Optimal Safety System Performance

John D. Andrews; Loughborough University, Loughborough
Rachel L. Pattison; Loughborough University, Loughborough

Key Words: Fault trees, Binary decision diagrams, Optimization, Genetic algorithms

SUMMARY

This paper investigates the efficiency of a design optimization scheme which is appropriate for systems which require a high likelihood of functioning on demand. Traditional approaches to the design of safety critical systems follows the **preliminary design, analysis, appraisal** and **redesign** stages until what is regarded as an acceptable design is achieved. 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.

The methodology presented in the paper retains the commonly used fault tree method to analyse the individual system designs. By the use of house events a single fault tree is constructed to represent the failure causes of each potential design to overcome the time consuming task of constructing a fault tree for each design investigated during the optimization procedure.

The final design specification is achieved using a genetic algorithm to perform the optimization with the constraints incorporated by penalising the fitness of infeasible designs. To demonstrate the practicality of the method developed it has been applied to a High Integrity Protection System (HIPS).

1. INTRODUCTION

For safety systems installed on potentially hazardous plant it is essential that they have the maximum likelihood of working on demand possible for the resources available This is particularly true when the hazard could kill or injure members of the workforce or the public.

Techniques such as fault tree analysis (Ref. 1), networks (Ref. 2), Markov analysis (Ref. 3) and simulation (Ref. 4) are now commonly used for system availability assessment. However, as with many other engineering disciplines they are used in a traditional design process of preliminary design, analysis, appraisal and redesign. The initial design specification is analysed to predict its likelihood of failure to perform according to the design intention. Its predicted performance is then compared to that which is considered acceptable. The criterion used to determine the adequacy of the design is usually a comparison with a pre-determined target figure for its availability. If the system performance is not acceptable then deficiencies in the design are removed and the analysis and appraisal stages repeated. When the predicted system performance is regarded as adequate the design process stops and the design is adopted.

It is highly unlikely that the design parameters can be manually selected such that the optimal system performance is 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. 5). The methodology presented in this paper improves the efficiency of that used in the 1994 paper by incorporating the latest advances in the fault tree analysis technique using Binary Decision Diagrams (Refs. 6 - 10) and a Genetic Algorithm (Ref. 11) to perform the optimization. Genetic Algorithms have been shown to provide good results for reliability problems where distributions represent the component failure rates (Ref. 12).

2. SYSTEM DESIGN CONSIDERATIONS

Safety systems are designed to operate when certain conditions occur and act to prevent their development into a hazardous situation. As such there are certain features common to all safety protection systems. All safety systems have sensing devices which monitor for the occurrence of the triggering events. These sensors usually measure some process variable and transmit its current level to a controlling device. The controlling device determines whether the current situation is acceptable by comparing the input signal to a set point. When the sensed variable violates the set point the protective action is initiated. The protective action may either prevent a hazardous situation occurring or reduce its consequences.

The design engineer has a number of choices to make regarding the structure and operation of the safety system which can influence reliability. These design options are described below.

2.1 Redundancy and diversity levels

The safety system must be designed to have a high likelihood of working on demand. Thus single component failure should not be able to prevent the system from functioning. One means of achieving this is by incorporating redundancy or diversity into the system structure. Redundancy duplicates elements within a system while diversity involves the addition of a totally different means of achieving the same function. Both redundancy and diversity can be used at component level or sub-system level.

0-7803-3783-2/97/$5.00 © 1997 IEEE

Increased levels of duplication in the form of fully redundant and fully diverse elements will also increase the number of spurious systems trips. To counteract this, partial redundancy is commonly utilised where k sensors of the n sensors fitted have to indicate the trip condition for action to be taken.

2.2 *Component selection*

Each component selected for the design will be chosen from a group of possible alternatives. For every valve, relay, pressure sensor, etc., for which a selection is to be made there will be several choices, each with associated characteristics such as failure rate in each failure mode, cost and time taken for their scheduled maintenance. The design engineer has to decide how to trade-off these characteristics to give the most effective option for the overall system performance.

2.3 *Maintenance interval*

Generally the time interval between preventive maintenance activities is assigned on an *ad hoc* basis. This is now changing with the more frequent application of reliability centred maintenance. Even so, the allocation of available maintenance effort is only considered after the system design has been finalised. Since component selection determines the time taken to maintain the system there are significant gains to be made by considering the maintenance frequency at the design stage.

2.4 *Limitations on the design choices*

There are many options open to the design engineer. To produce the most effective system these parameters need to be selected to give optimal system performance. The choice of design is not, however, unrestricted. Some of the possible design variations will not be feasible. Practical considerations of limits placed on resources will prevent a completely free choice of system design. Such considerations may include cost, limited maintenance effort, system weight, space limitations and other requirements of the system performance such as limits to the spurious trips occurrence rate.

3. *OPTIMIZATION SCHEME*

There are many mathematical optimization methods available. The application of methods such as linear programming, dynamic programming, non-linear programming and sequential unconstrained optimization (SUMT) to reliability problems is described by Tillman (Ref. 13). However, the features offered by the methods make them inappropriate for the type of problem considered in this paper.

Many of the optimization methods require an explicit function (objective function) which defines how the characteristic to be minimized is related to the design variables. A variation in some of the design variables, such as redundancy levels, gives a discrete change in the structure of the system and prevents an objective function being deduced.

Design variables which represent the levels of redundancy for a component are integer. If partial redundancy is incorporated through the use of voting systems, the number of successful channels to give the trip condition also needs to be chosen. This too is an integer variable for the design.

When component types are selected there will be a choice from several options which fulfil the same function. For example equipment supplied from different manufacturers. A variable corresponding to each potential piece of equipment can take the value 1 to indicate its selection and 0 to represent non-selection. This Boolean variable is again an integer over a restricted range. The optimization scheme must be appropriate for integer variables.

Constraint forms which are linear or non-linear functions of the design variables need to be incorporated in a general solution scheme. Constraints which determine aspects of the system performance (such as limitations on the expected number of spurious trips) which can only be evaluated by a full analysis of each potential design (implicit constraints) may also be specified.

Recent algorithms such as simulated Annealing and Genetic Algorithms (Ref. 11) provide alternative methods which can cope with the features of design optimization. Any algorithm which is used to optimize the availability of a safety system must possess the following characteristics:

1 Efficient analysis routine to quickly evaluate a large number of potential designs.
2. An optimization strategy which:
   (a) does not require an explicit objective function,
   (b) is appropriate for integer variables,
   (c) can cope with implicit and explicit constraint forms of equality or inequality type.
   (d) does not require linear forms of explicit constraints.

4. *SYSTEMS ANALYSIS*

Whichever optimization scheme is employed (but particularly for Genetic Algorithms) the analysis of a large number of potential designs is required. The most effective means of assessing the availability of safety systems is fault tree analysis. However, it is a time consuming task to construct and analyse a fault tree for each potential design. Manual construction of the fault tree for each design would make the optimization scheme impractical and computer construction processes are not yet developed to the required standard. This difficulty can be resolved by constructing a single fault tree to represent all possible design variations. 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 system failure for each design alternative. The house event structures are used as defined below for the different design variables:

4.1 *Component type selection*

Consider a component type which is to be selected such as a valve. If there are three potential valves to select from, then define the design variables $H1$, $H2$ and $H3$ take the value 1 if the appropriate valve is selected and 0 if it is not. If $H1$, $H2$ and $H3$ are non-negative integers then the following constraints need to be incorporated in the optimization problem definition. $H1 \leq 1$, $H2 \leq 1$, $H3 \leq 1$, $H1 + H2 + H3 = 1$. The

cost of each component and the time required for their maintenance may also need to be incorporated into other potential constraints. The fault tree structure for failures of the valve, whichever is selected, is shown in figure 1.

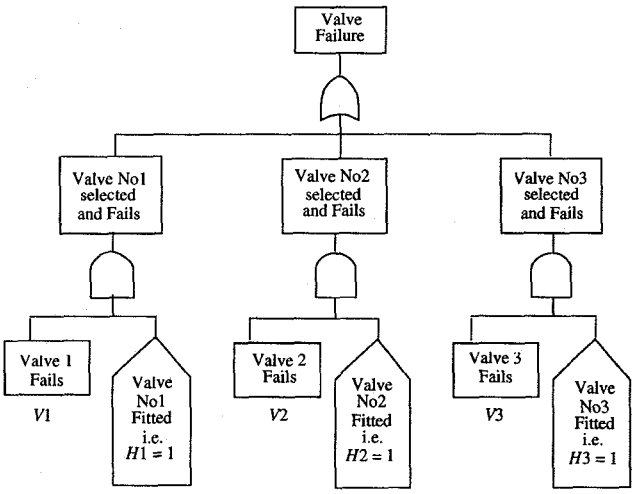

Figure 1. Fault Tree Structure for Component Selection

4.2 Redundancy Levels

For a sub-section or component which may be duplicated in the system design to provide redundancy, an integer design variable may be set to indicate the number of duplications. For example, if N represents the number of pressure transmitters on a system then for a failure of this sub-section to result in system failure it would need to affect all fitted channels. The fault tree structure can again be constructed for each design option, using house events as shown in figure 2 where a maximum of three levels of redundancy is allowed.

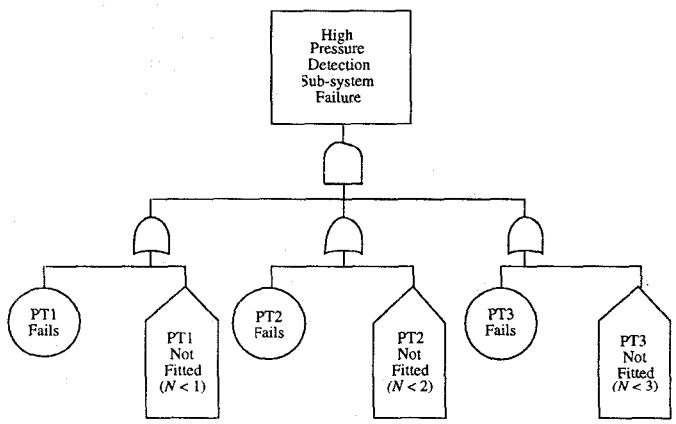

Figure 2. Fault Tree Structure for Redundancy Levels

The constraints $1 \leq N \leq 3$ would be added to the problem constraint list and depending on the value of N one or more house events would be set.

Changing the time interval between inspections will increase the probability of the existence of dormant, unrevealed component failures. Variation in this design parameter will change the probability of occurrence of the relevant basic events in the fault tree.

5. BINARY DECISION DIAGRAMS

During the optimization phase of the design scheme it will be required to derive the system failure probability for a large number of potential designs. To ensure that this can be achieved in the most efficient manner the fault tree structure representing the failure of all system designs will be converted to a Binary Decision Diagram (BDD). Analysis of BDDs has been shown to be much faster then the quantification of the fault tree structure itself.

A BDD is a directed acyclic graph. 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. All the paths terminating in a 1 state give the cut sets of the fault tree. A BDD is 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). As an example BDD consider figure 3.

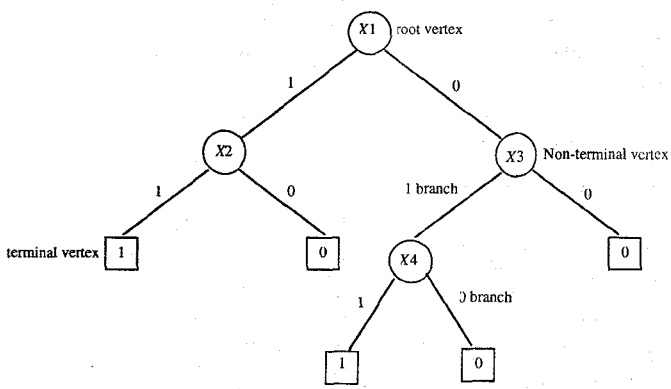

Figure 3. Example BDD

All the left hand branches leaving each vertex are the 1 branches and all the right hand branches are the 0 branches.

Every path starts from the Top basic event, called the root vertex, and proceeds down through the diagram to the terminal vertices. Only the vertices that lie on a 1 branch on the way to a terminal 1 vertex are included in a path which represents the cut sets. For example the paths, or cut sets, of the BDD shown in figure 3 are:

$X1.X2$ (1)

$X3.X4$ (2)

The BDD represents the same logical function as its equivalent fault tree.

Approaches to construct and quantify the BDD are given in detail elsewhere (Refs. 6-7). Where House events are encountered in the fault tree they are treated as Basic events for the purpose of BDD construction. Using this process the fault tree shown in figure 1 for the component selection design variables can be represented by the BDD shown in figure 4 (using an ordering of basic/house events.

$V1 < H1 < V2 < H2 < V3 < H3$).

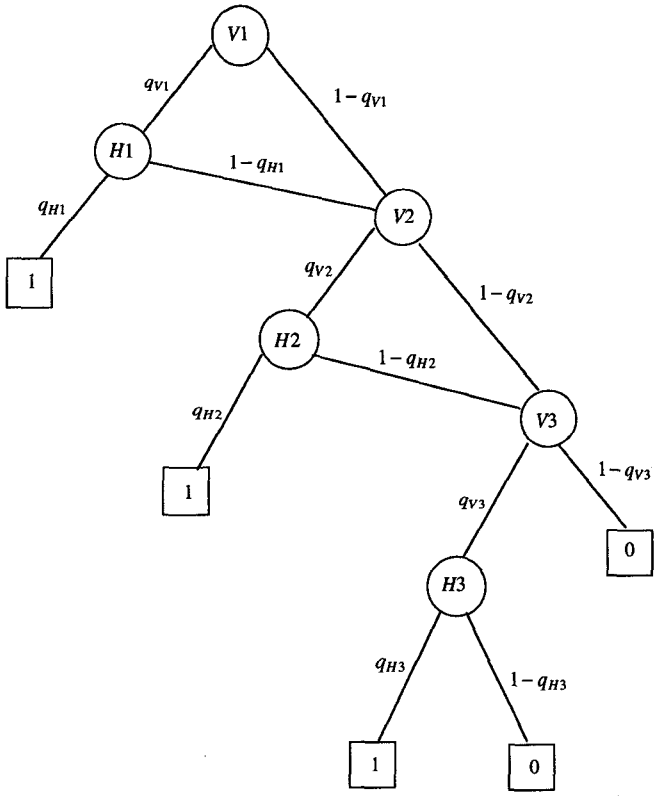

Figure 4. Component Selection BDD

The quantities appearing on the 1 and 0 branches developed from each node in figure 4 represent the probability of going down each path (component failure, component success). The probability of each path to a terminal one node can be obtained by taking the product of the individual probabilities encountered in travelling along the path. Since each of these paths represents a mutually exclusive set of events the system failure probability is determined by summing the probability of each such path to a terminal 1.

The house events are turned on and off by setting their probability to one or zero respectively. For example consider 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. Therefore considering probabilities $q_{H1} = 1, q_{H2} = 0$ and $q_{H3} = 0$ on the BDD shown in figure 4. If q_{H2} and q_{H3} are zero the only paths which can exit from nodes $H2$ and $H3$ are along the zero branches which only lead to a terminal 0. The only path to a terminal 1 node leaves nodes $V1$ and $H1$ on their 1 branches which has probability q_{V1} as required.

The representation of redundancy levels by house events is manipulated in the same way. This means that in converting the fault tree to a BDD all design variables are adjusted by altering node event probabilities.

6. GENETIC ALGORITHMS

Genetic Algorithms are a class of optimization procedures that use principles mimicking those of natural selection and genetics.

Genetic algorithms require the parameter set of an optimization problem to be coded as a finite-length string over some finite alphabet. These strings are analogous to chromosomes in nature and the ensemble of possible values the parameter set can take is comparable to a gene pool.

The genetic algorithm initialises and maintains a population of individuals, i.e. strings, over successive generations which evolve according to rules of selection. Each individual in the population receives a measure of its fitness in the environment. Reproduction then focuses attention on highly fit individuals, by exploiting the available fitness information.

The whole process is influenced by the action of genetic operators, typically this consists of "genetic" crossover and mutation, though others have also been used. These operators perturb individual strings providing potential for exploration around the gene pool.

Crossover involves mating two individuals, selected to enter the next generation, by crossing parts of their solution strings. Consider, for example, two strings:

$$S1 = 011010 \qquad S2 = 101100$$

If the crossover point is chosen to be 3; the strings used in the next generation are:

$$S1' = 011100 \qquad S2' = 101010$$

Mutation is generally the random alteration of a parameter value on the solution string. As regards a binary coded string this simply means changing a 1 to a 0 and vice versa.

Reproduction and crossover effectively explore the search space. Occasionally, however, they may become overzealous and loose some potentially useful 'genetic material'. Mutation helps maintain sufficient diversity in the population.

Genetic algorithms are robust in application to a wide variety of problem domains, including high dimension, stochastic problems, with many nonlinearities and discontinuities. They differ from traditional optimization techniques in that they work with a coding of the parameter set and not the parameters themselves. They simultaneously search a population of points in the solution space and use objective, i.e. the fitness, rather than auxiliary information, such as gradients.

7. EXAMPLE

As an example, the technique has been applied to the simple high pressure protection system taken from (Ref. 5). The basic features of the high pressure protection system are shown in figure 5. 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 vessels 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 the 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 considers. In this 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

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.

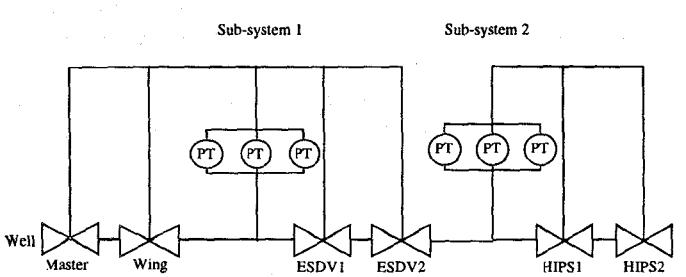

Figure 5. High-integrity protection system (HIPS)

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

8. GENETIC ALGORITHM IMPLEMENTATION

SGA_C is a C-language translation and extension of the original Pascal SGA, simple genetic algorithm, code presented by Goldman (Ref. 11). This package was used as a framework to build the genetic algorithm software package for the safety protection system optimization. The user may choose parameters governing population size, number of generations, crossover and mutation rate. The following is a discussion of the significant changes and extensions necessary to carry out the modelling for the HIPS optimization.

8.1 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.

Initialising random strings of 1's and 0's directly may create non-feasible system designs. For example, consider N_1, the parameter governing the number of pressure transmitters for subsystem 1. The design can accommodate 0 through to 4 pressure transmitters, thus 3 bits of the string are allocated to accommodate this parameter. Random initialisation of these 3 bits of the string can produce eight different combinations, i.e. corresponding to 0 through to 7 in denary form. The combinations 0 to 5 are feasible, however, the other 3 are not. To overcome this the initialisation procedure uses two routines to create the binary strings in two stages.

The first function initialises the string in integer form parameter by parameter using a random number function which only generates numbers between specific stated ranges. These integer values for each string are stored in data structures.

A second function than inserts each initialised integer value, in its binary form into the correct area of the 32 bit string. This cycle is carried out as many times as required to create a population of M feasible individuals.

8.2 Evaluating string fitness

A simple explicit objective function to calculate the fitness of each string does not exist. The fitness of each string comprises 3 parts;

- Probability of system unavailability, QSYS,
- penalty for exceeding the total cost of the string,
- penalty for exceeding the total maintenance down-time constraint.

Calculating each string's cost and maintenance downtime (MDT) is fairly straightforward. Both these constraints do use an explicit formula with data values.

The probability of system unavailability proved more difficult. This involved utilising a BDD, incorporated into the Genetic Algorithm software package. By entering the specific probabilities of each basic event for a particular string, this method is capable of representing the causes of dormant failure for each possible system design.

String fitness must, however, be represented in the SGA as one value only. This value is then used to determine which members of the population will be reproduced into the next generation. The probability of system unavailability is the value in which we are ultimately interested, but as mentioned before the sting must not exceed certain design limitations. For this reason the values for cost and MDT must be used to penalise the probability of system unavailability if, and only if, either exceed their limits, i.e.,

$$\text{cost} > 1000 \text{ units}, \qquad \text{MDT} > 130 \text{ hours}.$$

Penalty formulae were thus derived for both cost and MDT.

During MDT components in the safety system are being inspected so the safety system is unavailable. Hence MDT above 130 hours per year directly affects the unavailability of the system. 130 hours or below is feasible and incurs no penalty. If the MDT of a particular string exceeds 130 hours;

$$Q_{SYS}' = Q_{SYS} + (\text{MDT} - 130)/8760 \quad \text{for MDT} > 130.$$

Where:

Q_{SYS} = unpenalised probability of system unavailability

8760 = hours/year

Q_{SYS}' = penalised probability of system unavailability

It is relatively easy to prescribe a penalty for exceeding this constraint since it directly affects the unavailability.

The second constraint is cost which must be less than 1000 units. It is difficult to fix a penalty value for exceeding this constraint. There are many approaches which could be used. It is assumed here that a direct relationship exists between cost and performance. If cost exceeds its limit by 100 units, i.e. 10 percent, we would expect an increase in performance by the particular design of at least 10 percent.

As an example a theoretical system with a probability of system unavailability equal to 0.02 and cost 1000 units was considered. The value 0.02 is accepted as a reasonable value for system unavailability. If the cost of the system increases to 1100 units, we would expect that the probability of system unavailability would decrease to 0.018. This relationship is linear. String designs with excessive cost will not be adopted and must be heavily penalized. This implies the use of an exponential relationship would be more appropriate. Hence the following form is assured; $y = x^{5/4}$. Consequently the penalty function for excess cost was chosen to be;

$$\text{COST PENALTY} = (\text{EXCESS COST} / 1000)^{5/4} \times 0.002$$

and hence,

$$Q_{SYS}' = Q_{SYS} + \text{COST PENALTY}$$

N.B. The value of 0.002 comes from 10% of the value 0.02 chosen initially as a reasonable estimate of system unavailability.

8.3 Converting string fitness

As stated earlier the main purpose of the fitness value is to establish which strings will be reproduced in the next generation.

In the safety system optimization problem the smaller the fitness value the fitter the string and hence, the greater its chance of reproduction. For cases such as these a possible approach is to use its reproduction probability as one minus the fitness value. This, however, causes problems. Most final system unavailability values lie between 0 and 0.1. One minus these values results in the majority of numbers around 0.95. This method does not give a high enough priority to the fitter strings. Hence, a more specific method to convert the fitness values to a reproduction probability was derived.

Predominantly only those strings with a final system unavailability of less then 0.1 are of interest. For this reason any string with an unavailability greater than 0.2 is immediately eliminated, i.e. given a fitness percentage of 0, so that they have no chance of being chosen for the next generation. Strings with an unavailability between 0.1 and 0.2 are assigned reproduction probabilities which take up a total of 5% of the chance of being selected for the next generation. The main reason for this being to combat premature convergence in that they have a chance to be chosen for the next generation, all

be it very small. This results in the fittest strings occupying 95% of the wheel.

The fitness values are now in a state appropriate for random selection using the reproduction probabilities.

9. *RESULTS*

The program was used with a population of 10 strings. A maximum of 50 generations was entered, along with a mutation rate of 0.005 and crossover rate 0.7. Thus, in total, 500 results were obtained over the 51 populations analysed. The running time of the program was an order of minutes.

From the results of the program two important factors were considered, the average string fitness and the best string fitness in each generation. These are shown in figures 6 and 7 respectively.

The initial population produced a set of strings with string fitness values ranging between 1.0 and 0.0052. Over successive generations the action of the genetic algorithm successfully produced string fitness convergence.

Generation 10 showed a reduced range of fitness values, 0.01208 to 0.00356. Additionally, the parameters of each string started to show convergence. Generation 30 onwards then began to represent dominance by highly fit strings.

The fittest string from the entire process arose in generation 38, however, significant variation in the population still existed. Generation 50 showed almost total convergence to a very slightly less fit string with the final population's average fitness value of 0.002268. The fittest strings, having a penalized probability of system unavailability equal to 0.00172. This string design has the following characteristics.

M.T.I. for subsystem 1 : 54 hours.
M.T.I. for subsystem 2 : 32 hours.
and hence M.D.T. : 115.3 hours.
Cost: 923 unit.

Subsystem 1:

1 Pressure transmitter, hence 1 to trip,
1 ESD valve.

Subsystem 2:

1 Pressure transmitter, hence 1 to trip,
2 HIPS valves.

Valves of type 2
Pressure transmitter of type 1.

This design achieves an unavailability of 1.72×10^{-3}. The overall fittest string of generation 38 has a fitness value of 0.001395. This design is an exact replica of the description above except that the M.T.I. for subsystem 2 is slightly lower at 26 hours and hence the M.D.T. is 128.8 hours. This system gives an unavailability of 1.395×10^{-3} which optimizes the test intervals.

Due to the random nature of the algorithm the genetic operator convergence is not necessarily smooth. It is, however, very effective. Unfit strings 'thrown up' due to mutation or crossover are promptly eliminated. Conversely highly fit strings lost through the action of the genetic algorithm are subsequently reintroduced into later generation.

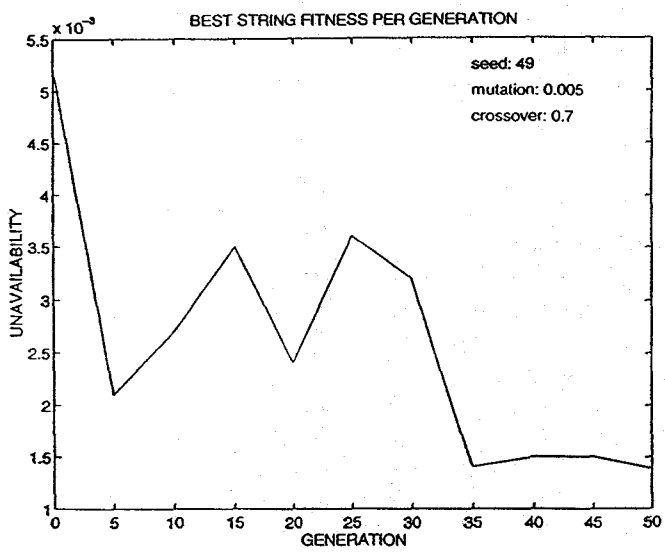

Figure 6.

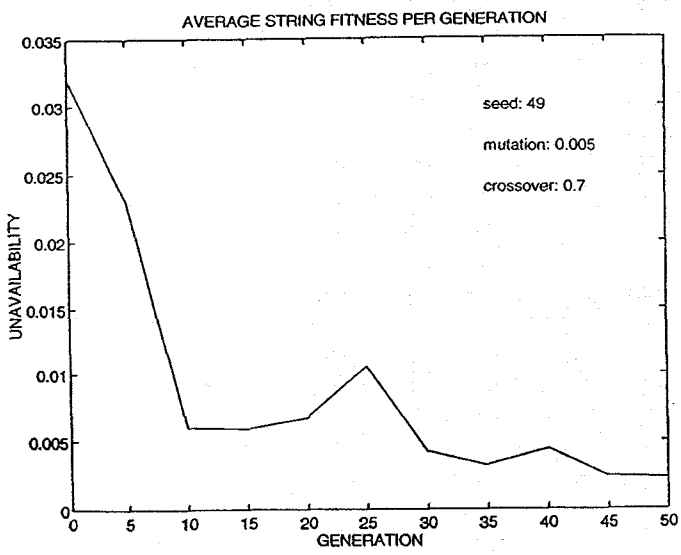

Figure 7.

CONCLUSIONS

1. The use of House events in a fault tree structure can be used to represent the failure causes of all potential designs.
2. Converting the fault tree structure to a Binary Decision Diagram results in all types of design parameters (component selection, redundancy levels and test intervals) being represented by 'probability' values in the analysis. The BDD also provides an efficient analysis method for all design options.
3. The use of a Simple Genetic Algorithm provides a means of optimization which is capable of coping with all the requirements of the design problem.
4. The approach was successfully tested on a High Pressure Protection System.

REFERENCES

1. J.D. Andrews, T.R. Moss, "Reliability and Risk Assessment", 1993; Longmans.

2. R. Billinton, R. Allan, "Reliability Evaluation of Engineering Systems, 2nd edition, 1993; Pitman.

3. A. Villmeur, "Reliability, Maintainability and Safety Assessment", Volume 2, 1992; Wiley.

4. E.J. Henley, H. Kumamota, "Reliability Engineering and Risk Assessment", 1981; Prentice-Hall.

5. J.D. Andrews, "Optimal Safety System Design Using Fault Tree Analysis", *Proc. IMechE*, Vol. 208, 1994, pp 123-131.

6. A. Rauzy, "New Algorithms for Fault Tree Analysis", *Reliability Engineering and System Safety*, Vol 40, 1993, pp 203-211.

7. 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.

8. R.M. Sinnamon, J.D. Andrews, "New Approaches to Evaluating Fault Trees", *Proceedings of ESREL 95 conference*, June 1995, pp 241-254.

9. R.M. Sinnamon, J.D. Andrews, "Improved Efficiency in Qualitative Fault Tree Analysis", *Proceedings of 12th ARTS*, Manchester, April 1996.

10. R.M. Sinnamon, J.D. Andrews, "Improved Accuracy in Quantitative Fault Tree Analysis", *Proceedings of 12th ARTS*, Manchester, April 1996.

11. D.F. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", 1989; Addison-Wesley.

12. L. Painton, J. Campbell, "Genetic Algorithms in Optimization of System Reliability", *IEEE Trans. Rel*, Vol. 44, No2, June 1995, pp 172-178.

13. F.A. Tillman, C. Hwang, W. Kuo, "Optimization of Systems Reliability", 1980; Marcel Decker.

BIOGRAPHIES

John D. Andrews, PhD, BSc.
Department of Mathematical Sciences
Loughborough University
Loughborough, Leics. LE11 3TU, U.K.
E-mail:J.D.Andrews@lboro.ac.uk

Dr John Andrews currently lectures in Risk and Safety Assessment Techniques in the Department of Mathematical Sciences at Loughborough University. Prior to this appointment he was a Senior Lecturer in the Department of Mechanical and Production Engineering at Birmingham Polytechnic and has also had two periods of employment as a Senior Scientist Engineer in the Research and Development Division at British Gas

His industrial work has involved research into methods of assessing the safety and risk of potentially hazardous industrial activities. This research is now continuing at Loughborough University. Dr Andrews is currently a member of committees of the Institution of Mechanical Engineers and the Safety and Reliability Society which focus on Risk, Safety and Reliability issues.

Rachel L. Pattison, BSc.
Department of Mathematical Sciences
Loughborough University
Loughborough, Leics. LE11 3TU, U.K.
E-mail:R.L.Pattison@lboro.ac.uk

Rachel Pattison is a second year PhD student working with Dr John Andrews on Reliability theory, mainly optimization of safety systems. She also tutors in Reliability and Mathematics of engineering at Loughborough University. Her BSc Joint Honours is in Mathematics, Physical Education and Sports Science and was gained at Loughborough.