Y. Vidya KIRAN
Biswajit MAHANTY

# RELIABILITY DESIGN OF EMBEDDED SYSTEMS

*In the current era, the role of smart devices is expanding every day. These devices depend on both software and hardware functions to produce the desired results. The success of such devices depends on a new design paradigm that considers reliability in virtually every aspect of the devices' software and hardware content. Design of a hardware system involves selection from numerous discrete choices among available component types based on cost, reliability, performance, weight, etc. Design of software systems involves the selection of the best choice from a stack of available choices with variable reliabilities and costs. We try to design an embedded system which optimizes the reliability in the perspective of cost or vice versa. An Integer Programming approach for simplified assumptions and an Evolutionary approach for the non-simplified case is proposed.*

***Keywords***: *reliability design, embedded systems, integer programming approach, evolutionary approach*

## 1. Introduction

An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular kind of application device. In this paper, we try to discuss how reliability can be designed efficiently in to embedded systems given a constraint on cost. These concepts can be extended to encompass other constraints as well. A hardware device may experience failure due to temperature, vibration etc. On the other hand, the operational profile provides the foundation of software reliability assessment. It is the operational profile that determines unit utilization and how often one or more units will cause a failure. We start with the design problem and discuss the two approaches to tackle the problem.

## 2. The Reliability Design Problem

We try to design a combined software-hardware system which satisfies the design objectives.

For a problem where cost is the design objective, the problem is formulated as:

$$Min \quad \sum_{i=1}^{s} C_i(\mathbf{x}_i) \quad Subject \cdot to \quad R(s) \geq R$$

where $C_i = cost$ of $i^{th}$ subsystem, $\mathbf{x}_i$ = solution vector, $R(s)$ = reliability of the system, $R$ = reliability constraint.

## 3. System Reliability Calculation

Reliability of a functionally similar (not identical) k-out of-n G system was calculated by Barlow and Heidtmann method. Now, we define $p_{ij}$ as the probability that control transfers from one element $i$ to another element $j$. It is independent of how element $i$ was entered. Each element is characterized by reliability $r_i$. The probability $p_{ij}$ that the control transfers from one hardware subsystem to another are 1. The system successfully completes the operation when it reaches the terminal element S. At any element $i$ the following equation holds $p_{is} + \sum_{j=1}^{n} p_{ij} = 1$

The Markov chain thus has $n$+2 states and a transition matrix $Q$ where $q_{ij} = r_i p_{ij}$ for $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,n,S$; $q_{iF} = 1\text{-}r_i$ for $i = 1,2,\ldots,n$; and $q_{FF} = q_{SS} = 1$, with all other $q_{ij} = 0$. Now the reliability of the system is calculated by the following formula: $R(s) = [(I - T)^{-1}]_{1i} r_i p_{is}$.

Where $I$ is the identity matrix of order $n$, $T$ is the $(n \times n)$ submatrix of $Q$ obtained by dropping its last two rows and columns.

## 4. Approach

Integer Programming and Evolutionary algorithms were used to arrive at the optimal design for the system. In the former, we assume that an identical component is used to serve for redundancy for any hardware element. The EA (or GA) approach is more encompassing and needs no simplifications on the design problem.

### 4.1. Integer Programming Approach

This approach to solve the design problem of embedded systems is inspired from MIP algorithm earlier proposed by Misra and Sharma. For a typical problem where reliability should be maximized given a upper limit on system cost. *ie* $\sum C_j(x_j) \le S$ ; $x_j$ represents the amount of redundancy in case of hardware systems and it represents the choice number of sorted(in order of increasing values of reliability) software modules. $C_j(x_j)$ gives the cost of $x_j^{th}$ choice number in case of software modules and $C_j(x_j)=c_j \times x_j$ in case of hardware modules. We start the procedure by calculating the upper bounds of each element. They are calculated by assigning the whole resource of the constraint to $x_j$ and determine $x_j^{max}$ by keeping all other variables at lower bound. This is repeated for all constraints and the minimum of $x_j^{max}$ is selected as upper bound. We start our search at the point $\mathbf{x} = (x_1^u, x_2^l, x_3^l, \ldots x_n^l)$ . If any $x_k$ reaches its maximum, $x_k^u$, then we initialize all $x_j$ to $x_j^l$, for $j < k$, $j \neq 1$ and increase $x_{k+1}$ by 1. We calculate a maximum value of $x_1$ which does not violate the constraints, while we retain the previous allocation to other subsystems. This would narrow our search space to only the feasible region close to the boundary. It is possible that even after finding $x_1 = x_1^{max}$, the slacks for some constraints are large enough that we can increment some $x_k$, $2 \le k \le n$ without violating any of the constraints. To avoid this we ensure that the slack i doesn't exceed $mps_i$ during the search. Each $mps_i$ (for every constraint) can be assigned a alue less than the minimum of the incremental costs of the components. We compute the objective function for all those search points which have $x_1^{max} \neq 0$ and $slack_i \le mps_i$. The optimal result of all these search points is reported.

### 4.2. Evolutionary Algorithms Approach

Biologically inspired Genetic Algorithms open a new vista both in terms of robustness and reliability of computation, which we could successfully exploit during this study of the design of reliability of embedded systems. Each element is given *nmax* positions in the chromosome which is defined as the upper bound on the number of components each element of the system can have. For software elements *nmax* = 1. The following is the representation of the chromosome for the test case:

$$C = \underbrace{1\,2\,4}_{1(\mathrm{HW})}\ \underbrace{2\,0\,4\,2\,3}_{2(\mathrm{HW})}\ \underbrace{1}_{3(\mathrm{SW})}\ \underbrace{3}_{4(SW)}\ \underbrace{2}_{5(\mathrm{SW})}\ \underbrace{4}_{6(\mathrm{SW})}\ \underbrace{4\,6\,0\,1\,3}_{7(\mathrm{HW})}\ \underbrace{7\,6\,3\,4\,1\,2\,0}_{8(\mathrm{HW})}$$

Note that the zero means that no component has been selected from the choices available.Tournament selection was used and the following is the fitness assignment procedure:

$$if\,(R(s) < \mathbf{Re}quired\ \mathbf{Re}liability)$$

$$Then\quad Fitness = \sum_i C_i(\mathbf{x}_i) + \mathbf{max}\,cost * * (\mathbf{1} + genrno * \alpha * (R(s) - \mathbf{Re}quired\ \mathbf{Re}liability)^2$$

$$else\quad Fitness = \sum_i C_i(\mathbf{x}_i)$$

*maxcost* is calculated by substituting the costliest components in to all the elements of the system. *genrno* is the current value of the generation running. $\alpha$ is a conversion parameter. Uniform crossover was used. In this crossover each gene of the offspring is selected randomly from the corresponding genes of the parents. Mutation is carried out by randomly selecting a chromosome position and substituting it with a choice randomly from the available list. The following example gives a glimpse of mutation. The component with a "-" over it is randomly replaced with another possible choice at that position.

$$P1 = 1\,3\,2\,4\,4\,\bar{6}\,0\,1\,3\,7\,6\,3\,4\,1\,2\,0 \rightarrow P1' = 1\,3\,2\,4\,4\,\bar{2}\,0\,1\,3\,7\,6\,3\,4\,1\,2\,0$$

## 5. Results

Both the cases have the following transfer probabilities. The rectangles represent software modules and the rhombuses represent the hardware elements.

### 5.1. Test Case 1

This case has the assumptions which we used for integer programming approach. Following are the choices for each element given in the form (Reliability, Cost). **HW** represents Hardware Element and **SW** means Software module.

**1(HW)**: (0.8,2); **2(HW)**: (0.9,4);
**3(SW)**: (0.8,2.5), (0.85,3), (0.9, 4), (0.95,5);
**4(SW)**: (0.9,3), (0.95,4.5), (0.98,6);
**5(SW)**: (0.95,3.5), (0.97,5);
**6(SW)**: (0.92,2.5), (0.96,4.0), (0.98,5.5);
**7(HW)**: (0.85,5); **8(HW)**: (0.95,7)

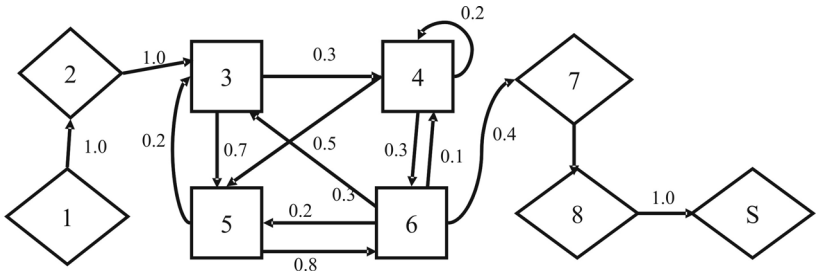


*Fig. 1. Profile of the embedded system used in the test cases*

The constraint in the above problem was taken to be the cost which was not supposed to exceed 55 units. Integer Programming and Genetic Algorithms (with parameters p_mut = 0.05; p_crossover = 0.65, Generations = 400, population = 15) returned the same answer for this deterministically solvable problem. Reliability = 0.757683 Cost = 54.5

### 5.2. Test Case 2

Following are the choices for each element given in the form (Reliability,Cost)

**1(HW)**-(0.8,2),(0.9,3),(0.95,3.5),(0.97,5);
**2(HW)**- (0.9,4);(0.92,4.5);(0.97,6)
**3(SW)**- (0.8,2.5), (0.85,3), (0.9, 4), (0.95,5);
**4(SW)**- (0.9,3); (0.95,4.5);(0.98,6)
**5(SW)**-(0.95,3.5), (0.97,5);
**6(SW)**-(0.92,2.5), (0.96,4.0), (0.98,5.5)
**7(HW)**-(0.85,5), (0.88,6), (0.92,7.5), (0.95,8.5), (0.99,10);
**8(HW)**-(0.95,7),(0.98,9),(0.99,10.5)

Constraint is that the Reliability of the system should be greater than 0.8

Genetic Algorithms produced the following result: Reliability = 0.800576 Cost = 57.5; p_mut = 0.05; p_crossover = 0.65; Generations = 400, population = 15

From the plots of the variation of cost and reliability with respect to generations the following interesting feature of the GA can be observed. The GA searched for lower and lower system costs till generation 600 and thereafter the algorithm was able to find higher reliabilities for the same cost.

*Table 1. Results for Test Case 1*

| Component | Choice | Allocation | Component | Choice | Allocation |
|---|---|---|---|---|---|
| 1(HW) | | 4 | 5(SW) | 2 | |
| 2(HW) | | 2 | 6(SW) | 3 | |
| 3(SW) | 4 | | 7(HW) | | 2 |
| 4(SW) | 3 | | 8(HW) | | 1 |

*Table 2. Results for Test Case 2*

| Component | Choice | Allocation | Component | Choice | Allocation |
|---|---|---|---|---|---|
| 1(HW) | 3 | 2 | 5(SW) | 2 | |
| 2(HW) | (1,2) | (1,1) | 6(SW) | 3 | |
| 3(SW) | 4 | | 7(HW) | 5 | 1 |
| 4(SW) | 3 | | 8(HW) | 3 | 1 |

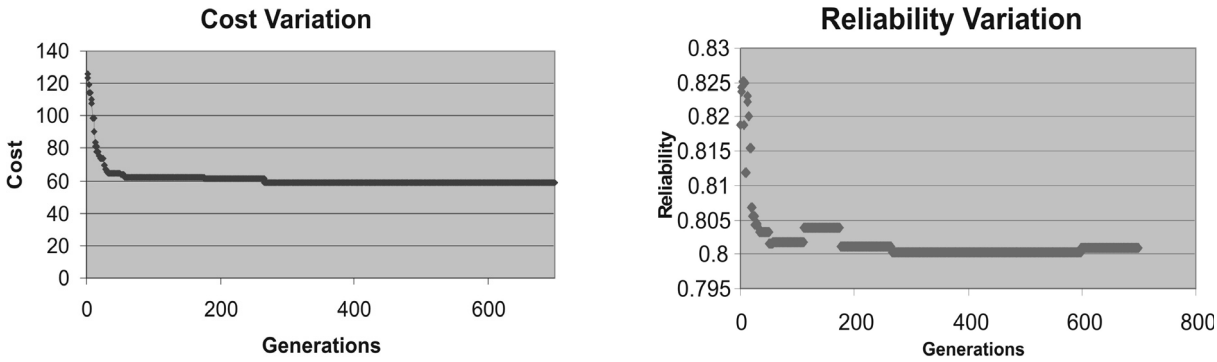


*Fig. 2. Cost and Reliability Variation with Generations*

## 6. Conclusions

The design problem was successfully tackled with the two approaches discussed. The results are very promising with proven optimal convergence on a simplified problem wherein both the approaches give the same optimal results. The GA approach has a probable near-optimal convergence on the complex problem too. Future work may be directed at solving the design problem with more realistic assumptions.

## 7. References

[1] Barlow, R. E. and K. D. Heidtmann (1984). *Computing k-out-of-n system reliability*. IEEE Trans. Reliability. R-33, 322–323.

[2] Siegrist, K. (1988). *Reliability of Systems with Markov Transfer of Control*. IEEE Transactions on Software Engineering. 14, 1049-1053.

[3] Misra, K. B. and U. Sharma (1991). *An efficient algorithm to solve integer programming problems arising in system reliability design*, IEEE Transactions on Reliability, 40, 81-91.

[4] Smith, A. E. and D. M. Tate (1993). *Genetic optimization using a penalty function*. Proceedings of the 5th International Conference on Genetic Algorithms, 499-505.

**Ing. Y. Vidya KIRAN**
**Dr. Prof. Biswajit MAHANTY**
Department of Industrial Engineering and Management
Indian Institute of Technology
Kharagpur (W.B) 721 302, India
e-mail: kiran@iem.iitkgp.ernet.in
e-mail: bm@hijli.iitkgp.ernet.in