Printed in India

Phased Mission System Design Optimisation Using Genetic Algorithms

D.ASTAPENKO and L.M.BARTLETT*

Department of Aeronautical and Automotive Engineering, Loughborough University, Leicestershire, UK, LE11 2TU

Abstract: A phased mission system represents a system whose performance is divided into consecutive non-overlapping phases. It is important to ensure safety of a phased mission system since the failure of it can have both life threatening and financial consequences. The focus of this paper is to develop an optimisation method to construct an optimal design case for a phased mission system, with the aim of minimising its unreliability and at the same time ensuring optimal usage of available resources throughout all phases. The introduced phased mission optimisation is represented as the constrained single objective problem. Here failure of the overall mission is the objective function and the introduced constraints are employed to determine limits for resources within which their optimal usage is to be determined. The implemented optimisation method employs Fault Tree Analysis to represent system performance and Binary Decision Diagrams to quantify each phase failure probability. A Single Objective Genetic Algorithm has been chosen as the optimisation technique. An Unmanned Aerial Vehicle mission has been selected to demonstrate the methods application. The results and the influence of modifications to the optimisation algorithm are discussed.

Keywords: *phased missions, optimisation, genetic algorithms*

1. Introduction

A phased mission system represents a system where performance is analysed as a sequence of non-overlapping phases. To complete a mission the system is required to accomplish a specific task without failure in each phase. A classical example of a phased mission system is for an aircraft which undergoes three phases: take-off, cruise and landing. The aircraft completes its mission if all three tasks have been completed successfully.

*Corresponding author's email : L.M.Bartlett@lboro.ac.uk

A large number of systems which can employ mechanical, chemical, electronic and nuclear technologies can be analysed as phased mission systems. Similarly, military missions can be analysed as phased mission systems. Thus system optimisation in terms of reliability, availability, maintainability and cost (RAM&C) is extremely relevant. In spite of its importance, however, there is limited demonstrated evidence in the literature for research that focuses on phased mission optimisation; Susova G.M. & Petrov A.N. (1997) introduced a model based on Markov homogeneous process which can be used to minimize operation cost and optimise flight safety.

Generally, methods used for phased mission system analysis can be grouped in two categories: Markov analysis based approaches and combinatorial methods. Combinatorial methods represent fault tree and binary decision diagram (BDD) approaches. The earliest applications of fault tree analysis in the analysis of non-repairable phased missions were made by Esary J.D. & Ziehms, H. (1975). La Band R.A. & Andrews J.D (2004) introduced their approach that is computationally more efficient since BDD methodology is employed together with the fault tree approach.

If a phased mission system is repairable or there are dependencies between component failures or phases, it is generally difficult to find the exact system reliability. In this case Markov reliability models are employed, as discussed by Kim & Park (1994) and Smotherman & Zemoudeh (1989).

Both combinatorial and Markov methods have some drawbacks. For example, in Markov models for systems with n components, up to 2^n equations are needed to represent each phase. In the combinatorial approach, however, the problem size expands with increasing number of phases. Therefore, employing a combination of both approaches for the same problem can help to overcome these individual drawbacks and enable the analysis of more complicated problems. As such, Ou Y. et al. (2002) and Wang D. & Trivedi K.S. (2007) introduced new approaches to analyse phased mission systems adopting this approach.

The developed optimisation approach is based on the single objective Genetic Algorithm (GA). GAs are widely used in reliability optimisation problems as stated by Levitin G. (2006) Torres-Echeverria A.C. *et al*. (2008) introduced a multi-objective genetic algorithm of a safety-instrumented system. Marseguerra M. (2006) discussed the application of GAs for reliability, availability, maintainability and safety optimisation problems. The principals of the GA allows easy implementation and adaptation of the algorithm according to a solved problem. Other important advantage is that only values of an objective function are required which represent the search space. Therefore a methodology for evaluation of objective function values and GA remain independent.

The purpose of this paper is to introduce an optimisation algorithm with implemented GA that aims to minimise overall mission failure probability and is applicable to different phased mission systems. The optimisation is based on the search of system design with the minimal failure probability. All different system designs introduced for the problem are developed by introducing redundant components and components with different reliability characteristics. The algorithm solves for minimum overall mission failure probability within the context of pre-defined resource constraints as well as system failure probability values during each phase.

2. Phased mission design optimisation algorithm

2.1. Optimisation problem introduction

A phased mission system design optimisation problem is represented as a general single objective minimisation problem. The problem is stated as a minimisation of overall mission failure probability for a system:

$$\min Q(\mathbf{X})_{mission} \tag{1}$$

where **X (***n*–dimensional vector of independent variables**)** is the result of the union of vectors of system component failure probability values, i.e.:

$$\mathbf{X} = \bigcup_{i=1}^{m} \mathbf{X}_i \tag{2}$$

Here, *m* is the number of phases in the mission and each $\mathbf{X}_i$ vector represents system components that appear in any minimal cut set of a fault tree of phase *i* (*i* = 1, 2, …, *m*). In other words, **X** is a vector of system components that appear in any failure event.

In the algorithm it is considered that the system failure probability is subject to a number of constraints. The constraints can be grouped in two categories. The first constraint group represents the limits of the available resources, such as cost ($Cost_{mission}$), weight ($Weight_{mission}$) and volume ($Volume_{mission}$) (Eq. 3). To ensure the efficient use of resources minimum constraints are introduced. If only limits for maximum values are considered then minimum constraint values are set to zero.

$$\begin{aligned} &Cost_{\min} < Cost_{mission} < Cost_{\max}, \\ &Weight_{\min} < Weight_{mission} < Weight_{\max}, \\ &Volume_{\min} < Volume_{mission} < Volume_{\max}, \end{aligned} \tag{3}$$

In some case a number of system parts may not be in use during certain phases that results in diverse allocation of resources throughout the phases. Therefore the ability to set resource constraints for each phase is introduced in the algorithm.

The second group of constraints represents the system failure probability during each phase:

$$\begin{aligned} &Q_1(\mathbf{X}_1) \le Q_1(\mathbf{X}_1)_{\max} \\ &Q_2(\mathbf{X}_2) \le Q_2(\mathbf{X}_2)_{\max} \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &Q_m(\mathbf{X}_m) \le Q_m(\mathbf{X}_m)_{\max} \end{aligned}, \tag{4}$$

where, $Q_i(\mathbf{X}_i)$ identifies the *i*th phase failure probability, $Q_i(\mathbf{X}_i)_{\max}$ is the maximum allowed system failure probability value at phase *i* and *m* defines the number of phases in the analysed mission.

Implementing these constraints allows component combinations to be identified that minimise the failure probability of the whole mission without exceeding limits set for system failure probability values during each phase.

2.2 Objective Function Description

One of the advantages of GAs is possibility to separate evaluation process of the objective function from the algorithm itself since the only information required for the optimisation process is values of the objective function. In the introduced case a value of the overall mission failure probability for a particular design system is an objective function value. However an explicit form of the objective function does not exist and methodology for the quantification of phased mission failures is employed instead.

There are a number of methods that can be employed to evaluate the mission failure probability. In the proposed optimisation algorithm a methodology introduced by Prescott at al. (2008) was chosen. Their approach provides failure probabilities for each phase (Q_i) together with the whole mission failure probability ($Q_{mission}$) where

$$Q_{mission} = \sum_{i=1}^{m} Q_i \tag{5}$$

It means that the method can be used to evaluate the objective function value and also that it can be employed to check the validity of constraints for the system failure probability at each phase.

In the proposed methodology, for Equation 5 to be valid, the logical expression for mission failure in each phase is considered as a combination of causes of success of previous phases and the causes of failure for the phase being considered. In other words, if a system fails in phase *i*, it means it could not have failed during any previous phase *j* (*j* = 1, 2, …, *i*-1). For example, system failure in phase *i* can be represented by an "AND" gate that incorporates the success of previous phases *j* (using NOT logic) and the failure for the phase *i*. Figure 1 represents the failure of a system in phase 3.

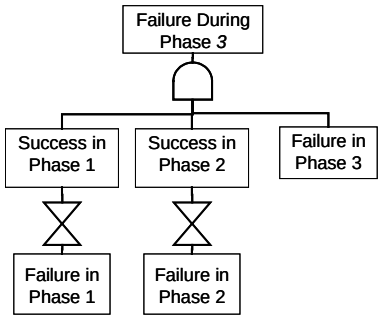

Fig 1: An Example of System Failure in Phase 3

The methodology includes the following steps. At the beginning a fault tree for each phase is converted into a BDD using its own variable ordering scheme. The next step involves assigning time intervals over which each variable contributes to phase failure for

each BDD representing the logical expression for the failure conditions being met in phase *i*. Then the earlier described logical expression for mission failure in phase *i* is built. The resulting BDDs are then constructed. Before the quantification starts the simplification process for every BDD is performed. It involves the simplification of the failure logic for each possible path from a BDD root vertex to its terminal 1 vertices which represent system failure states. Finally every BDD path terminating in a 1 vertex is quantified to give the probability of a corresponding BDD and the sum of all probability values determines the failure probability value for the whole mission of a particular design system.

Equation 1 expresses the objective function for the analysed problem. Even though the function does not have an explicit form it can be defined as the function of variables **X**. The vector is the result of the union of vectors of system component failure probability values as defined by Equation 2. Since failure rates for the components are defined a priory mission failure probability alters due to changes in the system design. Therefore, the contents of the employed component sets and the number of components in the sets vary through the analysis. The dimension *n* of vector **X** (number of system components) is not fixed and is altered during the whole optimisation process.

However it can not be affirmed that the vector **X** comprises decision variables of the optimisation problem even though it can be used as the ones. The vector is also an auxiliary set of data for setting a link between mission failure probability and changes of the system design.

Usually, in trying to improve system performance, a certain number of components are chosen to be replaced. A different number of redundant components and / or components with different failure characteristics, i.e. different type components can replace existing components. Therefore the notation "*design variables"* is used to identify possible types of new components, the numbers of redundant components and redundancy types. Each replaceable component can be associated with more than one design variable. For example, an existing safety system valve was chosen to be replaced. It can be substituted with either two or three redundant valves. There is also possibility to use valves of two different types. Thus two design variables applies to the valve: the type of the valve and the number of redundant valves.

One set of design variables is produced for an individual problem. Any two vectors of design variable values are considered to be different if at least one value of a variable is not the same. It means each set of design variable values corresponds to an individual system design which can be identified with a corresponding components set **X**. Therefore the search for the optimal design which minimises mission failure probability can also be performed by manipulating sets of design variable values.

2.3 Design of the Genetic algorithm

GAs has been initially introduced by Holland J. (1975). They are stochastic global search methods which can be used in a wide variety of problems. They are based on the mechanics of natural genetic variation and natural selection. Thus terminology used in GAs is analogous to biological systems. For example, *strings* that are used in the optimisation algorithm are analogues to *chromosomes* in biological systems. *Genes* form *chromosomes* and are located at particular *locus* (positions) on the chromosome. A total genetic package of an organism is called *a genotype*. Analogically in a GA *variables* correspond to genes and a total package of strings forms *a structure*.

A classical single objective GA can be divided into 5 main stages: initialisation, reproduction, crossover, mutation, and replacement. All these parts of the algorithm can be modified according to the particular problem that is to be solved.

The introduced can be briefly introduced as following. The number of chromosomes that constitute a population (N) is denoted by the user. The size of the population remains constant throughout the optimisation process. At the initialisation stage a feasible population is generated. To produce a new population of chromosomes at first the three genetic operators are implemented by utilizing two nonoveralpping populations, i.e. parent and offspring populations are stored separately. At the next step evaluation and penalisation of offspring chromosomes is performed which is followed by the replacement procedure. As a result a new population is generated comprising chromosomes from the offspring population and in some cases a number of chromosomes from the parent population may also be included. At this stage one generation cycle is completed. The new generation cycle is stared with three genetic operators where the new population becomes a parent population. However if the number of performed generation (n) is equal to a predefined maximum number of generation the algorithm is terminated.

The implemented GA is summarised by the flowchart in Figure 2. Each stage of the algorithm is discussed in detail in the following sections.

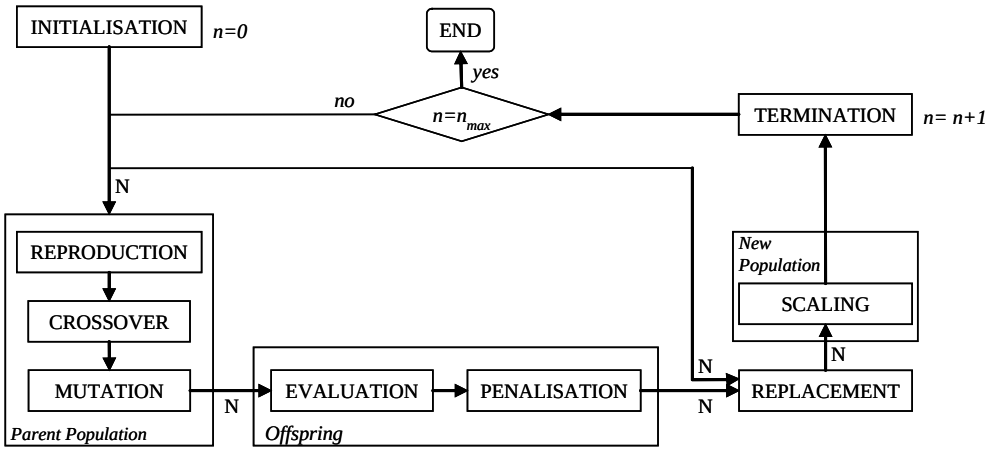

Fig 2: The GA Flowchart

2.3.1 Chromosome Encoding

Generally a chromosome is an encoded set of decision variables. However in the discussed case the used objective function is not in explicit form and a number of variable sets can be considered as decision variables for the problem. Due to the fact that a number of design variables is different to a total number of system components identifying a certain system design, coding of these sets of variables as chromosomes would effect a size of search space. Depending on the choice either fixed or variable length chromosomes would be used. Therefore it was decided to encode design variables as chromosome genes. This indirect chromosome encoding influences both the development of other stages of the algorithm and its efficiency. First of all the genetic operators are implemented for fixed

size chromosomes. As a result it enables to simplify implementation of the algorithm with no effect to quality of optimisation results.

Binary numbers were used for encoding of genes. In a chromosome each gene is allocated a particular number of bits required to code a maximum value of an associated variable. It ensures that the size of a chromosome is sufficient to store any values of design variables and its size remains constant. As a result each chromosome can be represented as a set of concatenated integers in binary form.

2.3.2 Initialisation

In the proposed algorithms a feasible initial population is generated. It is implemented in the following way. Each chromosomes is generated at a time by random sampling the bits. The generated chromosome is decomposed into genes. The obtained binary numbers are converted into decimal design variable values which identify a certain system design. At the next step it is checked if amounts of resources required for the generated design system do not overcome predefine limits. If all constraints for resources are satisfied the chromosome enters the initial population. The process is repeated until N chromosomes enter the initial population. Here N denotes a size of the population.

2.3.3 Reproduction, Crossover and Mutation

Three operators are used to create a new offspring population. At first the reproduction operator is implemented employing a biased roulette wheel which was discussed in detail by Goldberg D.E. (1989). As a result $N/2$ couples of parent chromosomes enter into a mating pool. Strings of each couple are crossed over employing one-point crossover operator. For each couple an integer position p which satisfies the following condition $1 \le p < l$ is selected randomly, where l is the length of the strings. All characters of the first string starting with the $p + 1$ position are assigned to the second string and characters of the second string are assigned to the first string. As a result two new string are created. During the crossover process, a bit-by-bit mutation is also carried out. It means each bit of a string is changed from 0 to 1 or vice versa with the predefined probability.

2.3.4 Penalisation

A new offspring population is created as a result of reproduction, crossover and mutation. At this stage the new strings are decoded and their corresponding fitness values are evaluated. Since the algorithm is developed to solve a constrained optimisation problem a penalty application was chosen as an approach to deal with possible violations of problem constraints. The main idea of this methodology is to apply some type of penalty to solutions which violate any constraint, i.e. a fitness function is defined as the sum of the objective and penalty function values.

A penalty function proposed by Coit D.W. et. al. (1996) was employed in the algorithm:

$$F_p(\mathbf{x}) = \left(F_{all} - F_{feas}\right)\sum_{i=1}^{nc}\left(\frac{d_i(\mathbf{x}, B)}{NFT_i}\right)^{\kappa_i}. \qquad (6)$$

Here, F_{all} is the best unpenalised value of the objective function yet found, F_{feas} is the best feasible value of the objective function yet found, NFT_i denotes the near-feasibility threshold that corresponds to a given constraint i, $d_i(\mathbf{x},B)$ is the magnitude of the violation of a given constraint i for solution $\mathbf{x}$, κ_i denotes a user-specified severity parameter and nc is the total number of constraints set for the problem.

In the implemented algorithm the near-feasibility threshold was defined employing a formula which allows the penalty value to be adjusted according to the search history:

$$NFT_i = \frac{NFT_{oi}}{1 + 0.1 \cdot g} \tag{7}$$

Here NFT_{oi} represents the actual value of a constraint i and g denotes the generation number. Parameter κ_i was set to 2 in order to implement Euclidian distances between any infeasible solutions to the feasible region over all constraints.

2.3.5 Population management

After an offspring population is created and a fitness value is evaluated for each chromosome the algorithm undergoes replacement. At this stage members of both parent and offspring populations can be selected to form a new generation population which will become a parent population and is used to form the next generation chromosomes. The replacement procedure was implemented employing an algorithm described by Chambers L. (ed.) (2001). The idea of this algorithm is to replace a parent population with an offspring population. However if the best parent chromosome is fitter than the best offspring chromosome then it replaces the worst offspring chromosome.

2.3.6 Improvement to the Algorithm

In order to improve the performance of the algorithm, fitness scaling was introduced. Research show that it is especially valuable when small population GA are employed.

A linear scaling procedure proposed by Goldberg D.E. (1989) was implemented in the algorithm. Parameters used in the linear scaling procedure are problem-independent. They depend on a population life and are found for a population in each generation.

The linear scaling method defines a linear relationship between an initial fitness value and the fitness value after the scaling:

$$f_{scaled} = af_{initial} + b\,. \tag{8}$$

Here, $f_{initial}$ is an actual chromosomes' fitness, f_{scaled} is the chromosomes' fitness after scaling and parameters a and b are linear function coefficients. In the implemented method these coefficients are selected so that the average fitness before scaling and the average scaled fitness are equal.

In the algorithm the scaling procedure is performed following replacement as it is shown in Figure 2.

3. Application Example and Results

The proposed optimisation approach was applied to analyse a simplified unmanned aerial vehicle (UAV) mission. The aim of the optimisation was to identify modifications of UAV

design from the given list that would minimise failure probability of the whole mission. The resulting system design also had to satisfy the requirement that mission failure probabilities for each phase would not increase. However small violations were acceptable if the overall mission failure decreased.

The optimisation problem was solved for two cases of the optimisation approach. At first the initially introduced GA was employed. Following, the improved GA algorithm with implemented scaling procedure was used to find the optimal UAV design. The results are presented for both cases.

The UAV mission comprises of six phases carried out in the following order: take-off, climb, en-route in controlled airspace, en-route in uncontrolled airspace, descent and land. Given data for the introduced problem included failure properties of the UAV components, data of external factors that can cause mission failure, the fault tree for every phase and the list of possible UAV design alterations.

To perform the optimisation analysis five UAV components were chosen to be replaced. The list of design variables and their values selected to characterise improvements in UAV performance is provided in Table 1. The structure of chromosome constructed according to the provided data is shown in Figure 3.

Table 1: List of Design Variables

Changeable component	Description of modifications	Possible values of design variables
Landing gear	Type of a landing gear	*type1, type 2*
Antiskid valve	Number of feed antiskid valves	*2, 1*
Cross feed valve	Number of cross feed valves	*3, 2, 1*
	Type of a cross feed valve	*type1, type 2*
	Minimum number of valves required for successful operation	*3, 2, 1*
Pump	Number of pumps	*2, 1*
	Type of a pump	*type1, type 2*
	Minimal number of pumps required for successful operation	*2, 1*
Navigation system	Number of sub navigation systems	*2, 1*
	Type of a navigation system	*type1, type 2*

The introduced requirement of maintaining and if possible improving reliability of the mission throughout each phase for necessitates to include mission failure probability constraints. Therefore performance of the original, i.e. initial design UAV, was analysed in order to set the limits for UAV failure probabilities at each phase. The failure probability of the first phase for the initial design UAV was 0.0306. The second phase failure probability was 0.0174, the third phase failure probability was equal to 0.0135, the fourth phase failure probability value was 0.0086, the fifth phase failure probability increased up to 0.0299 and after the sixth increased to 0.0357. The probability that the initial design UAV would fail to complete the mission was 0.1356. The following values were used as the limits of constraints. The first phase failure probability limit was set to 0.031, the

second limit was set to 0.018 and correspondingly the remaining limits were set to 0.014, 0.009, 0.03 and 0.036.

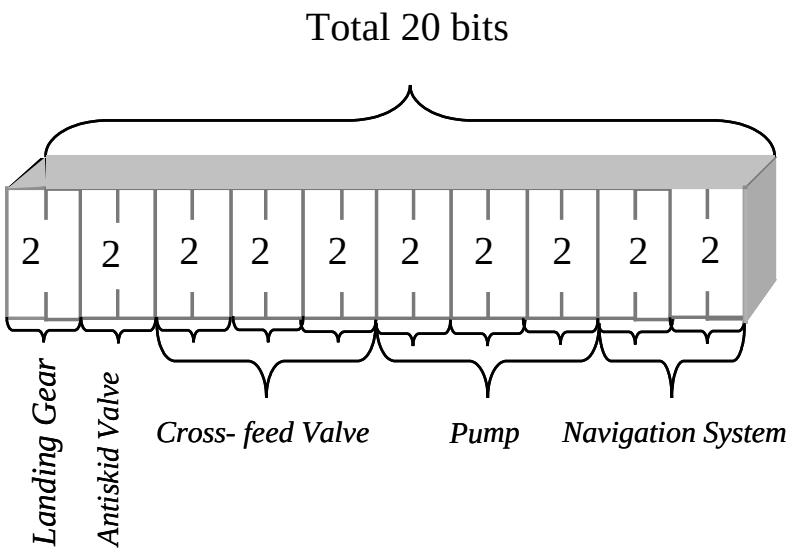

Fig 3: Chromosome Structure

GAs are guided search methods and the best values for the GA parameters are case dependant. The choice of GA parameter values for the UAV optimisation problem was based on the theoretical model discussed by Gibbs M.S. *et al*. (2008) and a trial-and-error approach. The best result, i.e. the smallest average number of generations required to find the minimal failure probability values was obtained when using a 50 chromosome population, a crossover rate equal to 0.95 and a probability of mutation equal to 0.05. Due to the stochastic nature of the GA five runs were performed to investigate the convergence of the optimisation results. Each time the process was terminated after 50 generations.

Obtained results presented in Figure 4 are the best fitness values in each generation. These results were obtained employing the GA without implemented scaling procedure. They show that the fitness values converge to the minimal value which is achieved on average after 35 generations.

Results presented in Figure 5 evidence the improvement of the optimisation algorithm after implementation of fitness scaling in the GA. The termination condition of 50 generations was reduced to 30 since the minimal value was achieved on average after 16 generations.

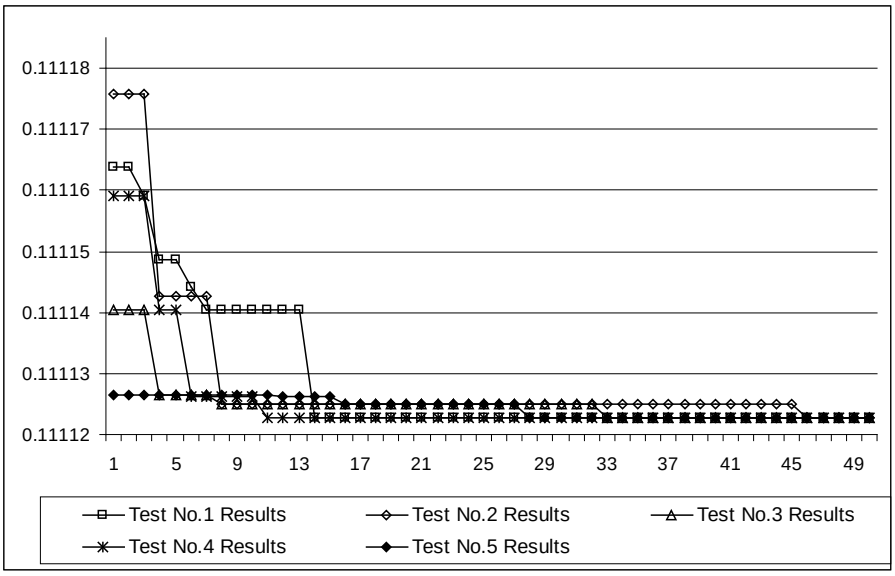

Fig 4: Minimal Mission Failure Probability Values for Each Generation

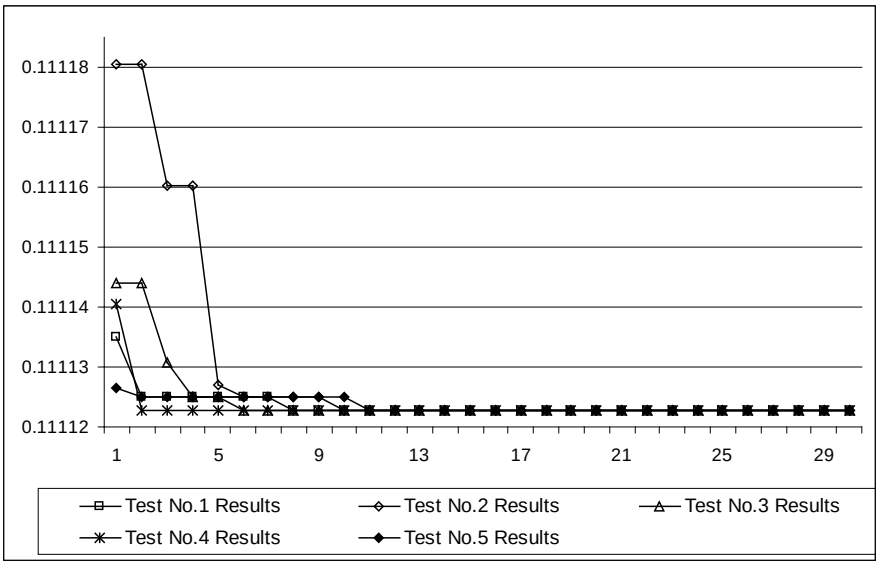

Fig 5: Minimal Mission Failure Probability Values for Each Generation (Scaling is Applied)

The minimal overall mission failure probability obtained from the model was equal to 0.1111176759, representing the failure probability for the optimal UAV design, while the mission failure probability of the initial design UAV was 0.1356551172. The optimally designed vehicle now includes new components that have replaced the some of the original components. The list of the new components is presented in Table 2.

Table 2: Values of Design Variables for the Optimal System Design

Changeable component	Design variable description	Design variable value
Landing gear	Type of a landing gear	*type1*
Antiskid valve	Number of feed antiskid valves	*2*
Cross feed valve	Number of cross feed valves	*3*
	Type of cross feed valves	*type 2*
	Minimum number of valves required for successful operation	*1*
Pump	Number of pumps	*2*
	Type of pumps	*type 1*
	Minimal number of pumps required for successful operation	*1*
Navigation system	Number of sub navigation systems	*2*
	Type of navigation systems	*type 1*

4. Conclusions

The introduced algorithm is employed to minimise phased mission system failure. The failure is minimised by defining a set of system components that constitutes an optimal system design and contributes to the improvement of system performance.

The introduced optimisation algorithm has been successfully applied to minimise a UAV mission failure probability. A set of UAV components characterising the optimal vehicle design was also identified. As a result, the optimal design UAV mission failure probability represented the global minimum of the optimisation process.

In the presented algorithm fault tree analysis was used to represent system failure modes during the whole mission for each system design. Fault tree and BDD approaches were also employed to evaluate the failure probabilities for each phase and the whole mission for different system designs. A single objective GA was chosen as an optimisation technique. It was considered two types of the GA: the GA without and with implemented fitness scaling. The obtained results showed the indisputable effect on the on convergence of the results. Fitness scaling improved the efficiency of the optimisation process.

Given the applicability of the method to the example mission, the next step would be analysis of more complicated phased mission system. Future work should also focus on improvement of the algorithm performance and its application for multi objective optimisation problems.

REFERENCES

Chambers L.D. (ed.) 2001. *The practical handbook of genetic algorithms.* Boca Raton, Fla: Chapman & Hall/CRC.

Coit D.W., Smith A.E. & Tate D. M. 1996. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing* 8 (2): 173-182

Esary, J.D. & Ziehms, H.1975. Reliability of phased missions. *Reliability and Fault-Tree Analysis*: 213-236

Gibbs M. S., Dandy G.C. & Maier H.R. 2008. A genetic algorithm calibration method based on convergence due to genetic drift, Information Sciences 178:2857-2869.

Goldberg D.E. 1989. *Genetic algorithms in search, optimization, and machine learning*. Wokingham: Addison-Wesley.

Holland J. 1975. *Adaptation in natural and artificial Systems*. USA: University of Michigan Press.

Kim K. & Park K. S. 1994. Phased-mission system reliability under Markov environment. *IEEE Transactions on reliability* 43(2): 301-309.

La Band R.A. & Andrews J.D. 2004 Phased mission modelling using fault tree analysis. *Proc. Instn Mech. Engrs, Part E: Journal of Process Mechanical Engineering* 218(2): 83-91.

Levitin G. 2006. Genetic algorithms in reliability engineering. *Reliability Engineering & System Safety* 91(9):975-976

Marseguerra M., Zio E. & Martorell S. 2006 Basics of genetic algorithms optimization for RAMS applications. *Reliability Engineering & System Safety* 91(9):977-991.

Ou P., Meshkat L. & Dugan J. B. 2002. Multi-phase reliability analysis for dynamic and static phases. *Proceedings of annual reliability and maintainability symposium*: 404-410.

Smotherman M. & Zemoudeh K. A non-homogeneous Markov model for phased-mission reliability analysis *IEEE Transactions on reliability* 38(5): 585-591

Susova G.M & Petrov A. N. 1997. Markov model-based reliability and safety evaluation for aircraft maintenance-system optimization. *Proceedings of annual reliability and maintainability symposium*: 29-36.

Torres-Echeverria A.C., Martorell S. & Thompson H.A. 2008 Design optimization of a safety instrumented system based on RAMS+C addressing IEC 61508 requirements and diverse redundancy. *Reliability Engineering & System Safety*.

Wang D & Trivedi K.S. 2007. Reliability analysis of phased-mission system with independent component repairs. *IEEE Transactions on reliability* 56(3): 540-551.