selection rate of 0.5, and sampling rate of 0.5. The lifespan of both the one-shot devices and support equipment units is 30 years (262,800 h) [16–21], and the number of replications is 100.

4.1. Effects of group number and target interval availability

With a target interval availability of 0.7, we first consider cases with 1, 5, 10, 20, and 50 groups corresponding to 500, 100, 50, 25, and 10 one-shot devices, respectively. Fig. 11 shows that the life cycle cost and the number of inspection points decrease as the number of groups decreases because inspections are less frequent, and the setup cost (fixed inspection cost) account for a bigger proportion of the life cycle cost.

Table 6 shows the average interval availability and standard deviation for cases with groups of different sizes. The average interval availability increases with the number of groups. For one group, the interval availability over the entire inspection period is higher than the target system availability (0.70) with the lowest standard deviation (0.065). Therefore, the best solution among all considered cases can be obtained without grouping if the target availability is 0.70.

Secondly, if we increase the target interval availability from 0.70 to 0.80, the life cycle cost increases because more frequent inspections are needed to satisfy higher target interval availability. However, not all interval availabilities satisfy this higher target interval availability (0.80) with just one group. Thus, even though using one group leads to the lowest expected life cycle cost, the one-shot devices require at least five groups, as shown in Fig. 12.

Table 5
Failure and repair distributions and repair costs of support equipment units.

Unit	Failure distribution	Scale parameter	Shape parameter	Repair distribution	Scale parameter	Shape parameter	Repair cost (KRW)
H	Exponential	70,000	–	Exponential	8	–	2000
I	Exponential	60,000	–	Exponential	8	–	1500
J	Exponential	70,000	–	Exponential	10	–	2000
K	Weibull	60,000	1.5	Exponential	8	–	1500
L	Weibull	70,000	1.5	Exponential	10	–	2000

Table 6
Average interval availability and standard deviation.

Number of groups	Average interval availability	Standard deviation
1	0.7516	0.0651
5	0.8139	0.1352
10	0.8257	0.1393
20	0.8345	0.1425
50	0.8395	0.1469

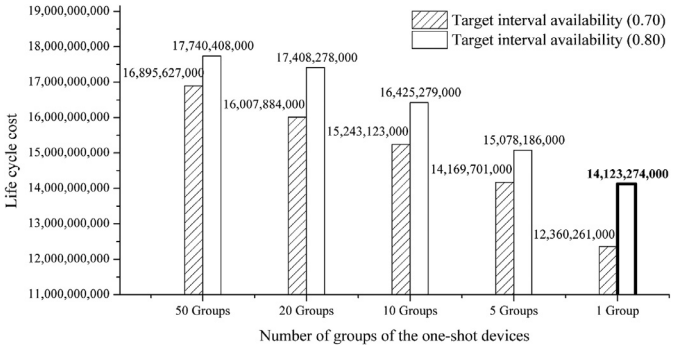


Fig. 12. Life cycle costs for different target interval availabilities.

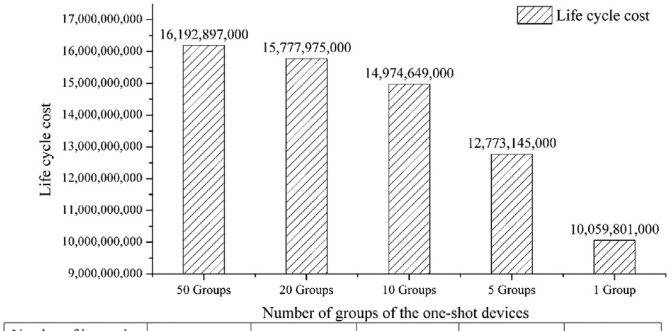
Table 7
Life cycle costs for different cost rates and numbers of groups.

Cost rate	Life cycle cost (in millions)				
	50 groups	20 groups	10 groups	5 groups	1 group
11/7	16,895.63	16,007.88	15,243.12	14,169.70	12,360.26
1	11,784.27	11,160.98	10,751.84	8903.03	5162.34
0.1	1296.25	1249.21	1202.98	961.08	730.61
0.01	248.84	245.50	237.07	210.22	175.78

Setup cost occurs at each inspection point, and inspection cost occurs for each system inspection. More frequent inspection will lead to higher total setup cost but lower total inspection cost because all one-shot devices are divided into groups. Therefore, if the setup cost is too high, we should reduce the number of inspection points to reduce the high total setup cost. However, if the setup cost is not very high, more points will be more appropriate. We also discuss the cost rate between the inspection and setup cost. Table 7 shows that the life cycle cost is lowest when we do not divide one-shot devices into groups and inspect all of them at one inspection point, despite the increased cost rate.

4.2. Effects of support equipment

If we assume that the support equipment does not fail during the system life cycle, the life cycle cost and number of inspection points would be smaller than if the support equipment could fail (see Figs. 13 and 11). This occurs because failed support equipment cannot support the functional one-shot devices. Therefore, to achieve the target interval availability for each interval, the reliability of the support equipment also needs to be considered. Table 8 shows the improvement value of the average interval availability for different numbers of groups. The value is 3.14% for one group and 1.95% for five groups.



Number of inspection points	44	42	37	28	18

Fig. 13. Life cycle costs and number of inspection points for different numbers of groups.

Table 8
Improvements in average interval availability when support equipment failure is not considered.

Number of groups	Average interval availability		Improvement
	Support equipment can fail	Support equipment cannot fail	
1	0.7516	0.7830	**+3.14%**
5	0.8139	0.8334	**+1.95%**
10	0.8257	0.8341	**+0.84%**
20	0.8345	0.8381	**+0.36%**
50	0.8395	0.8413	**+0.18%**

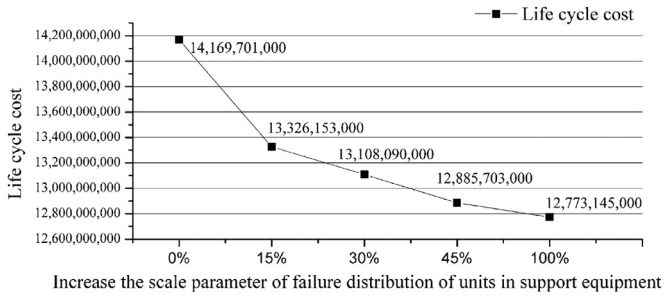


Fig. 14. Life cycle costs for different failure distribution scale parameters of support equipment for five groups.

We next increase the scale parameter of the failure distribution of the support equipment units by 15, 30 and 45%. Fig. 14 shows that the life cycle cost decreases because the reliability of the support equipment increases, so the support equipment can handle more operating one-shot devices. The interval availability may be increased, and more frequent inspections are not needed. Table 9 shows the improvement value of the average interval availability for when the scale parameter of the failure distribution is increased. The results show that using fewer groups increases the improvement of the average interval availability (see Table 8), and the average interval availability increases with the reliability of the support equipment (see Table 9).

Finally, we change the numbers of one-shot devices that can be supported by support equipment to 1, 5, 10, 20, and 25 for 500 one-shot

Table 9
Improvement of the average interval availability for five groups.

Increment	Average interval availability	Improvement
0%	0.8139	–
15%	0.8194	**+0.55%**
30%	0.8238	**+0.99%**
45%	0.8295	**+1.56%**
100%	0.8334	**+1.95%**

Table 10
Life cycle costs for different system rates and different number of groups.

System rate	Number of support equipment	Life cycle cost (in millions)				
		50 groups	20 groups	10 groups	5 groups	1 group
1	500	164,14.71	15,334.57	14,503.08	11,915.83	11,178.56
5	100	166,12.08	15,683.02	15,042.49	12,883.81	11,708.64
10	50	168,95.63	16,007.88	15,243.12	13,169.70	12,360.26
20	25	165,69.84	15,423.10	14,644.79	12,328.43	11,146.43
25	20	168,28.49	15,693.80	15,033.19	12,998.82	11,714.27

Table 11
Performance comparison between the Hybrid EDA-based algorithm and exhaustive method.

Item	Methods	Value	Gap (%)
Expected life cycle cost	Exhaustive method	61,150,880	−0.343%
	Hybrid EDA-based algorithm	60,941,060	
Inspection points	Exhaustive method	15th year	0%
	Hybrid EDA-based algorithm	15th year	
Average interval availability	Exhaustive method	0.7264	−0.192%
	Hybrid EDA-based algorithm	0.7250	
Computation time (s)	Exhaustive method	23.16	+0.561%
	Hybrid EDA-based algorithm	23.29	

devices. Table 10 shows that the life cycle costs for different system rates are not significantly different for each number of groups. In this case, the total number of one-shot devices is fixed (500), the number of support equipment decreases as the system rate increases, and the support equipment are independent. Because the failure probabilities of all support equipment are identical, the expected number of failed support equipment in a certain time period decreases as the system rate increases, and the expected number of malfunctioning one-shot devices caused by failed support equipment is the same.

4.3. Comparison between the proposed method and exhaustive method

To evaluate the proposed method, we check the difference between the proposed method and the exhaustive method for a small size problem. In this example, we consider 50 one-shot devices and 10 support equipment. For convenience, we also changed the unit interval from 0.5 to 5 years. This means the one-shot devices can be inspected at 5, 10, 15, 20, 25 years. For the hybrid EDA-based algorithm, the initial solution is narrowed down to 5. The other input information remains unchanged. The optimal solution based on the exhaustive method is shown in Table 11 for a target interval availability of 0.70. The one-shot devices are inspected at the 15th year, which is same as ones obtained by the hybrid EDA-based algorithm. The result shows that the proposed method gives the result similar to the result of the exhaustive method. However, as the problem size increases, the computation time of the exhaustive method will increase rapidly.

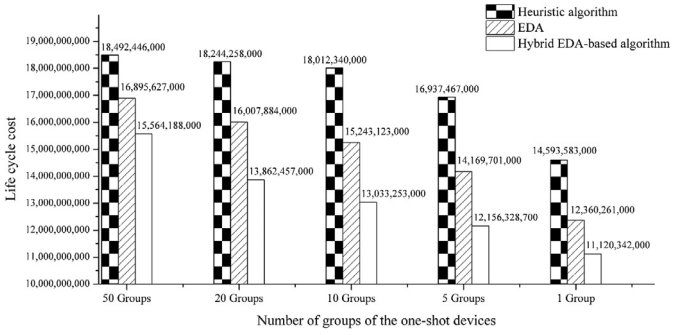


Fig. 15. Life cycle costs for different numbers of groups.

Table 12
Comparison results for different numbers of groups.

Number of groups	Method	Total number of inspection points	Computation time (min)
1	Hybrid EDA-based algorithm	21	782
	General EDA	27	421
	Heuristic algorithm	37	222
5	Hybrid EDA-based algorithm	26	804
	General EDA	36	449
	Heuristic algorithm	47	239
10	Hybrid EDA-based algorithm	30	798
	General EDA	42	450
	Heuristic algorithm	57	264
20	Hybrid EDA-based algorithm	33	809
	General EDA	44	454
	Heuristic algorithm	59	266
50	Hybrid EDA-based algorithm	39	827
	General EDA	47	461
	Heuristic algorithm	60	272

4.4. Comparison of algorithms proposed

We also compared the performance of the hybrid EDA-based algorithm, general EDA, and heuristic methods. The results were tested on a computer with an Intel(R) Core(TM) 3.20 GHz I5-6500 CPU and 4.00 GB of memory. The algorithms were coded in C++. Fig. 15 compares the life cycle costs for different numbers of groups achieved by all methods, and Table 12 shows the computation time needed to obtain the near-optimal solutions. The comparison results show that the hybrid EDA-based algorithm determines a lower average life cycle cost than both the general EDA and the heuristic algorithm. However, the heuristic method is faster on average than both the hybrid EDA-based algorithm and general EDA. The heuristic search is very fast because it does not wait for the search to end, but it does not guarantee the best solution.

5. Conclusions

We have investigated an inspection schedule problem for several one-shot systems with support equipment and the replacement times of Type 2 units within one-shot devices. The interval availability and life cycle cost were used as optimization criteria to evaluate the performance of the one-shot devices and support equipment and were estimated by simulation. EDA and a heuristic method were used to determine alternatives for the inspection schedule of the systems. The numerical results showed that the inspection is performed more frequently when the total number of groups is increased and when the target interval availability increases. The average interval availability increases as the scale parameter of the failure distribution for units of support equipment decreases over the life cycle of one-shot systems. For a fixed number of one-shot devices, the system rate and number of support equipment do not have any impact on the life cycle cost.

We also compared the hybrid EDA-based algorithm with, general EDA, and a heuristic algorithm. The hybrid EDA-based algorithm provides lower life cycle cost with higher processing times on average than both the general EDA and heuristic algorithm. In future work, we will consider an inspection scheduling problem with a multi-echelon structure for maintenance, as well as the maintenance for the support equipment.

Acknowledgment

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) and funded by the Ministry of Education, Science, and Technology (NRF-2016R1D1A1A09919337).

References

[1] Nakagawa T, Mizutani S. A summary of maintenance policies for a finite interval. Reliab Eng Syst Safety 2009;94:89–96.
[2] Nakagawa T, Mizutani S, Chen M. A summary of periodic and random inspection policies. Reliab Eng Syst Safety 2010;95:906–11.
[3] Hariga MA. A maintenance inspection model for a single machine with general failure distribution. Microelectron Reliab 1996;36:353–8.
[4] Chelbi A, Ait-Kadi D. An optimal inspection strategy for randomly failing equipment. Reliab Eng Syst Safety 1999;63:127–31.
[5] Huynh KT, Barros A, Bérenguer C, Castro IT. A periodic inspection and replacement policy for systems subject to competing failure modes due to degradation and traumatic events. Reliab Eng Syst Safety 2011;4:497–508.
[6] Van der Weide JAM, Pandey MD. A stochastic alternating renewal process model for unavailability analysis of standby safety equipment. Reliab Eng Syst Safety 2015;139:97–104.
[7] Cui L, Zhao X, Shen J, Xu Y. An availability model for storage products under periodical inspections. Int J Reliab, Qual Safety Engineering 2010;17:89–103.
[8] Ito K, Nakagawa T. Optimal inspection policies for a system in storage. Comput Math Appl 1992;24:87–90.
[9] Ito K, Nakagawa T. An optimal inspection policy for a storage system with high reliability. Microelectron Reliab 1995;35:875–82.
[10] Ito K, Nakagawa T. An optimal inspection policy for a storage system with three types of hazard rate functions. J Oper Res Soc Jpn 1995;38:423–31.
[11] Ito K, Nakagawa T. Optimal inspection policies for a storage system with degradation at periodic tests. Math Comput Model 2000;31:191–5.
[12] Ito K, Nakagawa T, Nishi K. Extended optimal inspection policies for a system in storage. Math Comput Model 1995;22:83–7.
[13] Badía FG, Berrade MD, Campos CA. Optimization of inspection interval based on cost. J Appl Probab 2001;38:872–81.
[14] Ten WoldeM, Ghobbar AA. Optimizing inspection intervals—reliability and availability in terms of a Cost model: a case study on rail way carriers. Reliab Eng Syst Safety 2013;114:137–47.
[15] Zhao J, Chan AHC, Roberts C, Madelin KB. Reliability evaluation and optimization of imperfect inspections for a component with multi-defects. Reliab Eng Syst Safety 2007;92:65–73.
[16] Yun WY, Han YJ, Kim HW. Simulation-based inspection policies for a one-shot system in storage over a finite time span. Commun Stat-Simul Comput 2013;43:1979–2003.
[17] Yun WY, Han YJ, Kim HW. Simulation-based inspection policies for a one-shot system. In: Yamamoto H, Qian C, Cui L, Dohi T, editors. Proceeding 5th Asia-Pacific international symposium on advanced reliability and maintenance modeling; 2012 November 1-3. Taiwan: McGraw-Hill; 2012. p. 621–8.
[18] Yun WY, Han YJ, Kim HW. Optimal inspection policies for a one-shot system with two types of units. In: Proceedings of the 9th international conference on intelligent manufacturing and logistics systems. Shanghai, China; 2013. p. 136–41. February 27-March 2.
[19] Yun WY, Liu L, Han YJ. Metaheuristic-based inspection intervals for a one-shot system with two Types of units. J Mech Sci Technol 2014;28:3947–55.
[20] Yun WY, Liu L, Han YJ, Rhee DW, Han CG. Optimal inspection schedules for one-shot systems with two types of units. In: Proceedings of the 2013 international conference on quality, reliability, risk, maintenance, and safety engineering, Sichuan, China; 2013 July15-18.
[21] Yun WY, Liu L, Han YJ, Rhee DW, Han CG. Simulation-based optimal inspection schedules for one-shot systems with two types of units. In: Proceedings of the 17th international conference on industrial engineering: theory, application and practice, Busan, Korea; 2013 October 6-9.
[22] Rausand M, Høyland A. System reliability theory: models, statistical methods, and applications. 2nd ed. Canada: Wilely-Interscience; 2003.
[23] Banks J, Carson JS II, Nelson BL, Nicol DM. Discrete-event system simulation. 4th ed. New Jersey: Prentice-Hall; 2005.
[24] Law AM. Simulation modeling & analysis. 4th ed. New York: McGraw-Hill; 2007.
[25] Lee JY, Han YJ, Yun WY, Bin JG. Simulation-based risk analysis of integrated power system. J Korean Inst Ind Eng 2016;42:151–64.
[26] Han YJ. Optimal RAM design and maintenance schedule for a multi-unit system (doctoral thesis). Busan, South Korea: Pusan National University; 2015.
[27] Han YJ, Yun WY. Joint optimization problem of RAM design and PM intervals for a multi-unit system. In: Proceedings of the 3rd international conference on materials and reliability, Jeju, Korea; 2015 November 23-25.
[28] Mühlenbein H, Bendisch J, Voigt H-M. From recombination of genes to the estimation of distributions II. continuous Parameters. In: Proceedings of the 4th conference on parallel problem solving from nature. 1996 September. Berlin: Springer; 1996. p. 188–97.
[29] Su W, Chow MY. Performance evaluation of an EDA-based large-scale plug-in hybrid electric Vehicle charging algorithm. IEEE Trans Smart Grid 2012;3:308–15.
[30] Hauschild M, Pelikan M. An introduction and survey of estimation of distribution algorithms. Swarm Evol Comput 2011;1:111–28.
[31] Harik G. Linkage learning via probabilistic modeling in the ECGA. University of Illinois; 1999. January Technical Report No.: 99010, IlliGAL.
[32] Sastry K, Goldberg DE, Llorà X. Towards billion-bit optimization via parallel estimation of distribution algorithm. In: GECCO 2007: Proceedings of the 9th annual conference on genetic and evolutionary computation. New York: AMC; 2007. p. 577–84.
[33] Larrañaga P, Lozano JA. Estimation of distribution algorithms: a new tool for evolutionary computation. 2nd ed. Boston: Kluwer Academic; 2001.