# ภาคผนวก

International Journal of Reliability, Quality and Safety Engineering


# OPTIMAL FAULT-TOLERANT EMBEDDED SYSTEM DESIGNS CONSIDERING RELIABILITY ESTIMATION UNCERTAINTY

N. WATTANAPONGSAKORN
*Department of Computer Engineering, King Mongkut's University of Technology Thonburi*
*91 Suksawad 48, Ratburana, Tung-Kru, Bangkok 10150, Thailand*

D. W. COIT
*Department of Industrial Engineering, Rutgers University*
*96 Frelinghuysen Rd., Piscataway, NJ 08854, USA*

In this paper, we consider embedded system design optimization, considering uncertainty in the component reliability estimate, by maximizing an estimate of system reliability and also minimizing the variance of the reliability estimate. The two most common fault-tolerant embedded system architectures: N-Version Programming (NVP) and Recovery Block (RB) can be applied to provide system redundancy. We present four distinct models to demonstrate our optimization techniques with or without redundancy for NVP/0/1, NVP/1/1 or RB/1/1. All models are designed under system cost constraints. Embedded systems consist of both software and hardware components. Failures of software components/versions are a major cause of system failures. For many experimental studies, multiple software versions, which are functionality equivalent, do have failure correlation even if they have been independently developed. The failure correlation may come from faults in the software specification, faults from the voting algorithm, and/or related faults from any two software versions. Our approach considers this correlation of failures in formulating practical optimization models. Unlike most research papers, we provide a technique to optimize system reliability considering software versions with different reliabilities and correlated failures. Genetic algorithms with a dynamic penalty function is applied in solving this optimization problem, and reasonable and interesting results are captured and discussed.

*Keywords*: Estimation Uncertainty, Reliability Analysis, Fault Tolerance, Embedded System, Genetic Algorithm

## 1. Introduction

In the determination of system designs to maximize system reliability under constraints such as system cost, weight, size and others, component reliability is not known exactly but must be estimated with some uncertainty. The selected components with high reliability estimation uncertainty would result to a system design which also has very high reliability uncertainty. This is undesirable because system designers and users seek an optimal design with high predicted reliability, but also one with low estimation uncertainty.

Our goal is to develop optimization models for fault-tolerant embedded system considering reliability estimate with some uncertainty. This is an extension of work from Wattanapongsakorn & Levitan where component reliability is known.[3]

An embedded system consists of both hardware and software components. To make it fault-tolerated, some redundant techniques can be applied with extra copies of components, resulting in fault-tolerant architectures. In this paper, N-Version Programming architectures and Recovery Block architectures are considered. The detailed description of these architectures is discussed in section 2. The fault-tolerant systems are capable of tolerating software faults and/or hardware faults. For many systems, it is known that most of the system failures are related to software faults.[1, 2, 8] Therefore, optimal design software fault-tolerance is often more critical than hardware fault-tolerance in the embedded system designs. The fault-tolerated embedded system architectures result from different strategies of integrating software and hardware redundancy, together with some decision algorithms such as voting, acceptance test and comparison.[1, 2, 9]

Similar to Wattanapongsakorn et al.[3], we consider a system where each subsystem is connected in series. Each subsystem consists of both software and hardware components. The software components are application software modules, and the hardware components are processing units (with operating system, disk, etc.) or network elements.

Unlike other optimization papers which consider only software or hardware individually, or software and hardware without any dependent failures, this paper considers systems consisting of both software components, hardware components, and related faults from multiple software versions and/or hardware components. The software failure behavior, which is different from the hardware failure behavior, is considered toward the system design.

Rubinstein et al consider a redundancy allocation problem with uncertain component reliability, by maximizing the expected values of the random system reliability using genetic algorithms (GA).[4] Their approach, considering only the expected values of reliability, is not sufficient for many decision makers and design problems. In practice, system designers and users desire a designed system with a high reliability estimate, in associated with low estimated variability. Coit et al solve the problem by considering variance of system reliability estimates in addition to the expected system reliability value.[5, 6] They focus on series-parallel hardware system with arbitrarily repeated components.

Our paper combines the approach of considering both the system reliability estimate and variance with the embedded system optimization approach. This results in a very practical reliability optimization models for the design of fault-tolerant embedded systems.

The systems that we model are series-parallel fault-tolerant systems. The redundancy allocation problem for series-parallel systems is known to be difficult (NP-hard). Many researchers have proposed a variety of approaches to solve this problem

using, for example, integer programming, branch-and-bound, dynamic programming, mixed integer and nonlinear programming. Recent optimization approaches are based on heuristics such as the Genetic Algorithm (GA), and the Tabu Search (TS). All of these approaches were developed for either optimizing reliability for software or hardware systems. Here we consider systems consisting of both software and hardware components. The software failure behavior, which is different from the hardware failure behavior, is considered.

Optimization models have been developed to select both software and hardware components and the degree of redundancy to optimize the overall system reliability, with a total cost constraint. In the system, there are a specified number of subsystems in series. For each subsystem, there are several hardware and software choices to be made. The system is designed using components, each with estimated reliability, but with known cost.

Genetic Algorithms (GA) are used as the optimization approach. The term 'genetic' derives from the roughly analogous natural re-producing new-born population by crossover and mutation. There are competitions among the population; the stronger ones will survive to the next generation and the weak ones will soon die out. GA is a heuristic optimization model that has been applied effectively to solve many difficult problems in different fields such as scheduling, facility layout, and graph coloring/ graph partitioning problems. It is a stochastic algorithm with performance depending on the solution encoding, crossover breeding operator, elitist selection and mutation operator.

In section 2, the description of the fault-tolerant system architectures that we model, shown in Fig. 1, is discussed. Section 3 presents the concept of reliability estimation variability for each of the system architecture models, including the higher-order information of component reliability estimates. Section 4 presents four optimization models to maximize reliability considering uncertainty. The first model does not consider component redundancy, while the other models each does consider a distinct fault-tolerant architecture type. Section 5 explains the GA and its parameter settings. Next, in section 6, the effectiveness of our optimization models is demonstrated using a numerical example with the GA optimization approach. Lastly, in section 7, we end the paper with a summary and conclusion.

## ASSUMPTIONS

1. Each software component, hardware component or the system has 2 states: functional or fail
2. Reliability of each software or hardware component is unknown, but estimable
3. There is no failure repair for each component or system
4. Hardware redundancy is in active mode (i.e., hot spares)
5. Failure of individual hardware components are s-independent

## NOTATION

- **X/i/j** System architecture *X*, with *i* hardware faults tolerated and *j* software faults tolerated
- **n** Number of subsystems within the distributed system
- $m_i$ Number of hardware component choices available for subsystem *i*
- $p_i$ Number of software versions available for subsystem *i*
- **R** Estimated reliability of the distributed system
- $R_i$ Estimated reliability of the subsystem *i*
- $Rhw_{ij}$ Reliability of hardware component *j* at subsystem *i*
- $Rsw_{ik}$ Reliability of software component *k* at subsystem *i*
- $Chw_{ij}$ Cost of using hardware component *j* at subsystem *i*
- $Csw_{ik}$ Cost of developing software version *k* at subsystem *i*
- **Cost** maximum allowable cost (constraint)
- **Px** Probability that event *X* occurs
- **Qx** Probability that event *X* does not occur; Qx = 1 – Px
- $Pv_i$ Probability of failure of software version *i*
- $Prv_{ij}$ Probability of failure from related fault between two software versions, *i* and *j*
- $P_{all}$ Probability of failure from related fault among all software versions, due to faults in specification
- **Pd** Probability of failure of decider or voter
- $Ph_i$ Probability of failure of hardware component *i*. If only one hardware is applied, $Ph_i$ = Ph for all *i*

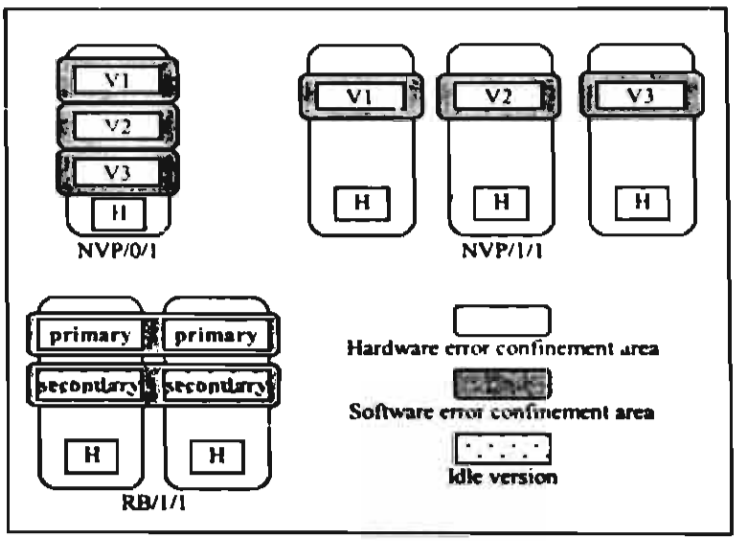


Fig. 1. Fault-tolerant architectures: NVP/0/1, NVP/1/1 and RB/1/1. [1, 2]

## 2. Fault Tolerant System Architectures to Maximize Reliability

### 2.1. *N-version Programming (NVP) architecture*

N-version Programming consists of an adjudication module called a voter, and N independently developed software versions, which are functionally equivalent.[1, 2] N is usually an odd number. This NVP model is based on the same concepts as N-Modular Redundancy (NMR), which is a hardware fault-tolerant architecture.[8] In the NVP model, all N software versions are executed for the same task at the same time (i.e., in parallel), and their outputs are collected and evaluated by the voter. The majority of the outputs determine the voter decision. Two subclasses of NVP architecture are discussed in detail next.

#### 2.1.1 *NVP/0/1 architecture*

This model has zero hardware faults tolerated and a single software fault tolerated, as shown in Fig. 1. The system architecture consists of three independent software versions (components) running in parallel on a single hardware component. This system fails if one of the following conditions are true: a single hardware fault, two out of three software version faults, a related fault between software versions, faults from software specification, and faults from the voting algorithm. The probability that an unacceptable result occurs during a single task iteration, 1-R(t), or P unreliability is given by:

The NVP/1/1 model consists of three independent software versions running on a hardware component. All conceivable system states where the system have failed have been considered and enumerated. The following equation provides the probability that an unacceptable result occurs during a single task cycle.[3] $a_i$, $k_{ij}$ and $h_{ij}$ values are presented in Table 1.

$$P = \sum_{i=1}^{s} \left( a_i \prod_{j \in C_i} p_j^{k_{ij}} q_j^{h_{ij}} \right) \qquad (1)$$

where

$s$ = number of additive terms when all failure probabilities have been enumerated; $s$ = 20 for NVP/1/1 architecture

$a_i$ = integer coefficient

$C_i$ = component type set for $i^{th}$ additive term

$k_{ij}$= power coefficient for $j^{th}$ component reliability in set $C_i$

$h_{ij}$= power coefficient for $j^{th}$ component unreliability in set $C_i$

$p_j$ = unreliability of $j^{th}$ type of component

$q_j$ = reliability of $j^{th}$ type of component, $p_j + q_j = 1$ for all $j$

$p_j$ and $q_j$ definitions and comparisons with notation from Wattanaponsakorn et al are as follows,

| | |
|---|---|
| $p_1$=Prv | $q_1$=Qrv |
| $p_2$=Pd | $q_2$=Qd |

| | |
|---|---|
| $p_3$=$P_{all}$ | $q_3$=$Q_{all}$ |
| $p_4$=$Pv_1$ | $q_4$=$Qv_1$ |
| $p_5$=$Pv_2$ | $q_5$=$Qv_2$ |
| $p_6$=$Pv_3$ | $q_6$=$Qv_3$ |
| $p_7$=Ph | $q_7$=Qh |

Table 1. $a_i$, $k_{ij}$ and $h_{ij}$ expressed in a matrix form for NVP/0/1 architecture.

| $k_{ij}$ | | | | | | | | $h_{ij}$ | | | | | | | | $a_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $j$=1 | 2 | 3 | 4 | 5 | 6 | 7 | | $j$=1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| $i$=1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $i$=1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 5 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 6 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 7 | 3 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 8 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 9 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 9 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

Eq. (1) is a general expression that can be adapted to other fault tolerant architectures by using the appropriate $a_i$, $k_{ij}$ and $h_{ij}$ values determined from enumerated the failure probabilities.

### 2.1.2. *NVP/1/1 architecture*

This model consists of three independent software versions, each running on a separate hardware component. Any hardware failure can cause the software running on it to produce unacceptable results. The system is functional if 2 out of 3 software versions (on working hardware) are functioning. Failures of a software version and an unrelated hardware component lead to system failures.

The probability that an unacceptable result occurs during a single task iteration, 1-R(t), or P unreliability is presented by Eq. (1) where $a_i$, $k_{ij}$ and $h_{ij}$ for NVP/1/1 architecture are listed in Table 2.

Table 2. $a_i$, $k_{ij}$ and $h_{ij}$ expressed in a matrix form for NVP/1/1 architecture.

| | $k_{ij}$ | | | | | | | $h_{ij}$ | | | | | | | $a_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $j=1$ | 2 | 3 | 4 | 5 | 6 | 7 | | $j=1$ | 2 | 3 | 4 | 5 | 6 | 7 | |
| $i=1$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $i=1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 5 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 6 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 7 | 3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 8 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 8 | 3 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 9 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 10 | 3 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 11 | 0 | 0 | 0 | 1 | 1 | 0 | 2 | 11 | 3 | 1 | 1 | 0 | 0 | 1 | 1 | -1 |
| 12 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 12 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 13 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 14 | 0 | 0 | 0 | 1 | 0 | 1 | 2 | 14 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | -1 |
| 15 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 15 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 16 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 17 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 17 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | -1 |
| 18 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 18 | 3 | 1 | 1 | 0 | 1 | 1 | 2 | 2 |
| 19 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 19 | 3 | 1 | 1 | 1 | 0 | 1 | 2 | 2 |
| 20 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 20 | 3 | 1 | 1 | 1 | 1 | 0 | 2 | 2 |

## 2.2. *Recovery Block (RB): RB/1/1 Architecture*

The RB model consists of an adjudication module called an *acceptance test*, and at least two software components, called alternates.[1, 2] At the beginning, the output of the first or primary alternate is tested for acceptance. If it fails, the process will *roll back* to the beginning of the process, and then let the second alternate execute and test its output for acceptance again. This process continues until the output from an alternate is accepted or all outputs of the alternates have been tested and fail.

The system consists of two hardware components, each running two independent software versions; primary and secondary. The primary version is active until it fails, and the secondary version is the backup spare. The system failures occur when both versions fail, or both hardware components fail. The probability that an unacceptable result occurs during a single task iteration, P is presented by Eq. (1) where $k_{ij}$ and $h_{ij}$ for RB/1/1 architecture are listed in Table 3.

Table 3. $a_i$, $k_{ij}$ and $h_{ij}$ expressed in a matrix form for RB/1/1 architecture.

| | $k_{ij}$ | | | | | | | $h_{ij}$ | | | | | | | $a_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | $j=1$ | 2 | 3 | 4 | 5 | 6 | 7 | $i$ | $j=1$ | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 1 | 1 | 0 | 2 | 6 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | -1 |

With these fault-tolerant architectures, we develop four optimization models for fault-tolerant embedded system considering reliability estimate with some uncertainty. The components, which are available for the system design, each has reliability uncertainty with reliability estimate and its variance.

In the next section, we formulate equations for system reliability estimate and variance for reliability estimate for the three system fault-tolerant architectures. These equations will be used in our four optimization models, discussed later in this paper in section 4.

### 3. Reliability Estimation Variability

Usually the exact component unreliability, $p_j$, or reliability, $q_j$, are not known. They are estimated from life test data or field failure records. The estimated $\hat{p}_j$ or $\hat{q}_j$ are used to replace the true but unknown in Eq. (1),

$$\hat{P} = \sum_{i=1}^{s} \left( a_i \prod_{j \in C_i} \hat{p}_j^{k_{ij}} \hat{q}_j^{k_{ij}} \right) \tag{2}$$

Direct calculation of $E[\hat{P}]$ and $Var(\hat{P})$ are difficult due to the coupling of $\hat{p}_j$ and $\hat{q}_j$. Therefore, Eq. (2) has been rearranged, as follows

$$\hat{P} = \sum_{i=1}^{s} \left( a_i \prod_{j \in C_i} (1-\hat{q}_j)^{k_{ij}} \hat{q}_j^{k_{ij}} \right) \tag{3}$$

Eq. (3) can be rearranged by expanding $(1-\hat{q}_j)^{k_{ij}}$ terms, resulting in equation Eq. (4)

$$\hat{P} = \sum_{i=1}^{t} \left( b_i \prod_{j \in C_i} \hat{q}_j^{n_{ij}} \right) \tag{4}$$

where $t$ = number of subsystems after expansion, $t > s$
$b_i$ = integer coefficient

$t$ is the number of terms after the expansion. $b_i$ and $n_{ij}$ are determined by grouping similar terms. This expansion procedure is conducted automatically using Matlab code based on the parameters in Tables 1, 2 or 3. Due to the length of the expansion, detailed computational procedure is omitted. Tables 4, 5 and 6 list all the expansion results.

Table 4 $n_{ij}$ and $b_i$ expressed in a matrix form for NVP 0/1 architecture.

| | $n_{ij}$ | | | | | | | $b_i$ |
|---|---|---|---|---|---|---|---|---|
| i | j =1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 7 | 3 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 8 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 9 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 10 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 11 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 12 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 13 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
| 14 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | -1 |
| 15 | 3 | 1 | 1 | 0 | 0 | 0 | 1 | -1 |
| 16 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | -1 |
| 17 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | -1 |
| 18 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | -1 |
| 19 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | -1 |
| 20 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | -1 |
| 21 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | -1 |
| 22 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 23 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 24 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 5. $n_{ij}$ and $b_i$ expressed in a matrix form for NVP/1/1 Architecture.

| | $n_{ij}$ | | | | | | | $b_i$ |
|---|---|---|---|---|---|---|---|---|
| i | j=1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 5 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 7 | 3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 8 | 3 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 9 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 10 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 11 | 3 | 1 | 1 | 0 | 0 | 1 | 1 | -1 |
| 12 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 13 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 14 | 3 | 1 | 1 | 0 | 1 | 0 | 1 | -1 |
| 15 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 16 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 17 | 3 | 1 | 1 | 1 | 0 | 0 | 1 | -1 |
| 18 | 3 | 1 | 1 | 0 | 1 | 1 | 2 | 2 |
| 19 | 3 | 1 | 1 | 1 | 0 | 1 | 2 | 2 |
| 20 | 3 | 1 | 1 | 1 | 1 | 0 | 2 | 2 |
| 21 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 22 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 23 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 24 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
| 25 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | -1 |
| 26 | 3 | 1 | 1 | 1 | 0 | 0 | 0 | -1 |
| 27 | 3 | 1 | 1 | 0 | 1 | 0 | 0 | -1 |
| 28 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | -1 |
| 29 | 3 | 1 | 1 | 0 | 1 | 1 | 0 | -1 |
| 30 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | -1 |
| 31 | 3 | 1 | 1 | 1 | 0 | 1 | 0 | -1 |
| 32 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 33 | 3 | 1 | 1 | 0 | 1 | 1 | 2 | -2 |
| 34 | 3 | 1 | 1 | 0 | 1 | 1 | 3 | 1 |
| 35 | 3 | 1 | 1 | 0 | 1 | 1 | 2 | -2 |
| 36 | 3 | 1 | 1 | 0 | 1 | 1 | 3 | 1 |
| 37 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 38 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 39 | 3 | 1 | 1 | 0 | 0 | 1 | 2 | 2 |
| 40 | 3 | 1 | 1 | 0 | 0 | 1 | 3 | -1 |
| 41 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 42 | 3 | 1 | 1 | 1 | 1 | 0 | 2 | -2 |
| 43 | 3 | 1 | 1 | 1 | 1 | 0 | 3 | 1 |
| 44 | 3 | 1 | 1 | 0 | 1 | 0 | 2 | -2 |
| 45 | 3 | 1 | 1 | 0 | 1 | 0 | 3 | 1 |
| 46 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 47 | 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 48 | 3 | 1 | 1 | 0 | 1 | 0 | 2 | 2 |
| 49 | 3 | 1 | 1 | 0 | 1 | 0 | 3 | -1 |
| 50 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 51 | 3 | 1 | 1 | 1 | 0 | 1 | 2 | -2 |
| 52 | 3 | 1 | 1 | 1 | 0 | 1 | 3 | 1 |
| 53 | 3 | 1 | 1 | 1 | 0 | 0 | 2 | -2 |
| 54 | 3 | 1 | 1 | 1 | 0 | 0 | 3 | 1 |
| 55 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 56 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 57 | 3 | 1 | 1 | 1 | 0 | 0 | 2 | 2 |
| 58 | 3 | 1 | 1 | 1 | 0 | 0 | 3 | -1 |
| 59 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | -2 |
| 60 | 3 | 1 | 1 | 0 | 1 | 1 | 3 | -2 |
| 61 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | -2 |
| 62 | 3 | 1 | 1 | 1 | 0 | 1 | 3 | -2 |
| 63 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | -2 |
| 64 | 3 | 1 | 1 | 1 | 1 | 0 | 3 | -2 |
| 65 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 66 | 3 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 67 | 3 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 68 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 69 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | -1 |
| 70 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 71 | 3 | 1 | 1 | 1 | 0 | 1 | 2 | -2 |
| 72 | 3 | 1 | 1 | 1 | 0 | 1 | 3 | 1 |
| 73 | 3 | 1 | 1 | 0 | 1 | 1 | 2 | -2 |
| 74 | 3 | 1 | 1 | 0 | 1 | 1 | 3 | 1 |
| 75 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 76 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | -1 |
| 77 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 78 | 3 | 1 | 1 | 1 | 1 | 0 | 2 | -2 |
| 79 | 3 | 1 | 1 | 1 | 1 | 0 | 3 | 1 |
| 80 | 3 | 1 | 1 | 0 | 1 | 1 | 2 | -2 |
| 81 | 3 | 1 | 1 | 0 | 1 | 1 | 3 | 1 |
| 82 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 83 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | -1 |
| 84 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | -1 |
| 85 | 3 | 1 | 1 | 1 | 1 | 0 | 2 | -2 |
| 86 | 3 | 1 | 1 | 1 | 1 | 0 | 3 | 1 |
| 87 | 3 | 1 | 1 | 1 | 0 | 1 | 2 | -2 |
| 88 | 3 | 1 | 1 | 1 | 0 | 1 | 3 | 1 |
| 89 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | 2 |
| 90 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | 2 |
| 91 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | 2 |
| 92 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 93 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | -1 |
| 94 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 95 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | -1 |
| 96 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 |
| 97 | 3 | 1 | 1 | 1 | 1 | 1 | 3 | -1 |

Table 6. $n_{ij}$ and $b_i$ Expressed in a matrix form for RB/1/1 architecture

| | $n_{ij}$ | | | | | | | $b_i$ |
|---|---|---|---|---|---|---|---|---|
| i | j =1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 6 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | -1 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 8 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | -1 |
| 10 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | -2 |
| 11 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | 1 |
| 12 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | -1 |
| 13 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | -1 |
| 14 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 16 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2 |
| 17 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | -1 |
| 18 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 19 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | -1 |
| 20 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | -2 |
| 21 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 1 |
| 22 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | -2 |
| 23 | 1 | 1 | 1 | 0 | 1 | 0 | 2 | 1 |
| 24 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 2 |
| 25 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | -1 |

From the tables, NVP/0/1, NVP/1/1 and RB/1/1 have $t$ = 24, 97 and 25, respectively. Based on the coefficients $n_{ij}$, $b_i$ and component reliability information, Eq. (4) can be used to obtain the mean and the variance of unreliability $\hat{P}$. Together with the higher-order moment information of component reliability estimates, the mean and the variance of unreliability $\hat{P}$, can be obtained as follows.[5]

$$E[\hat{P}] = \sum_{i=1}^{t}\left( b_i \prod_{j \in C_i} E[\hat{q}_j^{n_{ij}}] \right) \tag{5}$$

$$Var(\hat{P}) = \sum_{i=1}^{t}\left\{ b_i^2 \left( \prod_{j=C_i} E[\hat{q}_j^{2n_{ij}}] - \prod_{j=C_i} \left( E[\hat{q}_j^{n_{ij}}] \right)^2 \right) \right\} + 2\sum_{i<m}^{t}\left\{ b_i b_m \left( \prod_{j=C_{im}} E[\hat{q}_j^{n_{ij}+n_{mj}}] - \prod_{j=C_{im}} E[\hat{q}_j^{n_{ij}}] E[\hat{q}_j^{n_{mj}}] \right) \right\}$$

$$\text{where} \quad C_{im} = C_i \cup C_j \tag{6}$$

### 3.1 *Numerical example*

To show the relationship of reliability estimate and variance of reliability estimate for each component, we provide several numerical examples. Table 7 lists component reliability estimates or unreliability estimate values. These data are selected directly

from Wattanapongsakorn et al.[3] Eq. (5) and Eq. (6) are also valid, as long as high moments of component reliability estimates are known. Without loss of generality, it is assumed Bernoulli test and applied binomial distribution theory was used to estimate high moments.[5]

Table 7. Parameters and definition of component's variance of reliability estimate.

| Unreliability Estimate | Reliability Estimate | Variance of Reliability Estimate |
|---|---|---|
| $P_{rv}$=0.004 | $Q_{rv}$=0.996 | $Var(P_{rv})= Var(Q_{rv})= (P_{rv}\times Q_{rv})/\eta_1$ |
| $P_d$=0.02 | $Q_d$=0.98 | $Var(P_d)= Var(Q_d)= (P_d\times Q_d)/\eta_2$ |
| $P_{all}$=0.005 | $Q_{all}$=0.995 | $Var(P_{all})= Var(Q_{all})= (P_{all}\times Q_{all})/\eta_3$ |
| $P_{v_1}$=0.035 | $Q_{v_1}$=0.965 | $Var(P_{v_1})= Var(Q_{v_1})= (P_{v_1}\times Q_{v_1})/\eta_4$ |
| $P_{v_2}$=0.046 | $Q_{v_2}$=0.954 | $Var(P_{v_2})= Var(Q_{v_2})= (P_{v_2}\times Q_{v_2})/\eta_5$ |
| $P_{v_3}$=0.09 | $Q_{v_3}$=0.910 | $Var(P_{v_3})= Var(Q_{v_3})= (P_{v_3}\times Q_{v_3})/\eta_6$ |
| $P_h$=0.03 | $Q_h$=0.970 | $Var(P_h)= Var(Q_h)= (P_h\times Q_h)/\eta_7$ |

where $\eta = [\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6, \eta_7]$ is variance-factor vector, and $\eta_i$ = integer. For example, when $\eta = [6, 4, 2, 2, 4, 3, 6]$, the corresponding component variances are given in Table 8.

Table 8. Component reliability estimate variance.

| Unreliability Estimate | Reliability Estimate | Variance of Reliability Estimate |
|---|---|---|
| $P_{rv}$=0.004 | $Q_{rv}$=0.996 | $Var(P_{rv})= Var(Q_{rv})$=0.000664 |
| $P_d$=0.02 | $Q_d$=0.98 | $Var(P_d)= Var(Q_d)$=0.0049 |
| $P_{all}$=0.005 | $Q_{all}$=0.995 | $Var(P_{all})= Var(Q_{all})$=0.0024875 |
| $P_{v_1}$=0.035 | $Q_{v_1}$=0.965 | $Var(P_{v_1})= Var(Q_{v_1})$=0.016 |
| $P_{v_2}$=0.046 | $Q_{v_2}$=0.954 | $Var(P_{v_2})= Var(Q_{v_2})$=0.012 |
| $P_{v_3}$=0.09 | $Q_{v_3}$=0.910 | $Var(P_{v_3})= Var(Q_{v_3})$=0.03 |
| $P_h$=0.03 | $Q_h$=0.970 | $Var(P_h)= Var(Q_h)$=0.004 |

Table 9 lists six results with respect to different component variance. It is shown that system unreliability is unchanged even if component valiances vary as $\eta$ changes. It can be observed that as component variances become small, the overall variance of the system unreliability estimate $\hat{P}$ also decreases.

Table 9. Parameters of components and system unreliability $P$, system unreliability estimate $E[\hat{P}]$, and system variance of unreliability estimate $Var(\hat{P})$.

| $\eta$ | $P$ | $E[\hat{P}]$ | $Var(\hat{P})$ |
|---|---|---|---|
| [1, 1, 1, 1, 1, 1, 1] | 0.05571 | 0.03716 | 0.03578 |
| [2, 2, 2, 2, 2, 2, 2] | 0.05571 | 0.06317 | 0.02801 |
| [6, 4, 2, 2, 4, 3, 6] | 0.05571 | 0.06188 | 0.01441 |
| [12, 8, 4, 4, 8, 6, 12] | 0.05571 | 0.05925 | 0.00714 |
| [8, 8, 8, 8, 8, 8, 8] | 0.05571 | 0.06068 | 0.00777 |
| [12, 12, 12, 12, 12, 12, 12] | 0.05571 | 0.05925 | 0.00521 |

## 4.0 Optimization Models to Maximize Reliability Considering Uncertainty

In this section, we present our four-optimization models for reliability of distributed systems. A distributed system, shown on Fig. 2, consists of subsystems where software components/versions are mapped (or distributed) on to various hardware components. We consider the system having all subsystems connected in series. Each subsystem has both software and hardware components. Each of our optimization models allows different fault-tolerant architectures or component redundancy choices suitable for different situations. The models are formulated as follows.

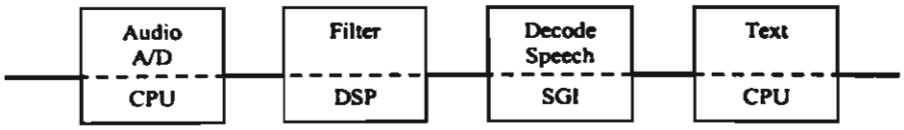


Fig. 2. A distributed system.

**Model 1**: Find the optimal set of software and hardware allocations for all subsystems (without redundancy). The problem formulation is to maximize the system reliability estimate, subjected to a specified cost constraint, *Cost*. The system has all subsystems connected in series. Each subsystem reliability estimate is the product of a chosen software version and a hardware component available for selection.

**Formulation:** Maximize system reliability estimate with its variance of reliability estimate by choosing the optimal set of hardware and software components for each subsystem by:

$$\max \quad \hat{R}(x) - penalty \times Var\left(\hat{R}(x)\right)$$

Subject to

$$\sum_{j=1}^{m} X_{ij} = 1, \quad \sum_{k=1}^{p} Y_{ik} = 1,$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}C_{ij} + \sum_{i=1}^{n}\sum_{k=1}^{p} Y_{ik}C_{ik} \le \text{Cost}$$

$$X_{ij} = 0,1 \quad Y_{ij} = 0,1 \quad i=1,2,\ldots,n \quad j=1,2,\ldots,m \quad k=1,2,\ldots,p$$

where $x$ = solution vector ($X_{ij}$, $Y_{ij}$)

$$\hat{R}(x) = \prod_{i=1}^{n} \left(\hat{R}_i(x)\right) = \prod_{i=1}^{n} \left( \sum_{j=1}^{m_i} \sum_{k=1}^{p_i} X_{ij} Y_{ik} \hat{R}hw_{ij} \hat{R}sw_{ik} \right) \tag{7}$$

$$Var\left(\hat{R}(x)\right) = \prod_{i=1}^{n} \left(Var\left(\hat{R}_i(x)\right) + \hat{R}_i(x)^2\right) - \prod_{i=1}^{n} \left(\hat{R}_i(x)\right)^2 \tag{8}$$

$$Var\left(\hat{R}_i(x)\right) = \left(Var\left(\hat{R}hw_i\right) + \hat{R}hw_i^2\right)\left(Var\left(\hat{R}sw_i\right) + \hat{R}sw_i^2\right) - \left(\hat{R}hw_i \hat{R}sw_i\right)^2 \tag{9}$$

$$\hat{R}hw_i = \sum_{j=1}^{m_i} X_{ij} \hat{R}hw_{ij}, \quad \hat{R}sw_i = \sum_{k=1}^{p_i} Y_{ik} \hat{R}sw_{ik}$$

$$Var\left(\hat{R}hw_i\right) = \sum_{j=1}^{m_i} X_{ij} Var\left(\hat{R}hw_{ij}\right), \quad Var\left(\hat{R}sw_i\right) = \sum_{k=1}^{p_i} Y_{ik} Var\left(\hat{R}sw_{ik}\right)$$

The design objective is to identify solutions with very high reliability, but also with a low reliability estimate variance. This is accomplished by penalizing the objective function, system reliability estimate. The 'penalty' is a tunable parameter based on a system designers tolerance for risk, i.e., actual reliability deviation from the estimate. By penalizing the variance, the final solution represents a compromise between high reliability and low variance.

**Model 2:** Find the optimal set of software and hardware allocations for all subsystems (with or without NVP/0/1 redundancy). This and the following models are suited for systems that handle more critical tasks. The problem formulation for this model is the same as in the previous model, except that each subsystem uses NVP/0/1 redundancy allocation as its reliability estimate and variance of reliability estimate, calculated according to the NVP/0/1 redundancy configuration. The parameters considered for the reliability of the NVP architecture are available as component reliability estimate and variance of reliability estimate. Each allocated software version is allowed to have a different reliability estimated value, unlike several proposed models where all of the software versions have the same reliability value.[2]

**Formulation**: Maximize system reliability estimate with its variance of reliability estimate by choosing an optimal set of hardware and software components for each subsystem by:

$$\max \quad \hat{R}(x) - penalty \times Var\left(\hat{R}(x)\right)$$

Subject to

$$\sum_{j=1}^{m_i} X_{ij} = 1, \quad \sum_{k=1}^{p_i} Y_{ik} = 1 \text{ or } 3$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} X_{ij} C_{ij} + \sum_{i=1}^{n} \sum_{k=1}^{p_i} Y_{ik} C_{ik} \le Cost$$

where $X_{ij} = 0,1 \quad Y_{ij} = 0,1 \quad i = 1,2,\ldots,n \quad j = 1,2,\ldots,m_i \quad k = 1,2,\ldots,p_i$

$$\hat{R}(x) = \prod_{i=1}^{n} \left(\hat{R}_i(x)\right)$$

$$Var\left(\hat{R}(x)\right) = Eq.(8)$$

$$\left.\begin{aligned} \hat{R}_i(x) &= \sum_{j=1}^{m_i} \sum_{k=1}^{p_i} X_{ij} Y_{ik} \hat{R}hw_{ij} \hat{R}sw_{ik} \\ Var\left(\hat{R}_i(x)\right) &= Eq.(9) \end{aligned}\right\} \quad if \quad \sum_{j=1}^{m_i} X_{ij} = 1, \quad \sum_{k=1}^{p_i} Y_{ik} = 1$$

$$\left.\begin{aligned} \hat{R}_i(x) &= 1 - (Eq.(5)) \\ Var\left(\hat{R}_i(x)\right) &= Eq.(6) \end{aligned}\right\} \quad if \quad \sum_{j=1}^{m_i} X_{ij} = 1 \quad \sum_{k=1}^{p_i} Y_{ik} = 3$$

**Model 3:** Find the optimal set of software and hardware allocations for all subsystems (with or without NVP/1/1 redundancy). This model extends model 2, but instead of zero hardware faults tolerated, it has a single hardware fault tolerated.

$$\max \quad \hat{R}(x) - penalty \times Var\left(\hat{R}(x)\right)$$

Subject to

$$\left(\sum_{j=1}^{m} X_{ij} = 1 \ and \ \sum_{k=1}^{n} Y_{ik} = 1\right) or \left(\sum_{j=1}^{m} X_{ij} = 3 \ and \ \sum_{k=1}^{n} Y_{ik} = 3\right)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} C_{ij} + \sum_{i=1}^{n} \sum_{k=1}^{n} Y_{ik} C_{ik} \le Cost$$

$$X_{ij} = 0,1,3 \quad Y_{ij} = 0,1 \quad i = 1,2,\ldots,n \quad j = 1,2,\ldots,m_i \quad k = 1,2,\ldots,p_i$$

where

$$\hat{R}(x) = \prod_{i=1}^{n} \left(\hat{R}_i(x)\right)$$

$$Var\left(\hat{R}(x)\right) = Eq.(8)$$

$$\left.\begin{aligned} \hat{R}_i(x) &= \sum_{j=1}^{m_i} \sum_{k=1}^{p_i} X_{ij} Y_{ik} \hat{R}hw_{ij} \hat{R}sw_{ik} \\ Var\left(\hat{R}_i(x)\right) &= Eq.(9) \end{aligned}\right\} \quad if \quad \sum_{j=1}^{m_i} X_{ij} = 1, \quad \sum_{k=1}^{p_i} Y_{ik} = 1$$

$$\left.\begin{aligned} \hat{R}_i(x) &= 1 - (Eq.(5)) \\ Var\left(\hat{R}_i(x)\right) &= Eq.(6) \end{aligned}\right\} \quad if \quad 3\sum_{j=1}^{m_i} X_{ij} = 3 \quad \sum_{k=1}^{p_i} Y_{ik} = 3$$

**Model 4:** Find the optimal set of software and hardware allocations for all subsystems (with or without RB/1/1 redundancy). This model is also based on model 2, but captures optimization analysis for the Recovery Block architecture.

$$Max\left\{\hat{R}(x) - penalty\left(Var\left(\hat{R}(x)\right)\right)\right\}$$

Subject to

$$\left(\sum_{j=1}^{m} X_{ij} = 1 \ and \ \sum_{k=1}^{n} Y_{ik} = 1\right) or \left(\sum_{j=1}^{m} X_{ij} = 2 \ and \ \sum_{k=1}^{n} Y_{ik} = 2\right)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} C_{ij} + \sum_{i=1}^{n} \sum_{k=1}^{n} Y_{ik} C_{ik} \le Cost$$

$$X_{ij} = 0,1,2 \quad Y_{ij} = 0,1 \quad i = 1,2,\ldots,n \quad j = 1,2,\ldots,m_i \quad k = 1,2,\ldots,p_i$$

where

$$\hat{R}(x) = \prod_{i=1}^{n} \left(\hat{R}_i(x)\right)$$

$$Var\left(\hat{R}(x)\right) = Eq.\ (8)$$

$$\left.\begin{aligned} \hat{R}_i(x) &= \sum_{j=1}^{m_i}\sum_{k=1}^{p_i} X_{ij} Y_{ik} \hat{R}hw_{ij} \hat{R}sw_{ik} \\ Var\left(\hat{R}_i(x)\right) &= Eq.\ (9) \end{aligned}\right\} \quad if \quad \sum_{j=1}^{m_i} X_{ij} = 1, \quad \sum_{k=1}^{p_i} Y_{ik} = 1$$

$$\left.\begin{aligned} \hat{R}_i(x) &= 1 - Eq.\ (5) \\ Var\left(\hat{R}_i(x)\right) &= Eq.\ (6) \end{aligned}\right\} \quad if \quad 2\sum_{j=1}^{m_i} X_{ij} = 2 \quad \sum_{k=1}^{p_i} Y_{ik} = 2$$

## 5. Genetic Algorithm Implementation

GA requires that the system design (phenotype) be encoded as a solution vector (genotype). Then, genetic operators (crossover, mutation) are applied over successive generations until the GA converges to a solution or a pre-determined maximum number of generations is reached.

### 5.1. *Encoding*

For an embedded (hardware and software) system with $n$ subsystems connected in series, the string encoding can be represented as :

H1 S1 | H2 S2 | H3 S3 | ... | Hn Sn

where *Hi*, with $0 \leq i \leq n$ is the selected hardware component for subsystem *i*, and *Si* is the selected software component/version for the specified subsystem.

Suppose we have $m$ choices of hardware components and $p$ choices of software components/versions for each subsystem.

Model 1: *Hi* can be 1, ..., $m$ and *Si* can be 1, ..., $p$.

Model 2: *Hi* can be 1, ..., $m$ and *Si* can be 1, ..., $\left(p + \frac{p!}{3!(p-3)!}\right)$. If NVP/0/1 redundancy is selected for the subsystem, three different software versions and one hardware component are chosen.

Another example: Assuming we have four different software versions available for a subsystem.

Let 1/2/3/4 = choose software version 1,2,3,4 respectively
5 = choose software version 2,3,4 (software version 1 is not chosen)
6 = choose software version 1,3,4 (software version 2 is not chosen)
7 = choose software version 1,2,4 (software version 3 is not chosen)
8 = choose software version 1,2,3 (software version 4 is not chosen)

Model 3 is similar to model 2 in the sense that the number of component choices are the same. *Hi* can be 1, ..., *m* and *Si* can be 1, ..., $\left(p + \frac{p!}{3!(p-3)!}\right)$. However, with NVP/1/1 redundancy selected for the subsystem, three different software versions and three identical hardware components must are chosen.

Model 4: *Hi* can be 1, ..., *m* and *Si* can be 1, ..., $\left(p + \frac{p!}{2!(p-2)!}\right)$. If RB/1/1 redundancy is selected for the subsystem, two different software versions and two identical hardware components must be chosen.

### 5.2. *Initial population*

We set the initial population by randomly generating a set of chromosomes consisting of genes, and calculate their fitness value according to the fitness function.

### 5.3. *Selection*

The chromosomes or population are sorted by their fitness values. The top 85% of the population with high fitness values are selected for the crossover process.

### 5.4. *Crossover*

We randomly select two systems or chromosomes from the current population for crossover, to produce two new chromosomes. Also we randomly select a subsystem for crossover. The positions P1 and P2 are labeled with bold font for crossover.

Example: 1 2 | 1 3 | 1 1 | 3 4
1 1 | 2 3 | 3 5 | 4 4
Random subsystem = 3, P1 = 1, P2 = 2
Results: 1 2 | 1 3 | 3 5 | 3 4
1 1 | 2 3 | 1 1 | 2 4

We select the highest 15% of the population with the maximum fitness values from the current population generation and combine with the best 85% from the crossover to be the next population generation.

### 5.5. *Mutation*

Firstly, the current population generation is sorted by fitness values. Then, each chromosome in the generation except the best 5 % is mutated with a mutation rate which is usually less than 10%. The chromosomes resulted from mutation are combined and considered as the chromosomes in the current population generation.

### 5.6. *Penalty function*

A dynamic penalty function (Ref. 10) is an effective approach to deal with problems with cost constraint.[10] It is applied to the selected chromosomes that violate the constraint (i.e., infeasible solution). For example, if the system cost is not greater than the "*Cost*" constraint, no cost penalty is applied, else the cost penalty would be applied to the objective function. Doing this, the selected infeasible solution search space is explored and considered as local or temporarily solutions which may lead in finding feasible global solutions.

## 6. Numerical Example

Table 5. Available components and their reliability estimates, variances, and costs.

| Inputs | | | | | | | |
|---|---|---|---|---|---|---|---|
| (i, j) | HW $Cost_{ij}$ | HW $\hat{R}_{ij}$ | HW Variance-Factor $\eta_{ik}$ | (i, k) | SW $Cost_{ik}$ | SW $\hat{R}_{ik}$ | SW Variance- Factor $\eta_{ik}$ |
| 11 | 30.0 | 0.995 | 4 | 11 | 30.0 | 0.950 | 3 |
| 12 | 10.0 | 0.980 | 5 | 12 | 10.0 | 0.908 | 2 |
| 13 | 10.0 | 0.980 | 4 | 13 | 20.0 | 0.908 | 4 |
| | | | | 14 | 30.0 | 0.950 | 2 |
| 21 | 30.0 | 0.995 | 2 | 21 | 30.0 | 0.965 | 1 |
| 22 | 20.0 | 0.995 | 3 | 22 | 20.0 | 0.908 | 3 |
| 23 | 10.0 | 0.970 | 1 | 23 | 10.0 | 0.887 | 3 |
| | | | | 24 | 20.0 | 0.908 | 2 |
| 31 | 20.0 | 0.994 | 4 | 31 | 20.0 | 0.978 | 4 |
| 32 | 30.0 | 0.995 | 1 | 32 | 30.0 | 0.954 | 1 |
| 33 | 100.0 | 0.992 | 2 | 33 | 20.0 | 0.860 | 2 |
| | | | | 34 | 30.0 | 0.954 | 3 |
| 41 | 30.0 | 0.990 | 2 | 41 | 20.0 | 0.950 | 1 |
| 42 | 10.0 | 0.980 | 4 | 42 | 10.0 | 0.908 | 2 |
| 43 | 10.0 | 0.985 | 1 | 43 | 20.0 | 0.910 | 3 |
| | | | | 44 | 20.0 | 0.950 | 7 |
| 51 | 30.0 | 0.995 | 3 | 51 | 30.0 | 0.905 | 2 |
| 52 | 20.0 | 0.980 | 10 | 52 | 20.0 | 0.967 | 8 |
| 53 | 30.0 | 0.995 | 1 | 53 | 10.0 | 0.967 | 1 |
| | | | | 54 | 30.0 | 0.905 | 5 |
| 61 | 30.0 | 0.998 | 3 | 61 | 10.0 | 0.908 | 1 |
| 62 | 20.0 | 0.995 | 4 | 62 | 30.0 | 0.968 | 2 |
| 63 | 20.0 | 0.994 | 2 | 63 | 20.0 | 0.968 | 3 |
| | | | | 64 | 20.0 | 0.955 | 2 |

We select the problem originally solved by Wattanapongsakorn et al [3] to provide an example problem considering the reliability estimate and variance of reliability estimate as multiple objectives. This example reliability optimization problem is a series system with six subsystems, having $n = 6$, $m_i = 3$, and $p_i = 4$. As an extension of

the previous work, the known component reliabilities used in the previous paper are now considered as reliability estimates with an associated variance. The component costs are unchanged and considered in this optimization problem. Table 5 shows the reliability estimate and its variance as well as cost of all the available components.

We apply this input data to our four-optimization models with various system design cost constraints at 180, 320, 460 and also with unlimited cost constraint. The penalty value (Variance Penalty), which is the weight of the variance of reliability estimate, is set to 1.0 for various cost constraints, and is set arbitrarily to 0.1, 1, 2, 3, 4, 5, and 10 for the system design cost constraint at 460. Other design conditions are Prv = 0.004, Pall = 0.005 and Pd = 0.002. Their corresponding variance-factors ($\eta$) are all equal to 20. The system design constraint is the maximum cost of 460. We apply a genetic algorithm as our optimization approach. The simulation results from our four optimization models are presented in Tables 6-10.

From the GA results presented in Table 6 at various system cost constraints, we can see that with no cost constraint, each model can offer the highest system reliability estimate and the lowest variance of the reliability estimate compared to all the solutions with low or high cost constraints. With a very tight cost constraint, where cost = 180, the best possible obtained solutions are not as good as when the cost constraint is relaxed to 320 or 460. Models 2, 3 and 4 cannot find solutions with component redundancy, thus resulting to the same solution as obtained by model 1. The selected component allocations are depicted in Table 7. Better solution means the solution with higher reliability estimate and lower variance of the reliability estimate.

Model 1 (with no redundancy) offers system reliability estimate equal to 0.771324 and its variance equal to 0.057973 at cost constraints 320, 460 or unlimited. With available component redundancy, models 2-4 can find better solutions than those offered by model 1, resulting in higher system reliability estimates and lower variances. At a certain cost constraint, model 4 (with RB redundancy) offers the best results compared to those offered by the other models.

The corresponding cost for each component allocation result is displayed at Table 8. Each solution cannot have cost exceed the constraints. Compared to all the models with no cost constraint, model 3 with NVP/1/1 redundancy offers its solution (with cost = 810) which is much more expensive than from any other models, because more resources are required.

Table 6. Results from GA with variance penalty = 1.0.

| Model | Estimate | Cost = 180 | Cost = 320 | Cost = 460 | Cost = Unlimited |
|---|---|---|---|---|---|
| 1: No Redundancy | $E[\hat{R}(x)]$ | 0.634367 | 0.771324 | 0.771324 | 0.771324 |
| | $Var(\hat{R}(x))$ | 0.085324 | 0.057973 | 0.057973 | 0.057973 |
| 2: NVP/0/1 | $E[\hat{R}(x)]$ | 0.634367 | 0.768547 | 0.815997 | 0.822285 |
| | $Var(\hat{R}(x))$ | 0.085324 | 0.032869 | 0.027585 | 0.024927 |
| 3: NVP/1/1 | $E[\hat{R}(x)]$ | 0.634367 | 0.796081 | 0.810220 | 0.845683 |
| | $Var(\hat{R}(x))$ | 0.085324 | 0.056880 | 0.025897 | 0.018535 |
| 4: RB/1/1 | $E[\hat{R}(x)]$ | 0.634367 | 0.864831 | 0.909946 | 0.914409 |
| | $Var(\hat{R}(x))$ | 0.085324 | 0.019187 | 0.006175 | 0.005879 |

Table 7. Component allocations for the results shown in Table 6.

| | Subsystem 1 | | Subsystem 2 | | Subsystem 3 | | Subsystem 4 | | Subsystem 5 | | Subsystem 6 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HW | SW | HW | SW | HW | SW | HW | SW | HW | SW | HW | SW |
| Cost = 180 Models 1-4 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 4 | 2 | 3 | 2 | 3 |
| Cost = 320: | | | | | | | | | | | | |
| Model 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 4 | 1 | 2 | 2 | 3 |
| Model 2 | 2 | 1,2,4 | 2 | 1,2,3 | 1 | 1 | 2 | 4 | 1 | 2 | 2 | 3 |
| Model 3 | 1 | 1 | 2 | 1 | 1 | 1 | 3 | 1,2,4 | 1 | 2 | 2 | 3 |
| Model 4 | 2 | 2,3 | 2 | 2,3 | 1 | 1 | 2 | 2,4 | 2 | 2,3 | 2 | 3 |
| Cost = 460: | | | | | | | | | | | | |
| Model 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 4 | 1 | 2 | 2 | 3 |
| Model 2 | 1 | 1,2,4 | 2 | 1,2,4 | 1 | 1 | 1 | 1,3,4 | 1 | 2 | 2 | 2,3,4 |
| Model 3 | 2 | 1,2,3 | 3 | 1,2,3 | 1 | 1 | 3 | 1,2,4 | 1 | 2 | 2 | 1,3,4 |
| Model 4 | 2 | 1,4 | 2 | 1,3 | 1 | 1,4 | 2 | 1,4 | 2 | 2,3 | 2 | 3,4 |
| Cost = Unlimited: | | | | | | | | | | | | |
| Model 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 4 | 1 | 2 | 2 | 3 |
| Model 2 | 1 | 1,3,4 | 2 | 1,2,4 | 1 | 1,2,4 | 1 | 1,3,4 | 1 | 2,3,4 | 2 | 2,3,4 |
| Model 3 | 1 | 1,3,4 | 3 | 1,2,4 | 2 | 1,2,4 | 3 | 1,3,4 | 3 | 2,3,4 | 2 | 2,3,4 |
| Model 4 | 1 | 1,4 | 2 | 1,2 | 1 | 1,4 | 2 | 1,4 | 1 | 2,3 | 2 | 2,3 |

Table 8. Costs of the solutions at various cost constraints.

| Model | Cost Constraint | | | |
|---|---|---|---|---|
| | 180 | 320 | 460 | Unlimited |
| 1. No Redundancy | 180 | 290 | 290 | 290 |
| 2. NVP/0/1 | 180 | 320 | 460 | 570 |
| 3. NVP/1/1 | 180 | 320 | 460 | 810 |
| 4. RB/1/1 | 180 | 320 | 460 | 540 |

To see the effect and relationship of system reliability estimate and its variance, we apply another parameter called variance penalty to the objective function that we intend to maximize. The unchanging goal is to find the best solution with the highest reliability estimate and the lowest variance of the reliability estimate. This is a multi-objective function formulated in each of the models 1-4 that we aim to maximize. When one objective is more important that the other objective, a certain solution would be obtained correspondingly.

The value of the variance penalty was varied; 0.1, 1, 2, 3, 4, 5 and 10. Greater variance penalty indicates higher effect to the obtained solutions, since a large value will be subtracted from the objective function to be maximized. Table 9 presents the GA results obtained for all the models with cost constraint 460 at each variance penalty. From the table, each model offers its highest system reliability estimate when variance penalty is the lowest at 0.1. However, the corresponding variances of the reliability estimates are relatively high. With higher variance penalty, each model tends to seek solutions with smaller variances of the reliability estimates; however at the same time unavoidably causing the lower value of the obtainable reliability estimates. This dependent relationship of reliability estimate and its variance is very significant and attractive. Different solution or system design is preferred when the variance penalty is varied. We can afford system design with high variance of reliability estimate if the reliability estimation uncertainty (or variance) is not a critical issue. Otherwise, for some systems that we cannot afford the uncertainty, we have the pay the price with lower reliability estimate in order to minimize the variance.

Table 9. Results from GA with various variance penalty at cost = 460.

| Variance Penalty | Estimate | Model 1 No Redundancy | Model 2 NVP/0/1 | Model 3 NVP/1/1 | Model 4 RB/1/1 |
|---|---|---|---|---|---|
| 0.1 | $E[\hat{R}(x)]$ | 0.772099 | 0.820891 | 0.818733 | 0.909946 |
| | $Var(\hat{R}(x))$ | 0.060387 | 0.050840 | 0.028837 | 0.006175 |
| 1 | $E[\hat{R}(x)]$ | 0.771324 | 0.815997 | 0.810220 | 0.909946 |
| | $Var(\hat{R}(x))$ | 0.057973 | 0.027585 | 0.025897 | 0.006175 |
| 2 | $E[\hat{R}(x)]$ | 0.771324 | 0.815997 | 0.810220 | 0.909946 |
| | $Var(\hat{R}(x))$ | 0.057973 | 0.027585 | 0.025897 | 0.006175 |
| 3 | $E[\hat{R}(x)]$ | 0.771324 | 0.813901 | 0.810220 | 0.909946 |
| | $Var(\hat{R}(x))$ | 0.057973 | 0.026650 | 0.025897 | 0.006175 |
| 4 | $E[\hat{R}(x)]$ | 0.771324 | 0.813901 | 0.810220 | 0.908700 |
| | $Var(\hat{R}(x))$ | 0.057973 | 0.026650 | 0.025897 | 0.005863 |
| 5 | $E[\hat{R}(x)]$ | 0.771324 | 0.813901 | 0.810220 | 0.908700 |
| | $Var(\hat{R}(x))$ | 0.057973 | 0.026650 | 0.025897 | 0.005863 |
| 10 | $E[\hat{R}(x)]$ | 0.707602 | 0.813901 | 0.805867 | 0.908700 |
| | $Var(\hat{R}(x))$ | 0.047696 | 0.026650 | 0.025097 | 0.005863 |

## 7. Summary and Conclusion

This paper analyzes and identifies system design choices when component reliability information is available in the forms of reliability estimate and variance of the reliability estimate. The system design objectives are to maximize the system reliability estimate and at the same time minimize its variance. These multiple objectives are in contrast of one another. When one objective is more importance than another one, a certain design choice is suggested. Therefore the system design decision depends on the degree of importance of the objective function.

This is probably the first time that a technique to optimize reliability of system using multiple software versions with different reliabilities and correlated failures is proposed, as indicated the non-existence by Gutjahr et al.[8] We believe that our reliability optimization of redundant systems consist of realistic assumption of failure correlation between/among software versions.

We provide four practical optimization models for embedded system design considering no redundancy (model 1) and with redundancy using NVP architectures (models 2-3) and RB architectures (model 4). In our approach, we consider redundant software with failure correlation. We apply a genetic algorithm with dynamic penalty function to solve our case-study optimization problem, providing satisfying results.

## References


1. J.-C. Laprie, J. Arlat, C. Beounes, and K. Kanoun, Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures, *IEEE Computer*, July (1990), pp. 39-51.
2. M. R. Lyu, Editor in Chief, *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, McGraw-Hill (1996).
3. N. Wattanapongsakorn, S. P. Levitan, "Reliability Optimization Models for Fault-Tolerant Distributed Systems," *Proceeding of Annual Reliability & Maintainability Symposium* (2001), pp. 193-199.
4. R. Rubinstein, G. Levitin, A. Lisnianski, H. Ben-Haim, "Redundancy Optimization of Static Series-Parallel Reliability Models Under Uncertainty," *IEEE Transactions on Reliability*, vol. 46, no. 4 (1997), pp. 503-511.
5. T. Jin, D. W. Coit, "Variance of System Reliability Estimates with Arbitrarily Repeated Components," *IEEE Trans on Reliability*, vol. 50, no. 4 (2001), pp. 409-413.
6. D. W. Coit, T. Jin, "Multi-Criteria Optimization: Maximization of a System Reliability Estimate and Minimization of the Estimated Variance," *Proceedings of the 2001 European Safety & Reliability International Conference* (ESREL) (2001).
7. D. W. Coit, T. Jin, N. Wattanapongsakorn, "System Optimization Considering Component Reliability Estimation Uncertainty: A Multi-Criteria Approach," accepted to *IEEE Transaction on Reliability* (2002).

8. W. J. Gutjahr, G. Uchida, "A Branch-and-Bound Approach to the Optimization of Redundant Software Under Failure Correlation," *Computers and Operations Research*, 29 (2002) pp. 1773-1791.
9. L.L. Pullum, *Software Fault Tolerance Techniques and Implementation*, Artech House (2001).
10. D. Coat, A. Smith and D. Tate, "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems," *INFORMS Journal on Computing*, vol. 8, no. 2, Spring (1996), pp. 173-182.

# Optimizing Software System Design with Recovery Blocks Considering Reliability Estimation Uncertainty

N. Wattanapongsakorn
King Mongkut's University of Technology Thonburi, Bangkok, Thailand
D. W. Coit
Rutgers University, New Jersey, USA

ABSTRACT: In this paper, we consider software system optimization design with Recovery Blocks considering uncertainty in component reliability estimate. The design objective is to maximize an estimate of system reliability and also minimize the variance of the reliability estimate. Recovery Block is one of the most common fault-tolerant system architectures, and it is applied for system redundancy, if needed, in this system optimization research effort. We present an optimization model where the system has a serial configuration, and each of the subsystems has choices of with or without RB/1/1 redundancy; a single software fault and a single hardware fault are tolerated. The model is designed under cost constraints. Our model can be easily applied for other types of fault-tolerant system architecture. This is the first time that a technique to optimize reliability of a system using multiple software versions with different reliabilities and correlated failures is proposed. We believe that our reliability optimization of redundant systems consists of realistic assumptions of failure correlation between/among software versions.

## 1 INTRODUCTION

In software system design, very often the information of available components, such as component reliability, is not known but can be approximated with a degree of uncertainty. This is the case, particularly when the system consists of new components with few failure data recorded. Therefore the system/component reliability is uncertain and can only be approximated. Mean and variance of the system/component reliability estimate are considered as reasonable parameters to represent the reliability estimate and its confidence interval (Coit & Jin 2001). Selecting components with high reliability uncertainty would result in a designed system with high reliability uncertainty. This is undesirable because system designers and users seek an optimal system design choice with high reliability estimate, while the reliability uncertainty is low.

This paper describes an optimization model for software system design with recovery blocks considering reliability estimation uncertainty. Recovery Block (RB) (Laprie et al 1990) is one of the most common fault-tolerant software system architectures, where component redundancy is applied. This model is an extended work from Wattanapongsakorn and Levitan (2001) where component reliability is exact and known.

We consider software systems that consist of both software and hardware components. Failures of software components/versions are the major causes of system failures. To provide fault-tolerance to a system, component redundancy is one of the most common approaches. Thus, multiple software versions and hardware components are considered in our optimization model.

Unlike most research papers, we provide a technique to optimize system reliability considering software versions with different reliabilities and correlated failures. For many experimental studies, multiple software versions, which are functionally equivalent, do have failure correlation even if they have been independently developed (Dugan 1994, Eckhardt et al. 1985 & Eckhardt et al. 1991). The failure correlation may come from faults in the software specification, faults from the voting algorithm, and or related faults from any two software versions. Our approach considers this correlation of failures in formulating our optimization model.

The systems that we model are series-parallel fault-tolerant systems. The redundancy allocation problem for series-parallel systems is known to be difficult (NP-hard). Many researchers have proposed a variety of approaches to solve this problem using, for example, integer programming, branch-and-bound, dynamic programming, mixed integer and nonlinear programming. Recent optimization approaches are based on heuristics such as Genetic Algorithms (GA), and Tabu Search (TS). All of these approaches were developed for either optimizing re-

liability for software systems or hardware systems. In this paper, we consider systems consisting of both software and hardware components. The software failure behavior, which is different from the hardware failure behavior, is considered.

GA is used as the optimization approach. The term 'genetic' derives from the roughly analogous natural re-producing new-born population by crossover and mutation. There are competitions among the population; the stronger ones will survive to the next generation and the weak ones will soon die out. GA is a heuristic optimization model that has been applied effectively to solve many difficult problems in different fields such as scheduling, facility layout, and graph coloring/ graph partitioning problems. It is a stochastic algorithm with performance depending on the solution encoding, crossover breeding operator, elitist selection and mutation operator.

Our optimization model is developed to select both software and hardware components and the degree of redundancy to optimize the overall system reliability, with a total cost constraint. In the system, there are a specified number of subsystems in series. For each subsystem, there are several hardware and software choices to be made. The system is designed using components, each with estimated reliability, but with known cost.

This paper is organized as follows. The assumption and notation used in this paper are listed next. In section 2, the software system design with recovery block architecture is discussed. Section 3 provides the concept of reliability estimation uncertainty. Section 4 presents our optimization model to maximize reliability considering uncertainty. Section 5 discusses the GA as our optimization approach. In section 6, we demonstrate our model with an example system, where reasonable and interesting results are obtained and discussed.

## ASSUMPTIONS

1) Each software component, hardware component or the system has two states: functional or failed
2) Reliability of each software or hardware component is unknown, but estimable
3) There is no failure repair for each component or system
4) Hardware redundancy is in active mode (i.e. hot spares)
5) Failure of individual hardware components are statistically independent

## NOTATION

RB/i/j system architecture RB with $i$ hardware faults tolerated and $j$ software faults tolerated
$n$ Number of subsystems in the series system
$m_i$ Number of hardware component types available for subsystem $I$
$p_i$ Number of software versions available for subsystem $I$
$R$ Estimated reliability of the distributed system
$R_i$ Estimated reliability of the subsystem $I$
$Rhw_{ij}$ Reliability of hardware component $j$ at subsystem $i$
$Rsw_{ik}$ Reliability of software component $k$ at subsystem $i$
$Chw_{ij}$ Cost of using hardware component $j$ at subsystem $i$
$Csw_{ik}$ Cost of developing software version $k$ at subsystem $i$
Cost Maximum allowable cost (constraint)
Px Probability that event $X$ occurs
Qx Probability that event $X$ does not occur; Qx = 1 – Px
$Pv_i$ Probability of failure of software version $I$
$Prv_{ij}$ Probability of failure from related fault between two software versions, $i$ and $j$
$P_{all}$ Probability of failure from related fault among all software versions, due to faults in specification
Pd Probability of failure of decider or voter
$Ph_i$ Probability of failure of hardware component $i$. If only one hardware is applied, $Ph_i = Ph$ for all $i$

## 2 SOFTWARE SYSTEM DESIGN WITH RECOVERY BLOCKS

### 2.1 *Recovery Block (RB): RB/1/1 Architecture*

The RB model consists of an adjudication module called an *acceptance test*, and at least two software components called alternates (Laprie et al. 1990, Lyu 1996), as indicated in Figure 1. At the beginning, the output of the first or primary alternate is tested for acceptance. If it fails, the process will *roll back* to the beginning of the process, and then let the second alternate execute and test its output for acceptance again. This process continues until the output from an alternate is accepted or all outputs of the alternates have been tested and fail.

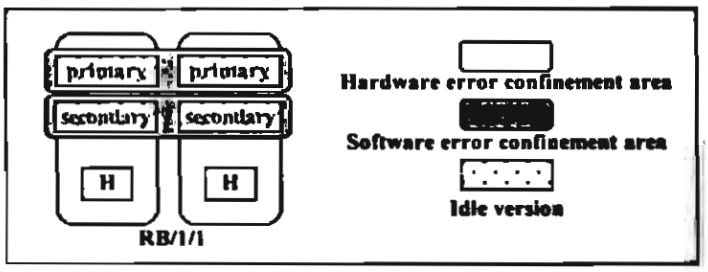


Figure 1. RB/1/1 fault-tolerant architecture.

RB/1/1 has a single hardware fault tolerated and a single software fault tolerated. The system consists of two hardware components, each running two independent software versions; primary and secondary. The primary version is active until it fails, and the secondary version is the backup spare. System failures occur when both ver

ions fail, or both hardware components fail. The probability that an unacceptable result occurs during a single ask iteration, P is presented by Equation 1 where $a_i$, $k_{ij}$ nd $h_{ij}$ values for RB/1/1 architecture are listed in Table 1.

$$P = \sum_{i=1}^{s}\left( a_i \prod_{j \in C_i} p_j^{k_{ij}} q_j^{h_{ij}} \right) \qquad (1)$$

where,

$s$ = number of additive terms when all failure probabilities have been enumerated; $s = 6$ for this RB1/1 architecture

$a_i$ = integer coefficient

$C_i$ = component type set for $i^{th}$ additive term

$k_{ij}$ = power coefficient for $j^{th}$ component reliability in set $C_i$

$h_{ij}$ = power coefficient for $j^{th}$ component unreliability in set $C_i$

$p_j$ = unreliability of $j^{th}$ type of component

$q_j$ = reliability of $j^{th}$ type of component, $p_j + q_j = 1$ for all $j$

$p_j$ and $q_j$ definitions and comparisons with notation from Wattanapongsakorn and Levitan (2001) are as follows,

| | |
|---|---|
| $p_1$=Prv | $q_1$=Qrv |
| $p_2$=Pd | $q_2$=Qd |
| $p_3$=$P_{all}$ | $q_3$=$Q_{all}$ |
| $p_4$=$Pv_1$ | $q_4$=$Qv_1$ |
| $p_5$=$Pv_2$ | $q_5$=$Qv_2$ |
| $p_6$=Ph | $q_6$=Qh |

Table 1. $a_i$, $k_{ij}$ and $h_{ij}$ expressed in a matrix form for RB/1/1 architecture.

| | $k_{ij}$ | | | | | | | $h_{ij}$ | | | | | | $a_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i | j = 1 | 2 | 3 | 4 | 5 | 6 | i | j = 1 | 2 | 3 | 4 | 5 | 6 | |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 1 | 1 | 0 | 5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 1 | 1 | 2 | 6 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

With RB/1/1 architecture, we develop an optimization model for a fault-tolerant embedded system considering reliability estimates with uncertainty. The components, which are available for the system design, each has reliability estimation uncertainty measured by the variance of the reliability estimate.

In the next section, we formulate equations for system reliability estimate and variance for the reliability estimate for the RB/1/1 fault-tolerant architecture. These equations will be used in our optimization model, discussed later in section 4.

## 3 RELIABILITY ESTIMATION UNCERTAINTY

Usually the exact component unreliability, $p_j$, or reliability, $q_j$, is not known. They are estimated from life test data or field failure records. The estimated $\hat{p}_j$ or $\hat{q}_j$ are used to replace the true but unknown in Equation 1.

$$\hat{P} = \sum_{i=1}^{s}\left( a_i \prod_{j \in C_i} \hat{p}_j^{k_{ij}} \hat{q}_j^{h_{ij}} \right) \qquad (2)$$

Direct calculation of $E[\hat{P}]$ and $Var(\hat{P})$ are difficult due to the coupling of $\hat{p}_j$ and $\hat{q}_j$. Therefore, Equation 2 has been rearranged, as follows.

$$\hat{P} = \sum_{i=1}^{s}\left( a_i \prod_{j \in C_i} (1-\hat{q}_j)^{k_{ij}} \hat{q}_j^{h_{ij}} \right) \qquad (3)$$

Equation 3 can be rearranged by expanding $(1-\hat{q}_j)^{k_{ij}}$ terms, resulting in Equation 4.

$$\hat{P} = \sum_{i=1}^{t}\left( b_i \prod_{j \in C_i} \hat{q}_j^{n_{ij}} \right) \qquad (4)$$

where

$t$ = number of additive terms after expansion, $t > s$

$b_i$ = integer coefficient

$t$ is the number of terms after the expansion. $b_i$ and $n_{ij}$ are determined by grouping similar terms. This expansion procedure is conducted automatically using Matlab code based on the parameters in Table 1. Due to the length of the expansion, detailed computational procedure is omitted. Table 2 lists all the expansion results.

Table 2. $n_{ij}$ and $b_i$ Expressed in a matrix form for RB/1/1 architecture.

| | $n_{ij}$ | | | | | | $b_i$ |
|---|---|---|---|---|---|---|---|
| i | j =1 | 2 | 3 | 4 | 5 | 6 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 6 | 1 | 1 | 1 | 0 | 0 | 0 | -1 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
| 8 | 1 | 1 | 0 | 0 | 0 | 0 | -1 |
| 9 | 1 | 1 | 1 | 0 | 0 | 0 | -1 |
| 10 | 1 | 1 | 1 | 0 | 0 | 1 | -2 |
| 11 | 1 | 1 | 1 | 0 | 0 | 2 | 1 |
| 12 | 1 | 1 | 1 | 1 | 0 | 0 | -1 |
| 13 | 1 | 1 | 1 | 0 | 1 | 0 | -1 |
| 14 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 15 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 16 | 1 | 1 | 1 | 0 | 0 | 1 | 2 |
| 17 | 1 | 1 | 1 | 0 | 0 | 2 | -1 |
| 18 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 19 | 1 | 1 | 1 | 1 | 1 | 0 | -1 |
| 20 | 1 | 1 | 1 | 1 | 0 | 1 | -2 |
| 21 | 1 | 1 | 1 | 1 | 0 | 2 | 1 |
| 22 | 1 | 1 | 1 | 0 | 1 | 1 | -2 |
| 23 | 1 | 1 | 1 | 0 | 1 | 2 | 1 |
| 24 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 25 | 1 | 1 | 1 | 1 | 1 | 2 | -1 |

From the table, $t = 25$. Based on the coefficients $n_{ij}$, $b_i$ and component reliability information, Equation 4 can be used to obtain the mean and the vari-

ance of unreliability $\hat{P}$. Together with the higher-order moment information of component reliability estimates, the mean and the variance of unreliability $\hat{P}$, can be obtained as follows (Jin & Coit 2001).

$$E[\hat{P}] = \sum_{i=1}^{I}\left(b_i \prod_{j \in C_i} E[\hat{q}_j^{n_j}]\right) \tag{5}$$

$$Var(\hat{P}) = \left(\sum_{i=1}^{I}\left\{b_i^2\left(\prod_{j=C_i} E[\hat{q}_j^{2n_j}] - \prod_{j=C_j}\left(E[\hat{q}_j^{n_j}]\right)^2\right)\right\} + 2\sum_{i<m}^{I}\left\{b_i b_m\left(\prod_{j=C_{im}} E[\hat{q}_j^{n_j+n_m}] - \prod_{j=C_{im}} E[\hat{q}_j^{n_j}]E[\hat{q}_j^{n_m}]\right)\right\}\right) \tag{6}$$

where $C_{im} = C_i \cup C_j$

To show the relationship of reliability estimate and variance of reliability estimate for each component, we provide a few numerical examples. Table 3 lists component reliability estimates or unreliability estimate values. These data are selected directly from Wattanapongsakorn and Levitan (2001). Equations. 5 & 6 are also valid, as long as high moments of component reliability estimates are known. Without loss of generality, it is assumed Bernoulli test and applied binomial distribution theory was used to estimate high moments (Jin & Coit 2001). $\eta = [\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6, \eta_7]$ is a variance-factor vector, and $\eta_i$ = integer. For example, when $\eta = [6, 4, 2, 2, 4, 3, 6]$, the corresponding component variances are given in Table 3.

Table 3. Parameters and definition of component's variance of reliability estimate.

| Unreliability Estimate | Reliability Estimate | Variance of Reliability Estimate |
|---|---|---|
| Prv=0.004 | Qrv=0.996 | Var(Prv)= (Prv× Qrv)/$\eta_1$ = 0.000664 |
| Pd=0.02 | Qd=0.98 | Var(Pd) = (Pd× Qd)/$\eta_2$ =0.0049 |
| $P_{all}$=0.005 | $Q_{all}$=0.995 | Var($P_{all}$) =0.0024875 |
| $Pv_1$=0.035 | $Qv_1$=0.965 | Var($Pv_1$)= ($Pv_1$× $Qv_1$)/$\eta_4$ =0.016 |
| $Pv_2$=0.046 | $Qv_2$=0.954 | Var($Pv_2$)= ($Pv_2$× $Qv_2$)/$\eta_5$ = 0.012 |
| $Pv_3$=0.09 | $Qv_3$=0.910 | Var($Pv_3$)= ($Pv_3$× $Qv_3$)/$\eta_6$ = 0.03 |
| Ph=0.03 | Qh=0.970 | Var(Ph)= (Ph× Qh)/$\eta_7$ =0.004 |

Table 4 lists four results with respect to different component variance. It is shown that system unreliability is unchanged even if component variances vary as $\eta$ changes. It can be observed that as component variances become small, the overall variance of the system unreliability estimate $\hat{P}$ also decreases.

Table 4. Parameters of components and system unreliability $P$, system unreliability estimate $E[\hat{P}]$, and system variance of unreliability estimate $Var(\hat{P})$.

| $\eta$ | $P$ | $E[\hat{P}]$ | $Var(\hat{P})$ |
|---|---|---|---|
| [1, 1, 1, 1, 1, 1, 1] | 0.05571 | 0.03716 | 0.03578 |
| [2, 2, 2, 2, 2, 2, 2] | 0.05571 | 0.06317 | 0.02801 |
| [8, 8, 8, 8, 8, 8, 8] | 0.05571 | 0.06068 | 0.00777 |
| [12, 12, 12, 12, 12, 12, 12] | 0.05571 | 0.05925 | 0.00521 |

## 4 OPTIMIZATION MODEL

In this section, we present our optimization model for reliability of software systems. The objective is to find the optimal set of software and hardware allocations for all subsystems (with or without RB/1/1 redundancy). The problem formulation is to maximize the system reliability estimate, subjected to a specified cost constraint, *Cost*. The system has all subsystems connected in series. This model is suited for systems that handle relatively critical tasks. The problem formulation for this model allow each subsystem to have RB/1/1 redundancy allocation as its reliability estimate and variance of the reliability estimate, calculated according to the RB/1/1 redundancy configuration. The parameters considered for the reliability of the RB/1/1 architecture are available as component reliability estimate and variance of reliability estimate. Each allocated software version is allowed to have a different reliability estimated value, unlike several proposed models when all of the software versions have the same reliability value (Lyu, 1996).

The problem formulation is to maximize system reliability estimate with its variance of reliability estimate by choosing the optimal set of hardware and software components for each subsystem by:

$$Max\left\{\hat{R}(x) - penalty\left(Var\left(\hat{R}(x)\right)\right)\right\}$$

Subject to $\left(\sum_{j=1}^{m_1} X_{ij} = 1 \text{ and } \sum_{k=1}^{p_1} Y_{ik} = 1\right) or \left(\sum_{j=1}^{m_1} X_{ij} = 2 \text{ and } \sum_{k=1}^{p_1} Y_{ik} = 2\right)$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij}C_{ij} + \sum_{i=1}^{n}\sum_{k=1}^{p_i} Y_{ik}C_{ik} \le Cost$$

$X_{ij} = 0,1,2 \quad Y_{ij} = 0,1 \quad i = 1,2,\ldots,n \quad j = 1,2,\ldots,m_i \quad k = 1,2,\ldots$

where $\hat{R}(x) = \prod_{i=1}^{n}\left(\hat{R}_i(x)\right)$

$Var\left(\hat{R}(x)\right) = Eq.\ (8)$

$$\left.\begin{array}{l}\hat{R}_i(x) = \sum_{j=1}^{m_i}\sum_{k=1}^{p_i} X_{ij}Y_{ik}\hat{R}hw_{ij}\hat{R}sw_{ik} \\ Var\left(\hat{R}_i(x)\right) = Eq.\ (9)\end{array}\right\} \text{ if } \sum_{j=1}^{m_i} X_{ij} = 1, \ \sum_{k=1}^{p_i} Y_{ik} =$$

$$\left.\begin{array}{l}\hat{R}_i(x) = 1 - Eq.\ (5) \\ Var\left(\hat{R}_i(x)\right) = Eq.\ (6)\end{array}\right\} \text{ if } 2\sum_{j=1}^{m_i} X_{ij} = 2 \quad \sum_{k=1}^{p_i} Y_{ik} = 2$$

The design objective is to identify solutions wi very high reliability, but also with a low variance the reliability estimate. If the decision maker is ris neutral, then the design objective is to maximize t reliability estimate. If the person is risk-aver where the worst case with high variance of the re ability estimate is unacceptable (i.e., highly critic & life-dependent systems), then minimizing the va ance is also an important design objective. O approach is to consider the problem as a mul criteria optimization: to maximize the system re ability estimate and at the same time minimize t variance. This approach was proposed by Coit a

Jin (2001). Another approach, which is our approach, is to penalize the estimation uncertainty by penalizing the system reliability estimate with its estimation variance. The '*penalty*' is a tunable parameter based on a system designer's tolerance for risk, i.e. actual reliability deviation from the estimate. By penalizing the variance, the final solution represents a compromise between high reliability and low variance.

## 5 GENETIC ALGORITHM OPTIMIZATION APPROACH

GA requires that the system design (phenotype) be encoded as a solution vector (genotype). Then, genetic operators (crossover, mutation) are applied over successive generations until the GA converges to a solution or a pre-determined maximum number of generations is reached.

### 5.1 *Encoding*

For an embedded (hardware and software) system with $n$ subsystems connected in series, the string encoding can be represented as :

H1 S1 | H2 S2 | H3 S3 | ... | Hn Sn

where $Hi$, with $0 \leq i \leq n$ is the selected hardware component for subsystem $i$, and $Si$ is the selected software component/version for the specified subsystem.

### 5.2 *Initial population*

We set the initial population by randomly generating a set of chromosomes consisting of genes, and calculate their fitness value according to the fitness function.

### 5.3 *Selection*

The chromosomes or population are sorted by their fitness values. The top 85% of the population with high fitness values are selected for the crossover process.

### 5.4 *Crossover*

We randomly select two systems or chromosomes from the current population for crossover, to produce two new chromosomes. Also we randomly select a subsystem for crossover. The positions P1 and P2 are labeled with bold font for crossover.

Example: 1 2 | 1 3 | 1 1 | 3 4
1 1 | 2 3 | 3 5 | 4 4
Random subsystem = 3, P1 = 1, P2 = 2
Results: 1 2 | 1 3 | **3 5** | 3 4
1 1 | 2 3 | **1 1** | 2 4

We select the highest 15% of the population with the maximum fitness values from the current population generation and combine with the best 85% from the crossover to be the next population generation.

### 5.5 *Mutation*

The current population generation is initially sorted by fitness values. Then, each chromosome in the generation, except the best 5%, is mutated with a mutation rate which is usually less than 10%. The chromosomes resulted from mutation are combined and considered as the chromosomes in the current population generation.

### 5.6 *Penalty function*

A dynamic penalty function is an effective approach to deal with problems with cost constraint (Coit et al. 1996). It is applied to the selected chromosomes that violate the constraint (i.e. infeasible solution). For example, if the system cost is not greater than the "*Cost*" constraint, no cost penalty is applied, else the cost penalty would be applied to the objective function. Doing this, the selected infeasible solution search space is explored and considered as local or temporary solutions which may lead in finding feasible global solutions.

## 6 AN EXAMPLE SYSTEM: DESIGN AND SIMULATION RESULT

We select the problem originally solved in Wattanapongsakorn and Levitan (2001) to provide an example problem considering the reliability estimate and variance of reliability estimate as multiple objectives. This example reliability optimization problem is a series system with six subsystems, having $n = 6$, $m_i = 3$, and $p_i = 4$. As an extension of the previous work, the known component reliabilities used in the previous paper are now considered as reliability estimates with an associated variance. The component costs are unchanged and considered in this optimization problem. Table 5 shows the reliability estimate and its variance as well as cost of all the available components.

We apply this input data to our optimization model with various system design cost constraints at 180, 320, 460 and also with unlimited cost constraint. The penalty value (variance penalty), which is the weight of the variance of reliability estimate is set arbitrarily to 0.01, 0.1, 1, 2, 3, 4, 5, 10 and 100 for various system design cost constraint i.e. 180, 320, 460 and unlimited. Other design conditions are Prv = 0.004, Pall = 0.005 and Pd = 0.002, with the corresponding variance-factors ($\eta$) each is equal to 20. We apply a genetic algorithm as our optimization approach. The simulation results, each is based on 10 runs, are presented in Tables 6-11.

Table 5. Available components and their reliability estimates, variances, and costs.

| Inputs | | | | | | | |
|---|---|---|---|---|---|---|---|
| (i, j) | HW $Cost_{ij}$ | HW $\hat{R}_{ij}$ | HW Variance-Factor $\eta_{ik}$ | (i, k) | SW $Cost_{ik}$ | SW $\hat{R}_{ik}$ | SW Variance-Factor $\eta_{ik}$ |
| 11 | 30.0 | 0.995 | 4 | 11 | 30.0 | 0.950 | 3 |
| 12 | 10.0 | 0.980 | 5 | 12 | 10.0 | 0.908 | 2 |
| 13 | 10.0 | 0.980 | 4 | 13 | 20.0 | 0.908 | 4 |
| | | | | 14 | 30.0 | 0.950 | 2 |
| 21 | 30.0 | 0.995 | 2 | 21 | 30.0 | 0.965 | 1 |
| 22 | 20.0 | 0.995 | 3 | 22 | 20.0 | 0.908 | 3 |
| 23 | 10.0 | 0.970 | 1 | 23 | 10.0 | 0.887 | 3 |
| | | | | 24 | 20.0 | 0.908 | 2 |
| 31 | 20.0 | 0.994 | 4 | 31 | 20.0 | 0.978 | 4 |
| 32 | 30.0 | 0.995 | 1 | 32 | 30.0 | 0.954 | 1 |
| 33 | 100.0 | 0.992 | 2 | 33 | 20.0 | 0.860 | 2 |
| | | | | 34 | 30.0 | 0.954 | 3 |
| 41 | 30.0 | 0.990 | 2 | 41 | 20.0 | 0.950 | 1 |
| 42 | 10.0 | 0.980 | 4 | 42 | 10.0 | 0.908 | 2 |
| 43 | 10.0 | 0.985 | 1 | 43 | 20.0 | 0.910 | 3 |
| | | | | 44 | 20.0 | 0.950 | 7 |
| 51 | 30.0 | 0.995 | 3 | 51 | 30.0 | 0.905 | 2 |
| 52 | 20.0 | 0.980 | 10 | 52 | 20.0 | 0.967 | 8 |
| 53 | 30.0 | 0.995 | 1 | 53 | 10.0 | 0.967 | 1 |
| | | | | 54 | 30.0 | 0.905 | 5 |
| 61 | 30.0 | 0.998 | 3 | 61 | 10.0 | 0.908 | 1 |
| 62 | 20.0 | 0.995 | 4 | 62 | 30.0 | 0.968 | 2 |
| 63 | 20.0 | 0.994 | 2 | 63 | 20.0 | 0.968 | 3 |
| | | | | 64 | 20.0 | 0.955 | 2 |

From the GA results presented in Table 6 at various system cost constraints, we can see that with no cost constraint, each model can offer the highest system reliability estimate and the lowest variance of the reliability estimate compared to all the solutions with low or high cost constraints. With a very tight cost constraint, where cost = 180, the best possible obtained solutions are not as good as when the cost constraint is relaxed to 320 or 460. With a more relaxed cost constraint, better solutions can be obtained. The selected component allocations for the corresponding cost constraints are depicted in Table 7. Better solution means the solution with higher reliability estimate and lower variance of the reliability estimate.

From Table 7, at cost 180, no component redundancy can be obtained, indicated by a software version and a hardware component selected for each subsystem. At higher cost constraints, the results show replicated software and hardware components allocated, for examples, at subsystems 1, 2, 4, and 5

Table 6. Optimization results from GA with variance penalty = 1.0.

| Estimate | Cost = 180 | Cost = 320 | Cost = 460 | Cost = Unlimited |
|---|---|---|---|---|
| $E[\hat{R}(x)]$ | 0.632460 | 0.862231 | 0.909249 | 0.914257 |
| $Var(\hat{R}(x))$ | 0.090987 | 0.019648 | 0.006210 | 0.005914 |

The GA optimization results with arbitrary values of variance penalty at system cost constraints equal to 180, 320, 460 and unlimited are depicted in Tables 8, 9, 10, and 11, respectively. From the results at a certain cost constraint, the GA seeks for solutions with less variance of the reliability estimate when the variance penalty is set higher. However these solutions also have lower reliability estimate as well. In other words, system design choice with high reliability estimate also has high variance. The design choice with high reliability estimate can be obtained when the uncertainty or variance of the reliability estimate is affordable i.e. when the variance penalty is not significant.

Table 7. Component allocations for the results shown in Table 6.

| Cost | Subsystem 1 i=1 | | Subsystem 2 i=2 | | Subsystem 3 i=3 | | Subsystem 4 i=4 | | Subsystem 5 i=5 | | Subsystem 6 i=6 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HW j= | SW k= | HW j= | SW k= | HW j= | SW k= | HW j= | SW k= | HW j= | SW k= | HW j= | SW k= |
| 180 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 4 | 2 | 3 | 2 | 3 |
| 320 | 2 | 2,3 | 2 | 2,3 | 1 | 1 | 2 | 2,4 | 2 | 2,3 | 2 | 3 |
| 460 | 2 | 1,4 | 2 | 1,3 | 1 | 1,4 | 2 | 1,4 | 2 | 2,3 | 2 | 3,4 |
| unlimited | 1 | 1,4 | 2 | 1,2 | 1 | 1,4 | 2 | 1,4 | 1 | 2,3 | 2 | 2,3 |

Table 8. Optimization results from GA with various variance penalties at cost = 180.

| Variance Penalty | $E[\hat{R}(x)]$ | $Var(\hat{R}(x))$ |
|---|---|---|
| 0.01 - 0.1 | 0.635687 | 0.090987 |
| 1 - 3 | 0.632460 | 0.085178 |
| 4 | 0.632460 | 0.085178 |
| 5 | 0.632460 | 0.085178 |
| 10 | 0.632460 | 0.085178 |
| 100 | 0.604499 | 0.083446 |

Table 9. Optimization results from GA with various variance penalties at cost = 320.

| Variance Penalty | $E[\hat{R}(x)]$ | $Var(\hat{R}(x))$ |
|---|---|---|
| 0.01 | 0.862231 | 0.019648 |
| 0.1 | 0.862231 | 0.019648 |
| 1 - 3 | 0.862231 | 0.019648 |
| 4 | 0.862231 | 0.019648 |
| 5 | 0.847706 | 0.015310 |
| 10 | 0.847706 | 0.015310 |
| 100 | 0.847706 | 0.015310 |

Table 10. Optimization results from GA with various variance penalties at cost = 460.

| Variance Penalty | $E[\hat{R}(x)]$ | $Var(\hat{R}(x))$ |
|---|---|---|
| 0.01- 0.1 | 0.909249 | 0.006210 |
| 1 - 3 | 0.909249 | 0.006210 |
| 4 | 0.908004 | 0.005898 |
| 5 | 0.908004 | 0.005898 |
| 10 | 0.908004 | 0.005898 |
| 100 | 0.906183 | 0.005796 |

Table 11. Optimization results from GA with various variance penalties at cost = unlimited

| Variance Penalty | $E[\hat{R}(x)]$ | $Var(\hat{R}(x))$ |
|---|---|---|
| 0.01 | 0.914486 | 0.006675 |
| 0.1 | 0.914257 | 0.005914 |
| 1 - 3 | 0.914257 | 0.005914 |
| 4 | 0.914257 | 0.005914 |
| 5 | 0.914257 | 0.005914 |
| 10 | 0.913637 | 0.005779 |
| 100 | 0.909938 | 0.005572 |

## 7 CONCLUSION

This paper analyses and identifies system design choices when component reliability information is available in the forms of reliability estimate and variance of the reliability estimate. The system design objectives are to maximize the system reliability estimate, and at the same time, minimize its variance. These multiple objectives are in contrast with one another. When one objective is more importance than another one, a certain design choice is suggested. Therefore the system design decision depends on the degree of importance of the objective function.

## REFERENCES


Coit D. W. & Jin T. 2001. Multi-Criteria Optimization: Maximization of a System Reliability Estimate and Minimization of the Estimate Variance. *Proceedings of the European Safety & Reliability International Conference* (ESREL), Turin, Italy.

Coit D. W., Smith A & Tate D. 1996. Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems, *INFORMS Journal on Computing*, Vol. 8, NO. 2, pp. 173-182.

Dugan J.B. 1994. Experimental Analysis of Models for Correlation in Multiversion Software, *Proceedings of the International Symposium on Software Reliability Engineering*, Los Alamitos, CA, pp. 36-44.

Eckhardt D.E., Caglayan A.K., Knight J.C., Lee. L.D., McAllister D.F., Vouk M.A. & Kelly, J.P. 1991. An Experimental Evaluation of Software Redundancy as a Strategy for Improving Reliability, *IEEE Transactions on Software Engineering*, Vol. 17, NO. 7, pp. 692-702.

Eckhardt D. E. & Lee. L. D. 1985. A theoretical Basis for the Analysis of Multiversion software Subject to Coincident Errors, *IEEE Transactions on Software Engineering*, Vol. 11 , pp.1511-1517.

Jin T. & Coit D. W. 2001. Variance of System Reliability Estimates with Arbitrarily Repeated Components, *IEEE Trans on Reliability*, Vol. 50, NO. 4, pp. 409-413.

Laprie J.-C., Arlat J., Beounes C. & Kanoun K., July 1990. Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures, *IEEE Computer*, pp. 39-51.

Lyu M. R. (Editor in Chief) 1996. *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, McGraw-Hill.

Rubinstein R., Levitin G., Lisnianski A. & Ben-Haim H. 1997. Redundancy Optimization of Static Series-Parallel Reliability Models Under Uncertainty, *IEEE Transactions on Reliability*, Vol. 46, NO. 4, pp. 503-511.

Wattanapongsakorn N. & Levitan S. P. 2001. Reliability Optimization Models for Fault-Tolerant Distributed Systems, *Proceedings of Annual Reliability & Maintainability Symposium*, pp. 193-199.

# Reliability Optimization for Embedded Systems using Genetic Algorithm

W. Phengprajit and N. Wattanapongsakorn
Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi, Bangkok, 10140, Thailand
Phone 02-470-9083, Fax 02-872-5050, E-mail: kjoriw@yahoo.com

**Abstract**

This article presents Genetic Algorithm (GA) techniques with dynamic penalty function for solving the reliability optimization problem for embedded (hardware and software) systems, considering redundancy techniques with N-Version Programming and Recovery Blocks, and various system cost constraints. This is the first time that GA is applied to optimize to this type of systems, where related faults or dependency in software components/versions are considered. It is our extended work from [6] where Simulated Annealing was applied to the problem with no guarantee for optimal solutions.

GA is a suitable approach since it provides robustness and efficiency in solving combinatorial optimization problems with large, and complex search spaces. The solutions are compared with a non-linear programming technique, where optimal solutions can be obtained and identified.

**Keywords:** Genetic Algorithm, Optimization Techniques, Embedded System, and Reliability Analysis

## 1. Introduction

Critical systems such as spaceships, airplanes, and nuclear power plants require high system reliability. Redundancy techniques are typical approaches to enhance system reliability. For embedded system redundancy can be achieved by applying extra copies of hardware and software components in parallel fashion to share the system workload or prevent some system faults. This technique does not only raise the reliability higher, but also increase the system cost. The problem occurred when there is restricted cost and very high system reliability is required. Hence several researchers attempted to propose the techniques on how to select appropriate components and how to connect/configure them in the system.

Existing optimization approaches, for examples, are integer programming, dynamic programming, mixed integer and non-linear programming and heuristics (such as Genetic Algorithm (GA), and Simulated Annealing (SA)).

In the next section, we give an overview of related works in reliability optimization. Section 3 discusses the embedded system architectures N-Version Programming (NVP) and Recovery Blocks (RB). Section 4 presents the Genetic Algorithm with penalty function concept. Then in section 5, we explain our research methodology in details. Section 6 presents our optimization results with the GA, compared to Lingo [4] results (a non-linear programming software). Lastly, section 7 concludes and summarizes our research.

## 2. Related Works

An early research group, Fyffe et al [1], applied a computational algorithm called dynamic programming, to solve reliability problem in 1968. In 1977, Tillma et al [2] proposed a mixed-integer and nonlinear programming to solve a redundancy allocation problem. Then in 1987, Kuo et al [3] represented a branch-and-bound technique to optimize reliability with Lagrange multiplier technique. All of these works obtained the optimal solutions for the problems. For the problems with large search space, very long computation time was required to get the solution. Thus, many researchers prefer to use heuristic algorithms to solve this kind of problems instead.

Recently, heuristic approaches have received a great attention. For examples, in 1996, Coit and Smith [5] presented Genetic Algorithm to optimize the reliability of series-parallel systems. In 2001, Wattanapongsakorn et al [6] used Simulated Annealing (SA) to optimize software and hardware reliability for fault-tolerant distributed systems considering related faults or system dependency. However, the optimal solutions are not guaranteed by the approach.

One significant advantage of the heuristic algorithms is the efficiency or speed to find feasible solutions. Even though optimal solutions are not guaranteed, the algorithms usually provide good solutions, and often they are optimal solutions.

In this paper, we use GA with dynamic penalty function to find the optimal component allocation for embedded systems with software and hardware components considering redundancy and cost constraints. Comparing the results with the result from Lingo [4], commercial non-linear programming software, we verify that our GA approach can provide optimal solutions. The notations and acronyms used throughout this paper are as follows:

- $P_x$ Probability that event $x$ occurs
- $Q_x$ Probability that event $x$ does not occur; $Q_x = 1-P_x$
- $Pv_i$ Prob. of failure of software version $i$, $Qv_i = 1- Pv_i$
- $Prv_i$ Prob. of failure from related fault between 2 software versions $i$ and $j$, $Qrv_i = 1- Prv_i$
- $P_{all}$ Prob. of failure from related fault among all software versions, due to faults in specification, $Q_{all} = 1- P_{all}$
- $P_d$ Prob. of failure of decider or voter, $Q_d = 1- P_d$
- $Ph_i$ Prob. of failure of hardware component $i$, $Qh_i = 1- Ph_i$

$R$ Estimated reliability of system
$R_i$ Estimated reliability of subsystem $i$
$Rhw_{ij}$ Reliability of hardware component $j$ at subsystem $i$
$Rsw_{ij}$ Reliability of software component $j$ at subsystem $i$
$C$ Cost of system
$Cost$ Affordable Cost
$n$ Number of subsystems within system
$m_i$ Number of hardware component choices available for subsystem $i$
$p_i$ Number of software component choices available for subsystem $i$
$X_{ij}$ If hardware component $j$ at subsystem $i$ is selected $X_{ij} = 1$ else $X_{ij} = 0$
$Y_{ij}$ If software component $j$ at subsystem $i$ is selected $Y_{ij} = 1$ else $Y_{ij} = 0$

## 3. Embedded System Architectures

N-Version Programming and Recovery Blocks are the fundamental fault-tolerant embedded system architectures.

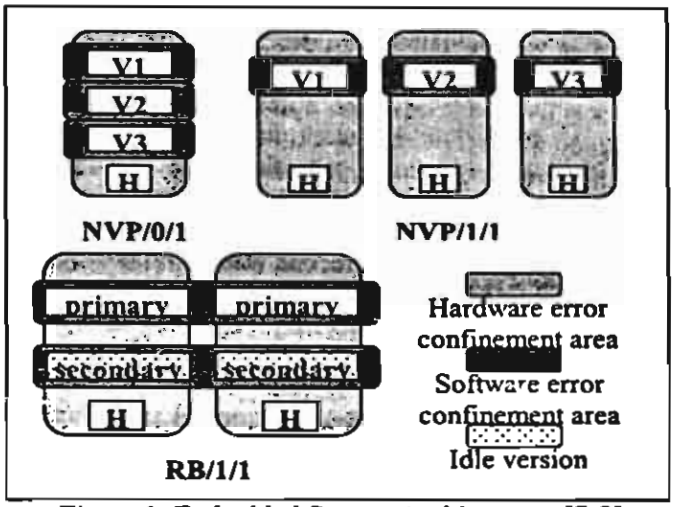


Figure 1: Embedded System Architectures [7-8]

### 3.1 N-version Programming (NVP) Architecture[7-8]

The system consists of parallel execution of $N$ independently developed functionally equivalent software versions with adjudication, called a voter, of their output modules. $N$ is usually an odd number. In this model, all $N$ software versions are executed (on hardware resources) for the same task generally at the same time, and produce their outputs to the voter. Then the voter uses the majority of the outputs to determine the correct or the final output of the system.

#### 3.1.1 NVP/0/1 Architecture

This model consists of three independent software components (or three different software versions) running in parallel on one-hardware component, as shown in Figure 1. It provides zero hardware fault tolerated (since no redundancy in hardware) and one-software fault tolerated (since 2 out of 3 software versions are required to be functional. The failure of 1 software version would be lead the system to failure).

The model unreliability ($P(t)$ or $1-R(t)$) is given by [6]:

$$\begin{aligned} &Prv_{12} + (Qrv_{12}\,Prv_{13}) + (Qrv_{12}\,Qrv_{13}\,Prv_{23}) \\ &+ (Qrv_{12}\,Qrv_{13}\,Qrv_{23}\,P_d) + (Qrv_{12}\,Qrv_{13}\,Qrv_{23}\,Q_d\,P_{all}) \\ &+ (Qrv_{12}\,Qrv_{13}\,Qrv_{23}\,Q_d\,Q_{all}\,P_h) \\ &+ [(Qrv_{12}\,Qrv_{13}\,Qrv_{23}\,Q_d\,Q_{all}\,Q_h) \times [(Pv_1\,Pv_2) \\ &+ (Qv_1\,Pv_2\,Pv_3) + (Qv_2\,Pv_1\,Pv_3)]] \end{aligned} \quad (1)$$

#### 3.1.2 NVP/1/1 Architecture

This model consists of three independent software components (or three different software versions), each running on an independent hardware component (could be the same type), as shown in Figure 1. The model provides one-hardware fault tolerated and one-software fault tolerated. The model unreliability ($P(t)$ or $1-R(t)$) is given by [6]:

$$\begin{aligned} &(Prv + Qrv\,Prv + Qrv^2\,Prv_{23}) + (Qrv^3\,P_d) \\ &+ (Qrv^3\,Q_d\,P_{all}) + (Pv_1\,Pv_2\,Qrv^3\,Q_d\,Q_{all} \\ &+ Pv_1\,Pv_3\,Qv_2\,Qrv^3\,Q_d\,Q_{all} \\ &+ Pv_2\,Pv_3\,Qv_1\,Qrv^3\,Q_d\,Q_{all}) \\ &+ Qv_2\,Qv_3\,Pv_1\,Qrv^3\,Q_d\,Q_{all}\,P_h^2\,Q_h \\ &+ Qrv^3\,Q_d\,Q_{all}\,P_h^2\,Q_h\,(1-Pv_3)\,(1-Pv_1\,Pv_2) \\ &+ Qv_1\,Qv_2\,Pv_3\,Qrv^3\,Q_d\,Q_{all}\,P_h^2\,Q_h \\ &+ Qrv^3\,Q_d\,Q_{all}\,P_h^2\,Q_h(1-Pv_2)\,(1-Pv_1\,Pv_3) \\ &+ Qv_1\,Qv_3\,Pv_2\,Qrv^3\,Q_d\,Q_{all}\,P_h^2\,Q_h \\ &+ Qrv^3\,Q_d\,Q_{all}\,P_h^2\,Q_h(1-Pv_1)\,(1-Pv_2\,Pv_3) \\ &+ 2(Pv_1\,Qv_2\,Qv_3\,Qrv^3\,Q_{all}\,Q_d\,P_h\,Q_h^2) \\ &+ 2(Qv_1\,Pv_2\,Qv_3\,Qrv^3\,Q_{all}\,Q_d\,P_h\,Q_h^2) \\ &+ 2(Qv_1\,Qv_2\,Pv_3\,Qrv^3\,Q_{all}\,Q_d\,P_h\,Q_h^2) \end{aligned} \quad (2)$$

where $Prv_{12} = Prv_{13} = Prv_{23} = Prv$, and $Ph_1 = Ph_2 = Ph_3 = P_h$

### 3.2 Recovery Block (RB) Architecture[7-8]

This model is different from the NVP models. The RB model uses an adjudication module called an acceptance test. It begins when the output of the first or primary module is tested for acceptability. If it is not acceptable, it rolls back to the state at the beginning of the process and the second module will execute and evaluate its output for the acceptance. If the outputs from all modules cannot pass the acceptance test, the system fails.

#### 3.2.1 RB/1/1 Architecture

This model consists of two hardware components (could be the same type) each running two independent software components (must be two different software versions called the primary and the secondary). The model provides one-hardware faults tolerated and one-software fault tolerated. The model unreliability ($P(t)$ or $1-R(t)$) is given by[6]:

$$\begin{aligned} &Prv + Qrv\,P_d + Qrv\,Q_d\,P_{all} + Qrv\,Q_d\,Q_{all}\,Ph_1\,Ph_2 \\ &+ Qrv\,Q_d\,Q_{all}\,(1-Ph_1\,Ph_2)\,Pv_1\,Pv_2 \end{aligned} \quad (3)$$

## 4. Genetic Algorithm with Penalty Function [9-11]

Genetic Algorithm (GA) is a stochastic optimization technique that uses the biological paradigm of evolution. GA was developed by John Holland [10]. It has a concept where good chromosome has a better potential of being carried to the next generation than the bad chromosome. It uses mathematical principle to indicate which chromosome is better or worse than the others are.

Firstly, to use GA we must encode the solution of problem into string called "chromosome" and each string has its unique characteristic inherited by "gene". Each chromosome is evaluated by a fitness function to indicate its potentiality toward the final solutions. The desirable fitness function value is depended on the problem (maximization/minimization).

After that we must generate an initial population (a set of chromosomes), and use the three main operators to find the best solution, as described next.

Step 1: Selection operator: The process of selecting potentially good chromosomes from the current population generation to the next generation.

Step 2: Crossover operator: The process of shuffling any two randomly selected chromosomes to generate the new offspring (like breeding).

Step 3: Mutation operator: The process that randomly selects one chromosome to change one or more genes into random value for generating the new offspring.

Step 4: Repeat step 1-3 until the goal is reached.

Traditional GA is not quite suitable for optimization problems with constraints. A major effective approach to deal with constraints is to apply a penalty to the chromosome when it falls outside a constrained search space (i.e., the solution that violates the constraint). The penalty is applied to reduce the fitness value of the chromosome.

There are many possible strategies for penalty function. The effective approach is used a distance metric of infeasible solution from the feasible region to calculate the penalty value.

In 1996, Coit and Smith proposed the adaptive penalty function for combinatorial reliability design [11]. This penalty function is dynamic. Their adaptive penalty function from [11] is:

$$V_{ip} = V_i - ((\Delta w_i/NFT_w)^k + (\Delta c_i/NFT_c)^k)(V_{all} - V_{feas}) \quad (4)$$

where $V_{ip}$ is the penalized objective function value of the solution $i$, $V_i$ is the unpenalized objective function value for solution $i$, $V_{all}$ denotes the unpenalized value of the best solution yet found, and $V_{feas}$ denotes the value of the best feasible solution yet found. The $k$ is a pre-set severity parameter. $NFT_c$ and $NFT_w$ are the "Near-Feasible Thresholds" for the cost and weight constraints respectively. And $\Delta w_i$ and $\Delta c_i$ represent the magnitude of any constraint weight and cost violations for the $i^{th}$ solution vector.

The dynamic $NFT$ is defined as follow [11],

$$NFT = NFT_0/(1+\lambda g) \quad (5)$$

where $NFT_0$ is an upper bound or starting point for $NFT$, $g$ is the generation number, and $\lambda$ is a constant, which assures that the entire region between NFT0, and zero is searched.

## 5. Methodology

In this research, we select the Genetic Algorithm with adaptive penalty function to optimize the reliability of embedded systems. We consider four different models from [6].

Model 1: Find all the optimal set of software and hardware without redundancy. Maximize the system reliability with a specified cost constraint.

Model 2: Find all the optimal set of software and hardware with or without NPV/0/1 redundancy. Maximize the system reliability with a specified cost constraint.

Model 3: Find all the optimal set of software and hardware with or without NPV/1/1 redundancy. Maximize the system reliability with a specified cost constraint.

Model 4: Find all the optimal set of software and hardware with or without RB/1/1 redundancy. Maximize the system reliability with a specified cost constraint.

We apply the GA to these four models by following these five steps:

### 5.1 Encoding

For example, an embedded (hardware and software) system with three subsystems connected in series can be represented with a string:

H1 S1 | H2 S2 | H3 S3

where $Hi$ is the selected hardware component for subsystem $i$, and $Si$ is the selected software component/version for subsystem $i$.

Suppose you have $n$ choices of hardware components and $m$ choices of software components/versions for each subsystem.

Model 1: $Hi$ can be 1..$n$ and $Si$ can be 1..$m$.

Model 2: $Hi$ can be 1..$n$ and Si can be 1.. $\left(m + \frac{m!}{3!(n-3)!}\right)$. If NVP/0/1 structure is selected for the subsystem, three different software versions are chosen.

Another example: If we have four versions of software.

1/2/3/4 = choose software version 1,2,3,4 respectively

5 = choose software version 2,3,4 (software version 1 is not chosen)

6 = choose software version 1,3,4 (software version 2 is not chosen)

7 = choose software version 1,2,4 (software version 3 is not chosen)

8 = choose software version 1,2,3 (software version 4 is not chosen)

Model 3: it is similar to model 2. *Hi* can be *1..n* and *Si* can be 1.. $\left(m + \frac{m_i!}{3!(m-3)!}\right)$. But if NVP/1/1 structure is selected for the subsystem, three different software versions and 3 identical hardware components are chosen.

Model 4: *Hi* can be *1..n* and *Si* can be 1.. $\left(m + \frac{m_i!}{2!(m-2)!}\right)$. If RB/1/1 structure is selected for the subsystem, two different software versions and two identical hardware components are chosen.

### 5.2 Initial Population

We set the initial population by randomly generating a set of chromosomes consisting of genes, and calculate their fitness value according to the fitness function.

### 5.3 Selection

The chromosomes or population are sorted by their fitness values. The higher 85% of the population with high fitness values are selected for the crossover process.

### 5.4 Crossover

We randomly select two systems or chromosomes from the current population for crossover, to produce two new chromosomes. Also we randomly select a subsystem for crossover. The positions P1 and P2 are labeled for crossover

Example: 1 2 | 1 3 | 1 1 | 3 4
1 1 | 2 3 | 3 5 | 4 4

Random subsystem = 3, P1 = 1, P2 = 2

Results: 1 2 | 1 3 | 3 5 | 3 4
1 1 | 2 3 | 1 1 | 2 4

We select the highest 15% of the population with the maximum fitness values from current population generation and compose with the best 85% from the crossover to be the next population generation.

### 5.5 Mutation

Firstly, the current population generation is sorted by fitness values. Then, each chromosome in the generation except the best 5 % is mutated with Mutation rate usually less than 10%. The chromosomes resulted from mutation are combined and considered as the chromosomes in the current population generation.

### 5.6 Objective Function

Our objective Function is defined as follows:

$$Max\ R = \prod_{i=1}^{n} R_i \qquad (6)$$

$$C = \sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij}C_{ij} + \sum_{i=1}^{n}\sum_{k=1}^{p_i} Y_{ik}C_{ik} \le Cost \qquad (7)$$

where $R_i = \sum_{j=1}^{m_i}\sum_{k=1}^{p_i} X_{ij}Y_{ik}Rhu_{ij}Rsu_{ik}$, if without redundancy

$$R_i = (1)\ , \text{if use NVP/0/1} \qquad (8)$$
$$R_i = (2)\ , \text{if use NVP/1/1} \qquad (9)$$
$$R_i = (3)\ , \text{if use RB/1/1} \qquad (10)$$

where $X_{ij} = 0,1$ $Y_{ij} = 0,1$ $i = 1,2,\ldots,n$ $j = 1,2,\ldots,m_i$ and $k = 1,2,\ldots,p_i$

If $C > Cost$, we use dynamic penalty function which is an effective approach to deal with constrains. It is applied when some chromosomes are infeasible. If $Cost \ge C$, penalty($x$) = 0, else the penalty function would be applied to the objective function according to equation (4), where $k$ =1 instead. We do not have $\Delta w_i$, therefore the equation (4) is changed to

$$V_{ip} = V_i - (\Delta c_i/NFT_c)\,(V_{all} - V_{feas}) \qquad (11)$$

We simplify the equation (5) to be as follows:

$$NFT_{g+1} = \lambda\, NFT_g \qquad (12)$$

where $NFT_g$ is $NFT$ for generation $g$, and $\lambda$ is a constant.

## 6. Example Systems and Computational Results

As an example, we select a series system with 6 subsystems, where $m_i = 3$ and $p_i = 4$ for $i = 1, 2, \ldots,6$, to test our four optimization models with the GA approach. We assume that $P_d = 0.002$, $Prv_i = 0.004$, and $P_{all} = 0.005$. We define the population numbers to be 350, mutation rate of 1 percent, $NFT_c = 1000$ and $\lambda = 0.8$.

We get the information of hardware and software components from [6]. The cost and reliability of each software and hardware component are shown in Table 1.

With our four models, we optimize solutions for 3 different cost constraints; 180, 320, 460 and without cost constraints. We also apply the problem with Lingo, commercial non-linear programming software. It identifies if its obtained solution is "Global Solution" or "Local Solution". At costs = 320, 460 and unlimited, Lingo gives global solutions or optimal solutions. Our results with GA and the result differences compared to Lingo are shown in Table 2.

From Table 2 we can see that the results from GA are very close to the results from Lingo. At cost constraint equals to 180, the differences between GA and Lingo are, however, undeniable. From our calculation, Lingo generates the wrong results, since the cost of the results equals to 200. The reason might be that Lingo couldn't reach to the global solution when the constraints are very tight.

Next step, we run ten GA simulation runs for each model and each cost constraint. We find that the results are not exactly the same in each run. The variations of the results, which are very small, are shown in Table 3.

| (i,j) | HW Cost | HW Reliability | (i,k) | SW Cost | SW Reliability |
|---|---|---|---|---|---|
| 1,1 | 30 | 0.995 | 1,1 | 30 | 0.950 |
| 1,2 | 10 | 0.980 | 1,2 | 10 | 0.908 |
| 1,3 | 10 | 0.980 | 1,3 | 20 | 0.908 |
| | | | 1,4 | 30 | 0.950 |
| 2,1 | 30 | 0.995 | 2,1 | 30 | 0.965 |
| 2,2 | 20 | 0.995 | 2,2 | 20 | 0.908 |
| 2,3 | 10 | 0.970 | 2,3 | 10 | 0.887 |
| | | | 2,4 | 20 | 0.908 |
| 3,1 | 20 | 0.994 | 3,1 | 20 | 0.978 |
| 3,2 | 30 | 0.995 | 3,2 | 30 | 0.954 |
| 3,3 | 100 | 0.992 | 3,3 | 20 | 0.860 |
| | | | 3,4 | 30 | 0.954 |
| 4,1 | 30 | 0.990 | 4,1 | 20 | 0.950 |
| 4,2 | 10 | 0.980 | 4,2 | 10 | 0.908 |
| 4,3 | 10 | 0.985 | 4,3 | 20 | 0.910 |
| | | | 4,4 | 20 | 0.950 |
| 5,1 | 30 | 0.995 | 5,1 | 30 | 0.905 |
| 5,2 | 20 | 0.980 | 5,2 | 20 | 0.967 |
| 5,3 | 30 | 0.995 | 5,3 | 10 | 0.967 |
| | | | 5,4 | 30 | 0.905 |
| 6,1 | 30 | 0.998 | 6,1 | 10 | 0.908 |
| 6,2 | 20 | 0.995 | 6,2 | 30 | 0.968 |
| 6,3 | 20 | 0.994 | 6,3 | 20 | 0.968 |
| | | | 6,4 | 20 | 0.955 |

Table 1: The cost and reliability of each software and hardware component [6]

| Model | Cost 180 | Cost 320 | Cost 460 | Unlimited |
|---|---|---|---|---|
| 1 | 0.635687 (1.4e-2) | 0.772099 (0) | 0.772099 (0) | 0.772099 (0) |
| 2 | 0.635687 (-3.2e-2) | 0.793362 (0) | 0.818126 (0) | 0.820888 (0) |
| 3 | 0.635687 (-3.2e-2) | 0.791116 (0) | 0.815219 (0) | 0.838132 (0) |
| 4 | 0.635687 (-3.2e-2) | 0.881536 (0) | 0.923596 (0) | 0.925244 (0) |

Table 2: Results of System Reliability from GA and the differences compared to Lingo

| Model | Cost 180 | Cost 320 | Cost 460 | Unlimited |
|---|---|---|---|---|
| 1 | 3.34e-5 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 6.40e-6 | 7.39e-7 | 0 |
| 4 | 0 | 1.82e-6 | 2.23e-8 | 1.47e-8 |

Table 3: Variant of the results from GA with Dynamic Penalty function

## 7. Conclusions

Comparing the GA optimization results with the LINGO results, we can conclude that our GA approach with the dynamic penalty function is able to find some optimal solutions for the embedded system design. Due to its heuristic characteristics, the GA has great potential to handle very large and complex combinatorial problems with tight constraint.

Each of our optimization models (1-4) is suitable for distinct sets or conditions. Models 2-4 provide system fault-tolerance with NVP/0/1, NVP/1/1 and RB/1/1, respectively. The models help us find the optimal system structures while considering basic information on reliability and cost of the available software and hardware components.

## References


[1] D.E. Fyffe, et al "System Reliability Allocation and a Computational Algorithm" *IEEE Trans. on Reliability*, Vol. R-17, 1968, pp 74-79.

[2] F.A. Tillman, et al "Optimization Techniques for System Reliability with Redundancy - A Review" *IEEE Trans. Reliability*, 1977, pp 148-155.

[3] W. Kuo, et al "Reliability Optimization with the Lagrange multiplier and branch-and-bound technique" *IEEE Trans. Reliability*, Vol. R-36, 1987 Dec, pp 1090-1095.

[4] H. Sarper "No Special Schemes are needed for Solving Software Reliability Optimization Models" *IEEE Trans. on Software Engineering*, Vol. 21, 1995, pp. 701-702.

[5] N. Wattanapongsakorn, S. Levitan, "Reliability Optimization Models for Fault-Tolerant Distributed Systems" *Reliability and Maintainability Symposium, 2001. Proc. Annual*, 2001, pp 193 –199.

[6] J.-C. Laprie, et al "Definition and Analysis of Hardware- and Software-Fault Tolerant Architectures" *IEEE Computer*, 1990, pp 39-51.

[7] M. R. Lyu, et al "Handbook of Software Reliability Engineering" *IEEE Computer Society Press*, McGraw-Hill, 1996.

[8] M. Foscoslos, S. Nilchan, "A Guide to Genetic Algorithms in Investment Management", *Quantitative Research Seminar*, 1998.

[9] J.H. Holland, "Genetic algorithm and the optimal allocation of trials", *SIAM J. Computing*, Vol. 2, 1973, pp 88-105.

[10] D.W. Coit et al "Penalty Guided Genetic Search for Reliability Design Optimization" *Computer & Industrial Engineering*, Vol. 30, 1996, pp 895-904.



Wiroj Phengprajit received B.Eng. from Mahidol University in 1999. Currently, he is working for his master degree in Computer Engineering Department at King Mongkut's University of Technology Thonburi.



Naruemon Wattanapongsakorn is a faculty member in Computer Engineering at the King Mongkut's University of Technology Thonburi. She received the B.S. degree and the M.S. degree, both from The George Washington University, and Ph.D. degree in Electrical Engineering from the University of Pittsburgh.