Genetic algorithms in optimal safety system design

Genetic Algorithm in Optimal Safety System Design

Rachel L. Pattison; Loughborough University, Loughborough

John D. Andrews: Loughborough University, Loughborough

SUMMARY

This paper describes a design optimization scheme for systems that require a high likelihood of functioning on demand. For safety systems whose failure could result in loss of life it is imperative that the best use of the available resources is made and a system which is optimal not just adequate is produced.

To demonstrate the practicalities of the method it has been applied to a High Integrity Protection System (HIPS). Analyses of individual system designs is carried out using the latest advances in the fault tree analysis technique, by utilising the more recent Binary Decision Diagram approach.

A Genetic Algorithm (GA) is used to perform the optimization resulting in the final design specification. Specific techniques are introduced to penalise the fitness of infeasible designs and to incorporate these values into the GA.

Parameters exist which affect the action of the GA. The latter part of the paper considers the effect of varying these parameters. A parameter combination is suggested which may achieve the most effective exploration of the search space and, thus, result in the best system design.

1. INTRODUCTION

Failure of a safety system for a potentially hazardous industrial system or process may have severe consequences, possibly injuring members of the workforce or public and occasionally resulting in loss of life. It is, therefore, imperative that such systems have a high likelihood of functioning on demand.

One measure of system performance is the probability that the system will fail to operate when necessary. Typically the design of a safety system follows the traditional design process of preliminary design, analysis, appraisal and redesign. If, following analysis, the initial design does not meet some predetermined acceptability target for system unavailability, deficiencies in the design are removed and the analysis and appraisal stages are repeated. Once the predicted system unavailability of a design reaches the acceptable criteria the design process stops and the system is adopted. For a system whose failure could result in fatality, an acceptable criteria for system unavailability is not sufficient. The aim should be to produce the optimal performance attainable within the constraints imposed on resources.

It is highly unlikely that the design parameters can be manually selected such that the optimal system performance can be achieved within the available resources. An approach by which optimal performance can be obtained using the fault tree analysis method to determine the availability of each system design, was described in a paper in 1994 (Ref. 1). An alternative methodology is presented in a later paper in 1997 (Ref. 2) which incorporates the latest advances in the fault tree analysis technique, using Binary Decision Diagrams (Ref. 3 - 7) and utilises a Genetic Algorithm (GA) (Ref. 8,9) to perform the optimization. This paper carries out an analysis of the genetic algorithm approach described in the 1997 paper and investigates the effect of altering some of the steps in the GA process.

2. SAFETY DESIGN CONSIDERATIONS

Safety systems are designed to operate when certain conditions occur and act to prevent or mitigate their development into a hazardous situation. Where possible, a safety system should not be designed such that single component failure can prevent the system from functioning. To ensure this redundancy or diversity can be incorporated into the system. Redundancy duplicates elements within a system while diversity involves the addition of a totally different means of achieving the same function

Component selection is a second design option. Each component selected for the design is chosen from a group of possible alternatives. The design engineer must decide how to trade-off the specific characteristics of each component to give the most effective overall system performance.

The time interval between preventative maintenance activities is a further consideration. This is generally assigned on an ad hoc basis. Significant gains can be made by considering the maintenance frequency at the design stage.

The choice of design is not, however, unrestricted. Practical considerations place limits on resources, which prevent a completely free choice of system design, rendering some design variations infeasible.

3. THE DESIGN OPTIMIZATION PROBLEM

The objective of the design optimisation problem is to minimise system unavailability by manipulating the design variables such that limitations placed on them by constraints are not violated. Commonly with mathematical optimisation problems, such as linear programming, dynamic programming and sequential unconstrained optimization (Ref. 10), there will be an explicit objective function which defines how the characteristic to be minimised is related to the variables.

In this problem an explicit objective function cannot be formulated. The system performance is assessed using a fault tree representing the design in question.

The nature of the design variables also adds difficulty to the problem. Design variables that represent the levels of duplication for fully or partially redundant systems and the number of weeks between maintenance activity are all integer. Selecting component types is governed by Boolean variables, i.e. selection or non-selection. A numerical scheme is, therefore, required that produces integer values for these variables since it will not be appropriate to utilise a method where real numbers are rounded to the nearest whole number.

Constraints involved in this problem fall either into the category of explicit or implicit constraints. The cost and maintenance downtime can be represented by an explicit function of the design parameters. However, the number of spurious trips can only be calculated via a full analysis of the system, which will again employ the fault tree analysis technique.

4. GENETIC ALGORITHMS

Genetic Algorithms are a robust class of optimization techniques that use principles mimicking those of natural selection and genetics. Each system design is coded as a string of parameter values. Each string being analogous to a chromosome in nature. The method then works with a population of strings.

The structure of the GA is that each string is assigned a measure of its fitness in the environment. Selection (or reproduction as it is also known) then exploits this fitness information. The greater the fitness value the higher the string's chance of being selected to enter the next generation.

The whole process is influenced by the action of the genetic operators typically crossover and mutation. These perturb the parameter information on each string and allow for greater exploration about the search space.

The basic method of selection allocates offspring to the next generation via a biased roulette wheel. Each string is assigned a certain percentage of the roulette wheel depending on the size of their fitness value in relation to the other strings in the population.

Crossover involves crossing information between two solution strings, already selected to enter the next generation, from some randomly determined crossover point. Mutation is the alteration of a specific parameter value on the solution string. Both operators enable exploration of different system designs.

5. SYSTEM ANALYSIS

5.1 Use of fault trees

As no explicit objective function exists fault trees are used to quantify the system unavailability of each potential design. It is, however, a time consuming, impractical task to construct a fault tree for each design variation.

To resolve this difficulty house events can be used to enable the construction of a single fault tree capable of representing causes of the system failure mode for each possible system design. House events in the fault tree, which are either TRUE or FALSE, are utilised to switch on or off different branches to model the changes in the causes of failure for each design alternative.

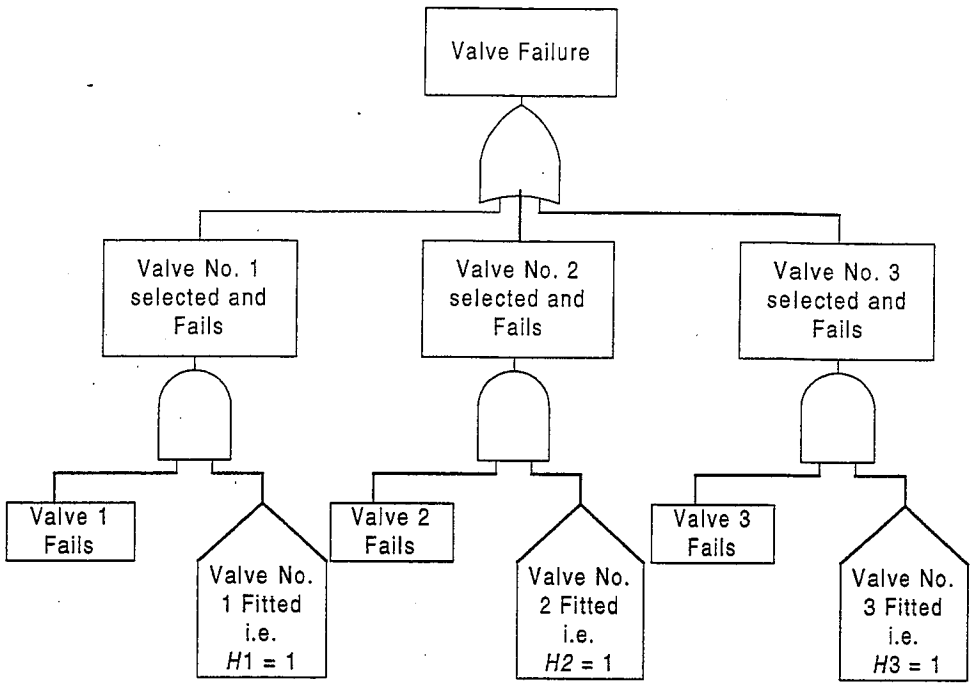

Figure 1. Fault Tree Structure for Component Select

Consider, for example, the choice of a valve type, V1,V2 or V3. The structure of the tree is as shown in figure 1. If valve type 1 is selected the house event, H1, corresponding to the selection of this valve is set to TRUE. House events H2 and H3, corresponding to the selection of valves 2 and 3 are conversely set to FALSE. A contribution to the top event arises from the left most branch only. The two right most branches are in effect switched off. Levels of redundancy are handled in a similar manner.

The spurious trip frequency for each design is an implicit constraint that requires the use of a fault tree analysis to assess its value. House events are again used to construct a fault tree capable of representing each potential design for this failure mode.

5.2 Use of binary decision diagrams

In order to improve efficiency the Binary Decision Diagram (BDD) method is used to solve the resulting Fault Tree. A BDD is a directed acyclic graph composed of terminal and non-terminal vertices, which are connected by branches. Terminal vertices have the value 0 or 1 and non-terminal vertices correspond to the basic events of the fault tree. Each vertex has a 0 branch which represents basic event non-occurrence (works) and a 1 branch which represents basic event occurrence (fails). Thus, all paths through the BDD terminate in one of two states, either a 1 state, which corresponds to system failure, or a 0 state, which corresponds to a system success. The BDD represents the same logical function as the fault tree from which it is developed. As an example consider the BDD illustrated in figure 2.

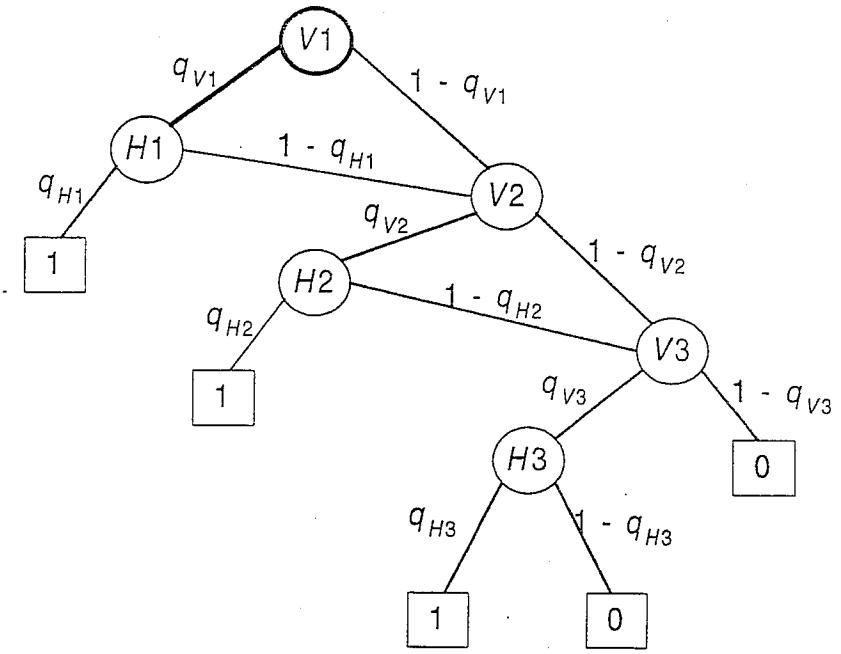

Figure 2. Component Selection BDD

Analysis of a BDD has proven to be more efficient than the quantification of the fault tree structure itself. In addition the BDD produces more accurate results.

The fault tree structures for each system failure mode are converted to their equivalent BDDs. For the purpose of BDD construction., where house events are encountered in the fault tree they are treated as basic events. Using this process the fault tree for the component selection design variables, shown in figure 1, can be represented by the BDD in figure 2.

The quantities appearing on the 1 and 0 branches developed from each node in figure 2 represent the probability of going down each path. The house events are turned on or off by setting their probability to 1 or 0 respectively. Consider, for example, the design where valve type number one has been selected for the fault tree shown in figure 1. This is presented by $H1 = 1$, $H2 = 0$, $H3 = 0$ in the design variables, and hence corresponding probabilities $q_{H1} = 1$, $q_{H2} = 0$ and $q_{H3} = 0$ are set on the equivalent BDD. The only path to a terminal 1 node leaves $V1$ and $H1$ on their 1 branches which has probability q_{V1} as required.

Quantification of the BDD can be carried out in the GA source code. The probability values assigned to each house event, determined by the design in question, are available within the package and hence, automatically assigned to the BDD. In terms of practicality this is the major advantage of the BDD.

6. EXAMPLE

As an example, the technique has been applied to the simple high-pressure protection system taken from (Ref. 2). The basic features of the high pressure protection system are shown in figure 3. Its function is to prevent a high-pressure surge passing through the system. In this way protection is provided for processing equipment whose pressure rating would be exceeded. The high pressure originates from a production well of a not normally manned offshore platform and the pieces of equipment to be protected are located downstream on the processing platform.

The first level of protection is to be the ESD (emergency shutdown) sub-system. Pressure in the pipeline is monitored using pressure transmitters (PTs). When the pipeline pressure exceeds the permitted value then the ESD system acts to close the Wing and Master valves on the well together with any ESD valves that have been fitted.

To provide an additional level of protection a second level of redundancy can be incorporated by inclusion of a HIPS (high-integrity protection system). This works in a similar manner to the ESD system but is completely independent in operation.

Even with a relatively simple system such as this there are a vast number of options for the designer to consider. In the example it is required to determine values for the design variables that represent the following:

Designer Options	Design Variable
• How many ESD valves are required (0, 1, 2)?	E
• How many HIPS valves are required (0, 1, 2)?	H
• How many pressure transmitters for each sub-system (0, 1, 2, 3, 4)?	N_1, N_2
• How many transmitters are required to trip?	K_1, K_2
• Which of two possible ESD/HIPS valves to select?	V_1, V_2
• Which of two possible pressure transmitters to select?	P_1, P_2
• Maintenance test interval for each sub-system (1 week – 2 years)?	θ_1, θ_2

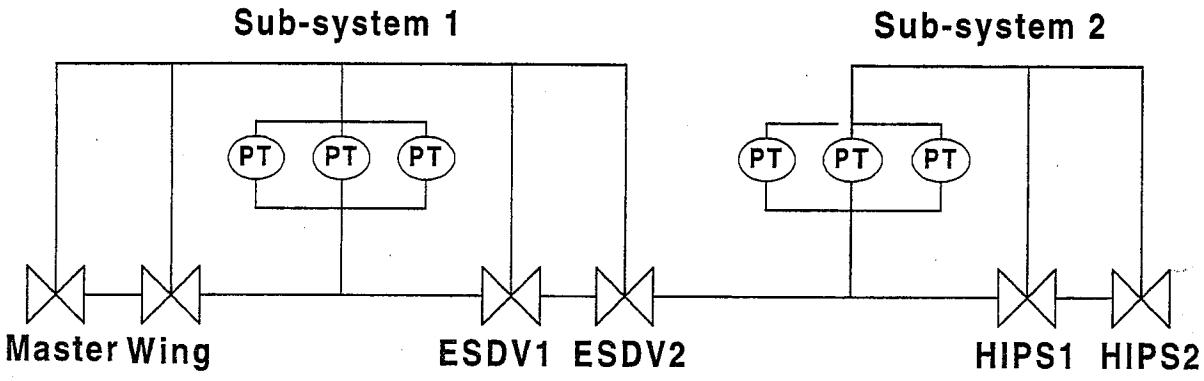

Figure 3. High-integrity protection system

Table 1. Component data

Dormant failures				
Component	**Failure rate**	**Mean repair time**	**Cost**	**Test time**
Wing valve	1.14×10^5	36.0	100	12
Master valve	1.14×10^5	36.0	100	12
HIPS valve 1	5.44×10^6	36.0	250	15
HIPS valve 2	1×10^5	36.0	200	10
ESDV valve 1	5.44×10^6	36.0	250	15
ESDV valve 2	1×10^5	36.0	200	10
Solenoid valve	5×10^6	36.0	20	5
Relay contacts	0.23×10^6	36.0	1	2
Pressure transmitter 1	1.5×10^6	36.0	20	1
Pressure transmitter 2	7×10^6	36.0	10	2
Computer logic	1×10^5	36.0	20	1

Limitations have been placed on the design such that:

1. The total system cost must be less than 1000 units. Hardware costs are given in table 1.
2. The average time each year that the system resides in the down state due to preventive maintenance must be less than 130 hours. Times taken to service each component at each maintenance test are also shown in table 1.
3. The number of times that a spurious system shutdown occurs would be unacceptable if it was more than once per year.

7. GENETIC ALGORITHM IMPLEMENTATION

SGA_C is a C-language translation and extension of the original Pascal Simple Genetic Algorithm (SGA) code presented by Goldberg (Ref. 8). This package was used as a framework to build the genetic algorithm software for the safety protection system optimization. Significant changes and extensions were necessary to carry out the modelling of the HIPS optimization.

7.1 Coding and initialising the population

Each solution string represents a particular system design depending on the values assigned to each of its 10 parameters. Each parameter must be allowed a particular length of the string, i.e. a particular number of bits, in order to accommodate its largest possible value in binary form. For example θ_1, the parameter governing the maintenance test interval for subsystem 1, requires 7 bits to accommodate its maximum time span of 104 weeks. In total each string representing all design variables is 32 bits in length and can be interpreted as a set of concatenated integers coded in binary form.

The restricted range of values assigned to each parameter does not in each case correspond to the representative binary range on the solution string. For this reason a specialized procedure is used to code, initialize and, in subsequent generations, check the feasibility of each string.

7.2 Evaluating string fitness

Constraints are incorporated into the optimization by penalising the fitness when they are violated by the design. The fitness of each string is comprised of 4 parts:

- Probability of system unavailability, Q_{SYS},
- penalty for exceeding the total cost of the string,
- penalty for exceeding the total maintenance down-time constraint
- penalty for exceeding the spurious trip constraint.

The result is a sole fitness value for each design, referred to as the penalised system unavailability of the design.

Calculating the penalised system unavailability involves the derivation of penalty formula for excess cost, maintenance down time (MDT) and spurious trip occurrences. If a particular design exceeds any of the stated limits the respective penalty is added to the system unavailability of the design in question.

$$Q'_{SYS} = Q_{SYS} + C_{PEN} + MDT_{PEN} + ST_{PEN}$$

Where,

Q'_{SYS}	=	penalised probability of system unavailability
Q_{SYS}	=	unpenalised probability of system unavailability
C_{PEN}	=	the penalty exerted due to excess cost
MDT_{PEN}	=	the penalty due to excess MDT
ST_{PEN}	=	the penalty due to excess spurious trips

7.3 Reproduction probabilities

The fitness value, or penalised system unavailability, is evaluated for each string. For the purpose of selection in the GA, each string is assigned a reproduction probability which is directly related to its fitness value.

In the safety system optimization problem the smaller the fitness value, the fitter the string and hence, the greater should be its chance of reproduction. For cases such as these a possible approach is to let the reproduction probability be one minus the fitness value. However, using a string's availability produces all reproduction probabilities of a similar value, thus detracting from the fitness information available to the GA. A more specific method is required which retains the accuracy of each string's fitness value during conversion to its corresponding reproduction probability. Further discussion off this method is considered in a later section.

8. RESULTS

The program was used with a population of 10 strings. A maximum of 50 generations was allowed along with a mutation rate of 0.01 and crossover rate 0.7. In total, therefore, 500 system evaluations were performed in determining the best design. The running time of the program was an order of minutes.

The fittest string from the entire process arose in generation 15. The characteristics of this design are specified in table 2.

Table 2. Characteristics of the best design

Subsystem 1	No. of ESD valves	0
	No. of PTs	3
	No. of PTs to trip system	2
	M.T.I.	23
Subsystem 2	No. of HIPs valves	2
	No. of PTs	3
	No. of PTs to trip system	2
	M.T.I.	57

Valve type	2
PT type	1

MDT	123 hours
Cost	842 units
Spurious trip	0.455

System Unavailability	0.001102

Convergence to a fit design through the GA is not necessarily smooth. A particularly fit string may be produced in an early generation as a result of its random nature, else the structure of the GA may enable uniform convergence to a fit string over later generations.

1. THE SGA PARAMETERS

The GA requires the following selection parameters to be set:

- population size,
- crossover rate,
- mutation rate,
- number of generations.

Values entered for these parameters have a marked affect on the action of the GA. Using the SGA previously described, an analysis was carried out to investigate the effect of changing these parameter values. De Jong, amongst others, has researched this area with a test bed of five trial functions, and two criteria of goodness defined (Ref. 8). Regarding his concluding discussion a limited set of values for each parameter was chosen:

Mutation rate	:	0.1	0.01	0.001	
Crossover rate	:	0.5	0.6	0.7	0.8
Population size	:	10	20	50	

In total 36 runs of the SGA were carried out, thus ensuring each possible combination of the values above was analysed. The penalised system unavailability of the best overall string per run was then investigated for each parameter set.

To obtain an indication of the effect of setting each parameter to a particular value the best penalised system unavailability obtained for all of the other parameter values were summed and averaged. A summary of these results are given in table 3, for the mutation rate, crossover rate and population size respectively.

Table 3. A summary of the quantitative results for each parameter

	MUTATION RATE		
	0.1	**0.01**	**0.001**
SUM OF FITNESS VALUES	0.015031	0.017101	0.018992
AVERAGE FITNESS VALUE	1.25E-3	1.42E-3	1.58E-3

	CROSSOVER RATE			
	0.5	**0.6**	**0.7**	**0.8**
SUM OF FITNESS VALUES	0.01253	0.013641	0.013	0.011953
AVERAGE FITNESS VALUES	1.044e-3	1.37e-3	1.08e-3	9.96e-4

	POPULATION SIZE		
	10	**20**	**50**
SUM OF FITNESS VALUES	0.02093	0.016642	0.013552
AVERAGE FITNESS VALUE	1.74e-3	1.39e-3	1.13e-3

9.1 Discussion of quantitative results

As might be expected, larger populations lead to a better performance. When the population size doubles from 10 to 20 strings the fitness value improves by 20 percent. An additional 18 percent improvement is incurred when the initial population is further increased to 50 individuals.

The mutation rate parameter implies that the largest rate, 0.1, leads to the generation of a fitter string. Strings produced in each run of the program with the largest mutation rate were on average 20 percent fitter than program runs with the lowest rate.

The crossover parameter again produced a slight bias toward the highest value.

These results appear to support a more random search. It is interesting to note that as the population size increases the beneficial effect of a larger mutation and crossover rate is diminished. A possible deduction from this is that the mutation and crossover rates do not have a significant effect if the population size is sufficiently large. A population of only10 strings may not incorporate enough diversity from the onset and for this reason a high degree of mutation moves to areas in the search space which would otherwise not be explored.

10. CONCLUSION

The use of house events and BDDs for system analysis proved successful. In addition the GA was able to cope with all the requirements of the design problem and the approach successfully tested on the High Pressure Protection System.

A large population ensures that a large pool of parameter values is introduced from the onset. This enables a greater area of the solution space to be simultaneously searched. This is, however, at the expense of a greater number of function evaluations.

A balance between the diversity and processing time was made to obtain the best results and it is suggested that a high population size is selected together with a crossover rate of 0.7 and mutation rate of 0.01.

REFERENCES

1. J.D. Andrews, T.R. Moss, "Reliability and Risk Assessment", 1993; Longmans.
2. J.D. Andrews, R.L. Pattison, "Optimal Safety-system Performance", *Proceedings of 1997 Reliability and Maintainability Symposium*, Philadelphia, Jan 1997, pp 76- 83.
3. A. Rauzy, "New Algorithms for Fault Tree Analysis", Reliability Engineering and Safety System, Vol 40, 1993, pp 203-211.
4. R.M. Sinnamon, J.D. Andrews, "Fault Tree Analysis and Binary Decision Diagrams", *Proceedings of 1996 Reliability and Maintainability Symposium*, Las Vegas, Jan 1996, pp 215-222.
5. R.M. Sinnamon, J.D. Andrews, "New approaches to Evaluating Fault Trees", *Proceedings of ESREL 95 conference*, June 1995, pp 241-254.
6. R.M. Sinnamon, J.D. Andrews, "Improved Efficiency in Quantitative Fault Tree Analysis", *Proceedings of 12th ARTS*, Manchester, April 1996.#
7. R.M. Sinnamon, J.D. Andrews "Improved Accuracy in Quantitative Fault Tree Analysis", *Proceedings of 12t ARTS*, Manchester, April 1996.
8. D.F. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", 1989; Addison-Wesley.
9. L. Painton, J. Campbell, "genetic Algorithms in Optimization of System Reliability", *IEEE Trans.* Rel, Vol. 44, No2, June 1995, pp172-178.
10. F.A. Tillman, C. Hwang, W. Kuo, "Optimization of Systems Reliability", 1980;Marcel Decker.