

Fault-tolerant embedded system design and optimization considering reliability estimation uncertainty

Naruemon Wattanapongskorn^{a,*}, David W. Coit^b

^a*Department of Computer Engineering, King Mongkut's University of Technology Thonburi, 91 Suksawad 48, Ratburana, Tung-Kru, Bangkok 10140, Thailand*

^b*Department of Industrial and Systems Engineering, Rutgers University, 96 Frelinghuysen Rd., Piscataway, NJ 08854, USA*

Received 3 March 2005; received in revised form 21 November 2005; accepted 19 December 2005

Available online 20 February 2006

Abstract

In this paper, we model embedded system design and optimization, considering component redundancy and uncertainty in the component reliability estimates. The systems being studied consist of software embedded in associated hardware components. Very often, component reliability values are not known exactly. Therefore, for reliability analysis studies and system optimization, it is meaningful to consider component reliability estimates as random variables with associated estimation uncertainty. In this new research, the system design process is formulated as a multiple-objective optimization problem to maximize an estimate of system reliability, and also, to minimize the variance of the reliability estimate. The two objectives are combined by penalizing the variance for prospective solutions. The two most common fault-tolerant embedded system architectures, *N*-Version Programming and Recovery Block, are considered as strategies to improve system reliability by providing system redundancy. Four distinct models are presented to demonstrate the proposed optimization techniques with or without redundancy. For many design problems, multiple functionally equivalent software versions have failure correlation even if they have been independently developed. The failure correlation may result from faults in the software specification, faults from a voting algorithm, and/or related faults from any two software versions. Our approach considers this correlation in formulating practical optimization models. Genetic algorithms with a dynamic penalty function are applied in solving this optimization problem, and reasonable and interesting results are obtained and discussed.

© 2006 Elsevier Ltd. All rights reserved.

Keywords: Estimation uncertainty; Reliability analysis; Fault tolerance; Embedded system; Genetic algorithm

1. Introduction

Determination of a recommended system design architecture involves the selection of available software and hardware components with the goal of maximizing system reliability, given constraints on the system. In the determination of optimal system designs, component reliability is not known exactly but must be estimated with some uncertainty. If the selected components have very high-reliability estimation uncertainty, then this can result in a system design which also has very high, and perhaps unacceptable, system reliability estimation uncertainty.

This is undesirable because system designers and users seek an optimal design with high-predicted reliability, but also one with low-estimation uncertainty.

There is existing research [1–6] on system reliability optimization considering component reliability/failure rate uncertainty. Rekab [1] considers reliability estimation uncertainty of a series system assuming all components in the system function *s*-independently, and the system is not fault-tolerant. The assumption of independent component failures has been used in many published research. However, there is evidence showing that such assumption is not realistic [7,8]. Rubinstein et al. [2] consider a redundancy allocation problem with uncertain component reliability, by maximizing the expected value of system reliability using genetic algorithms (GAs). Their approach,

*Corresponding author. Tel.: +662 470 9089; fax: +662 872 5050.

E-mail address: naruemon@cpe.kmut.ac.th (N. Wattanapongskorn).

considering only the expected values of reliability, is not sufficient for many decision makers because it ignores undesirable high uncertainty or risk associated with reliability estimation. In practice, system designers and users desire a designed system with a high-reliability estimate, associated with low-estimated variability. Coit et al. [3–5] solve the problem by considering variance of system reliability estimates in addition to the expected system reliability value. Zafiroopoulos et al. [6] perform reliability and cost optimization of an electronic system but does not consider failure dependencies.

There has also been published research [9–15] considering failure dependencies in the system design and optimization. Unlike these results, the models presented in this paper consider both component reliability estimation uncertainty and redundancy with heterogeneous software components, hardware components, and failure dependencies in multiple software versions.

The new models combine the approach of considering both the system reliability estimate and its variance, with the embedded system optimization approach. This is an extension of work from Wattanapongsakorn and Levitan [16] where component reliability is known with certainty. This results in practical reliability optimization models for the design of fault-tolerant embedded systems.

An embedded system consists of both hardware and software components where software components are embedded in (and running on) hardware components. To make it fault-tolerant, redundancy techniques can be applied to obtain fault-tolerant architectures. In this paper, *N*-Version Programming (NVP) architectures and Recovery Block (RB) 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 the majority of system failures are related to software faults. Therefore, optimal design of software fault-tolerance is often more critical than hardware fault-tolerance optimization. The fault-tolerant 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 [9,13,17].

Similar to Wattanapongsakorn et al. [16], 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. The systems that we model are series-parallel fault-tolerant systems. The redundancy allocation problem for series-parallel systems is known to be difficult (i.e., NP-hard). Many researchers have proposed a variety of approaches to solve this problem using, for example, integer programming, dynamic programming, mixed integer and nonlinear programming. Recent optimization approaches [16,18–20] are based on heuristic search algorithms (or meta-heuristics) such as simulated anneal-

ing, (GAs), and 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.

Optimization models have been developed to select both software and hardware components and redundancy levels given a total system cost constraint. In the system, there are a specified number of subsystems in series. For each subsystem, there are several choices of heterogeneous hardware and software components/versions to be selected. The system is designed using components, each with estimated reliability, but with known cost. Additionally, the variance or standard deviation of the estimated system reliability is known or can be approximated using standard statistical methods.

GAs are used as the optimization approach. The term ‘genetic’ derives from the roughly analogous natural reproduction of new populations by crossover and mutation operators. 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, a description of the fault-tolerant system architectures are presented. 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 three models each do consider a specific fault-tolerant architecture type. Section 5 explains the GA and its parameter settings. In Section 6, the effectiveness of our optimization models is demonstrated using numerical examples. Lastly, in Section 7, the paper ends with a summary and conclusions.

1.1. Assumptions

1. Each software component, hardware component and the system has 2 states: functional or failed.
2. Reliability of each software or hardware component is unknown, but it can be estimated.
3. There is no repair for each component or the system.
4. Hardware redundancy is in active mode (i.e., hot spares).
5. Failures of individual hardware components are *s*-independent.

1.2. Notation

$X/i/j$ system architecture X (NVP or RB) 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
- w_i number of software versions available for subsystem i
- \hat{R} estimated reliability of the distributed system
- \hat{R}_i estimated reliability of subsystem i
- \hat{R}_{hwij} estimated reliability of hardware component j for subsystem i
- \hat{R}_{swik} estimated reliability of software component k for subsystem i
- C_{hwij} cost of using hardware component j for subsystem i
- C_{swik} cost of developing software version k for subsystem i
- Cost maximum allowable cost (constraint)
- P_{v_i} probability of failure of software version i ($Q_{v_i} = 1 - P_{v_i}$)
- P_{rvij} probability of failure from related fault between two software versions i and j ($Q_{rvij} = 1 - P_{rvij}$)
- P_{rv} if P_{rvij} are the same for i and j , then $P_{rvij} = P_{rv}$ ($Q_{rv} = 1 - P_{rv}$)
- P_{all} probability of failure from some related fault among all software versions, due to faults in specification ($Q_{all} = 1 - P_{all}$)
- P_d probability of failure of decider or voter ($Q_d = 1 - P_d$)
- P_{h_i} probability of failure of hardware component i ($Q_{h_i} = 1 - P_{h_i}$). When only one hardware type is used, $P_{h_i} = P_h$ and $Q_{h_i} = Q_h$ for all i

2. Fault-tolerant system architectures to maximize reliability

General fault-tolerant approaches considered in this research are NVP and RB. Specifically, NVP/0/1, NVP/1/1 and RB/1/1 were considered. Fig. 1 depicts these general fault-tolerant architectures.

2.1. N-version programming (NVP) architecture

N-version programming [9,13] consists of an adjudication module called a voter, and N independently developed software versions, which are functionally equivalent. N is usually an odd number. This NVP model is based on the same concepts as N -modular redundancy (NMR) [13], which is a hardware fault-tolerant architecture. 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.

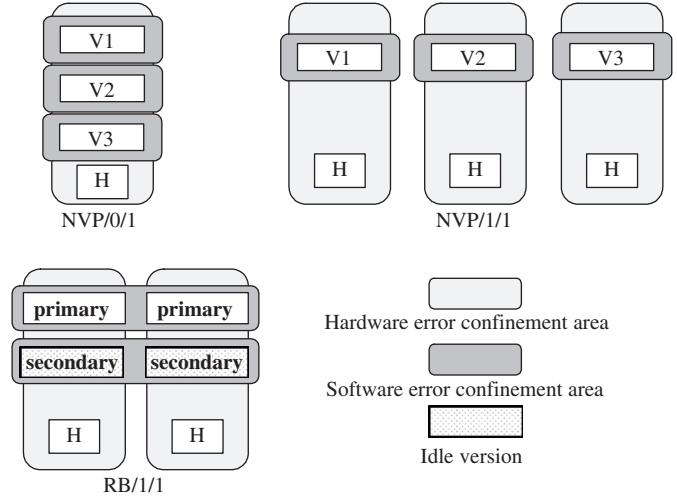


Fig. 1. Fault-tolerant architectures: NVP/0/1, NVP/1/1 and RB/1/1 [9,13].

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 is true: a single-hardware fault, at least two out of three software version faults, a related fault between software versions, faults from software specification, and faults from the voting algorithm.

For one particular subsystem, the NVP/0/1 model consists of three independent software versions running on a hardware component. All conceivable system states where the system has failed have been considered and enumerated. Wattanapongsakorn et al. [16] enumerated all possible failure conditions for a specified subsystem design to generate an equation for the probability that an unacceptable result occurs during a single-task iteration, $1 - R_i$ or P . This probability of failure for subsystem i is given by

$$\begin{aligned}
 P = & P_{rv12} + Q_{rv12} P_{rv13} + Q_{rv12} Q_{rv13} P_{rv23} \\
 & + Q_{rv12} Q_{rv13} Q_{rv23} P_d + Q_{rv12} Q_{rv13} Q_{rv23} Q_d P_{all} \\
 & + Q_{rv12} Q_{rv13} Q_{rv23} Q_d Q_{all} P_h \\
 & + Q_{rv12} Q_{rv13} Q_{rv23} Q_d Q_{all} Q_h P_{v1} P_{v2} \\
 & + Q_{rv12} Q_{rv13} Q_{rv23} Q_d Q_{all} Q_h Q_{v1} P_{v2} P_{v3} \\
 & + Q_{rv12} Q_{rv13} Q_{rv23} Q_d Q_{all} Q_h Q_{v2} P_{v1} P_{v3}.
 \end{aligned}$$

In practice, software versions may not have independent reliability, which is a common if not always accurate assumption in reliability analyses. They may not be independent because of correlated failures caused by system specification inadequacies and others. This is accommodated in this equation by the P_{rvij} and P_{all} terms.

This equation for unreliability can be written more compactly by redefining the reliability terms as reliability

Table 1
 a_i , k_{ij} and h_{ij} coefficients for NVP/0/1 unreliability

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

“components”. The following equation provides the unreliability for a specified subsystem with $P_{rvij} = P_{rv}$ for all i and j . a_i , k_{ij} and h_{ij} values are presented in Table 1.

$$P = \sum_{i=1}^s \left(a_i \prod_{j=1}^7 p_j^{k_{ij}} r_j^{h_{ij}} \right), \tag{1}$$

where,

- $p_1 = P_{rv12}, \quad r_1 = Q_{rv12},$
- $p_2 = P_{rv13}, \quad r_2 = Q_{rv13},$
- $p_3 = P_{rv23}, \quad r_3 = Q_{rv23},$
- $p_4 = P_d, \quad r_4 = Q_d,$
- $p_5 = P_{all}, \quad r_5 = Q_{all},$
- $p_6 = P_{v1}, \quad r_6 = Q_{v1},$
- $p_7 = P_{v2}, \quad r_7 = Q_{v2},$
- $p_8 = P_{v3}, \quad r_8 = Q_{v3},$
- $p_9 = P_h, \quad r_9 = Q_h,$

s = number of additive terms when all failure probabilities have been enumerated; $s = 9$ for

NVP/0/1 and 20 for NVP/1/1 architecture,

a_i = integer coefficient,

k_{ij} = power coefficient for j th component unreliability in the i th term,

h_{ij} = power coefficient for j th component reliability in the i th term,

p_j = unreliability of j th component,

r_j = reliability of j th component, $p_j + r_j = 1$ for all j .

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 enumerating 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

as shown in Fig 1. Any hardware failure can cause the associated 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 (i.e., does not host the software version) causes system failures.

Wattanapongsakorn et al. [16] enumerated all possible failure conditions for a specified subsystem design to generate an equation for the probability that an unacceptable result occurs during a single-task iteration, $1-R_i$ or P . The following equation predicts the probability of failure for subsystem i , and it pertains to the case where $P_{rvij} = P_{rv}$ for all i and j .

$$\begin{aligned}
 P = & P_{rv} + Q_{rv}P_{rv} + Q_{rv}^2P_{rv} + Q_{rv}^3P_d \\
 & + Q_{rv}^3Q_dP_{all} + P_{v1}P_{v2}Q_{rv}^3Q_dQ_{all} \\
 & + P_{v1}P_{v3}Q_{v2}Q_{rv}^3Q_dQ_{all} + P_{v2}P_{v3}Q_{v1}Q_{rv}^3Q_dQ_{all} \\
 & + P_{v1}P_{h1}P_{h2}Q_{v2}Q_{v3}Q_{rv}^3Q_dQ_{all}Q_{h3} \\
 & + Q_{rv}^3Q_dQ_{all}P_{h1}P_{h3}Q_{h2}Q_{v3}(1 - P_{v1}P_{v2}) \\
 & + P_{v3}P_{h1}P_{h3}Q_{v1}Q_{v2}Q_{rv}^3Q_dQ_{all}Q_{h2} \\
 & + Q_{rv}^3Q_dQ_{all}P_{h2}P_{h3}Q_{h1}Q_{v2}(1 - P_{v1}P_{v3}) \\
 & + P_{v2}P_{h2}P_{h3}Q_{v1}Q_{v3}Q_{rv}^3Q_{all}Q_dQ_{h1} \\
 & + Q_{rv}^3Q_dQ_{all}P_{h1}P_{h2}Q_{h3}Q_{v1}(1 - P_{v2}P_{v3}) \\
 & + P_{v1}P_{h3}Q_{v2}Q_{v3}Q_{rv}^3Q_{all}Q_dQ_{h1}O_{h2} \\
 & + P_{v1}P_{h2}Q_{v2}Q_{v3}Q_{rv}^3Q_{all}Q_dQ_{h1}Q_{h3} \\
 & + P_{v2}P_{h2}Q_{v1}Q_{v3}Q_{rv}^3Q_{all}Q_dQ_{h1}Q_{h2} \\
 & + P_{v2}P_{h1}Q_{v1}Q_{v3}Q_{rv}^3Q_{all}Q_dQ_{h2}Q_{h3} \\
 & + P_{v3}P_{h1}Q_{v1}Q_{v2}Q_{rv}^3Q_{all}Q_dQ_{h1}Q_{h3} \\
 & + P_{v3}P_{h2}Q_{v1}Q_{v2}Q_{rv}^3Q_{all}Q_dQ_{h2}Q_{h3}.
 \end{aligned}$$

An analogous, but more complicated, mathematical expression can be developed when the P_{rvij} terms are uniquely determined for each pair of software versions. In practice, it is common and practical [12,13] to use a common failure probability for each pair.

The unreliability of a specified subsystem ($1-R_i$ or P) can also be represented by Eq. (1) for the case where $P_{rvij} = P_{rv}$ for all i and j , and $P_{hj} = P_h$ for all i . a_i , k_{ij} and h_{ij} for NVP/1/1 architecture are listed in Table 2. The more compact equation is a result of expanding terms, and then, grouping together similar terms.

2.2. Recovery block (RB): RB/1/1 architecture

The RB model [9,13], as depicted in Fig. 1, consists of an adjudication module called an *acceptance test*, and at least two software components, called alternates. 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

Table 2
 a_i, k_{ij} and h_{ij} coefficients for NVP/1/1 unreliability

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

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. Failure of the system occurs when both versions fail, or both hardware components fail. Wattanapongsakorn et al. [16] enumerated all possible failure conditions for a specified subsystem design (subsystem i) to generate the following equation for the probability that an unacceptable result occurs during a single task iteration, $1-R_i$ or P .

$$P = P_{rv_{12}} + Q_{rv_{12}}P_d + Q_{rv_{12}}Q_dP_{all} + Q_{rv_{12}}Q_dQ_{all}P_{h_1}P_{h_2} + Q_{rv_{12}}Q_dQ_{all}(1 - P_{h_1}P_{h_2})P_{v_1}P_{v_2}.$$

Like the other models, the unreliability can also be represented by Eq. (1) for the case where $P_{rv_{ij}} = P_{rv}$ for all i and j , and $P_{h_i} = P_h$ for all i . a_i, k_{ij} and h_{ij} for RB/1/1 architecture are listed in Table 3.

With these fault-tolerant architectures, we develop four optimization models for fault-tolerant embedded systems considering reliability estimates with uncertainty. Each model has multiple objectives: (1) to maximize the system reliability estimate and (2) to minimize the variance of the reliability estimate. The two system-level objectives are combined by penalizing the system reliability estimate variance. The components, which are available for the system design, each has reliability uncertainty expressed by a reliability estimate variance. The variance can be estimated using standard statistical methods derived from

Table 3
 a_i, k_{ij} and h_{ij} coefficients for RB/1/1 unreliability

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

the binomial distribution or non-parametric statistics, such as Kaplan–Meier.

In the next section, we formulate equations for system reliability estimate and variance for the reliability estimates for the three system fault-tolerant architectures. These equations are used in the four optimization models in Section 4.

3. Reliability estimation variability

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

$$\hat{P} = \sum_{i=1}^s \left(a_i \prod_{j=1}^7 \hat{p}_j^{k_{ij}} \hat{r}_j^{h_{ij}} \right) = \sum_{i=1}^s \left(a_i \prod_{j=1}^7 (1 - \hat{r}_j)^{k_{ij}} \hat{r}_j^{h_{ij}} \right). \tag{2}$$

Direct calculation of $E[\hat{P}]$ and $\text{Var}(\hat{P})$ is difficult using Eq. (2). However, the equation can be re-arranged to accommodate available methods from Jin and Coit [3]. Therefore, Eq. (2) has been rearranged by expanding the $(1 - \hat{r}_j)^{k_{ij}}$ terms, and then, combining similar terms. In practice, this is a very tedious process. This expansion procedure is conducted automatically using Matlab code based on the parameters in Tables 1, 2 or 3.

$$\hat{P} = \sum_{i=1}^s \left(a_i \prod_{j=1}^7 (1 - \hat{r}_j)^{k_{ij}} \hat{r}_j^{h_{ij}} \right) = \sum_{i=1}^t \left(b_i \prod_{j=1}^7 \hat{r}_j^{n_{ij}} \right), \tag{3}$$

where b_i is the integer coefficient and t is the number of terms after expansion, $t > s$.

b_i and n_{ij} are determined by grouping similar terms. Tables 4–6 list all the expansion results.

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. (3) can be used to obtain the mean and the variance of unreliability, \hat{P} , as demonstrated by Jin and Coit [3]. The mean and the

Table 4
 n_{ij} and b_i coefficients for NVP/0/1 unreliability

i	n_{ij}							b_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

variance of unreliability, \hat{P} , can be obtained as follows [3]:

$$E[\hat{P}] = \sum_{i=1}^I \left(b_i \prod_{j=1}^7 E[\hat{r}_j^{n_{ij}}] \right), \tag{4}$$

$$\begin{aligned} \text{Var}(\hat{P}) = & \sum_{i=1}^I \left\{ b_i^2 \left(\prod_{j=C_i}^7 E[\hat{r}_j^{2n_{ij}}] - \prod_{j=1}^7 \left(E[\hat{r}_j^{n_{ij}}] \right)^2 \right) \right\} \\ & + 2 \sum_{i < m}^I \left\{ b_i b_m \left(\prod_{j=1}^7 E[\hat{r}_j^{n_{ij}+n_{mj}}] - \prod_{j=1}^7 E[\hat{r}_j^{n_{ij}}] E[\hat{r}_j^{n_{mj}}] \right) \right\}. \end{aligned} \tag{5}$$

Table 7 lists component reliability estimates or unreliability estimate values. These data are selected directly from Wattanapongsakorn et al. [16]. Eq. (4) and Eq. (5) can then be used providing that higher moments of component reliability estimates are known or can be estimated. Without loss of generality, it is assumed that reliability estimation is based on tests which can be represented as independent Bernoulli trials. Then, applied binomial distribution theory was used to estimate the higher moments as demonstrated by Jin and Coit [3].

In practice, the variance estimates are an input to the model and they can be obtained from any credible source or statistical model. For our exploratory studies, component variance was modeled by a variance-factor vector, $\boldsymbol{\eta} = [\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6, \eta_7]$. A higher value of η_i corresponds to lower variance, according to $\text{Var}(\hat{r}_i) =$

Table 5
 n_{ij} and b_i coefficients for NVP/1/1 unreliability

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

$\hat{r}_i(1 - \hat{r}_i)/\eta_i$. When the data originates from a binomial distribution, then η_i is just the sample size. When the data is non-binomial, the η_i values are analogous to the sample size. For that case, η_i is a scaling factor that is used to consider different uncertainty levels. For the examples considered, design configurations can be determined for different uncertainty levels by varying $\boldsymbol{\eta}$. In Table 7, when $\boldsymbol{\eta} = [4, 2, 2, 4, 3, 6, 6]$, the corresponding component reliability estimate variances are presented in Table 7 as one example.

To demonstrate the system variance estimation, we provide several numerical examples. Table 8 lists six results with respect to different component variance-factor vectors. It can be observed that as component variances become small, the overall variance of the system unreliability estimate \hat{P} also decreases.

4. Optimization models to maximize reliability considering uncertainty

In this section, four-optimization models are presented for reliability of distributed systems. A distributed system, shown in Fig. 2, consists of subsystems where software components/versions are mapped (or distributed) 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.

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, subject 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.

In practice, Model 1 can be considered as a special-case or subset of the following Models 2 and 3. However, it is presented separately to allow for more direct examination of the specific benefits of fault tolerant architectures.

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

$$\max \hat{R}(\mathbf{x}, \mathbf{y}) - \lambda \text{Var}(\hat{R}(\mathbf{x}, \mathbf{y}))$$

S. t.

$$\sum_{i=1}^n \sum_{j=1}^{m_i} C_{hw_{ij}} x_{ij} + \sum_{i=1}^n \sum_{k=1}^{w_i} C_{sw_{ik}} y_{ik} \leq \text{Cost},$$

$$\sum_{j=1}^{m_i} x_{ij} = 1,$$

$$\sum_{k=1}^{w_i} y_{ik} = 1,$$

$$x_{ij} \in \{0, 1\}, \quad y_{ik} \in \{0, 1\},$$

Table 6
 n_{ij} and b_i coefficients for RB/1/1 unreliability

i	n_{ij}							b_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

Table 8
System unreliability metrics

η	P	$E[\hat{P}]$	$\text{Var}(\hat{P})$
[1,1,1,1,1,1]	0.05571	0.03716	0.03578
[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]	0.05571	0.06068	0.00777
[12,12,12,12,12,12]	0.05571	0.05925	0.00521

Table 7
Parameters and definition of component's variance of reliability estimate when $\eta = [6,4,2,2,4,3,6]$

Unreliability estimate	Reliability estimate	Variance of component reliability estimate
$P_{rv} = 0.004$	$Q_{rv} = 0.996$	$\text{Var}(P_{rv}) = \text{Var}(Q_{rv}) = (P_{rv} \times Q_{rv})/\eta_1 = 0.000664$
$P_d = 0.02$	$Q_d = 0.98$	$\text{Var}(P_d) = \text{Var}(Q_d) = (P_d \times Q_d)/\eta_2 = 0.0049$
$P_{all} = 0.005$	$Q_{all} = 0.995$	$\text{Var}(P_{all}) = \text{Var}(Q_{all}) = (P_{all} \times Q_{all})/\eta_3 = 0.0024875$
$P_{v_1} = 0.035$	$Q_{v_1} = 0.965$	$\text{Var}(P_{v_1}) = \text{Var}(Q_{v_1}) = (P_{v_1} \times Q_{v_1})/\eta_4 = 0.016$
$P_{v_2} = 0.046$	$Q_{v_2} = 0.954$	$\text{Var}(P_{v_2}) = \text{Var}(Q_{v_2}) = (P_{v_2} \times Q_{v_2})/\eta_5 = 0.012$
$P_{v_3} = 0.09$	$Q_{v_3} = 0.910$	$\text{Var}(P_{v_3}) = \text{Var}(Q_{v_3}) = (P_{v_3} \times Q_{v_3})/\eta_6 = 0.03$
$P_h = 0.03$	$Q_h = 0.970$	$\text{Var}(P_h) = \text{Var}(Q_h) = (P_h \times Q_h)/\eta_7 = 0.004$

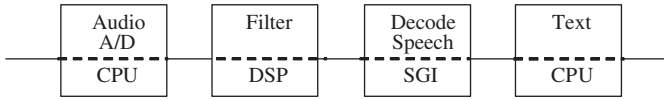


Fig. 2. Distributed system example.

where

$$\hat{R}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n (\hat{R}_i(\mathbf{x}, \mathbf{y})) = \prod_{i=1}^n \left(\sum_{j=1}^{m_i} \sum_{k=1}^{w_i} \hat{R}_{hw_{ij}} \hat{R}_{sw_{ik}} x_{ij} y_{ik} \right), \tag{6}$$

$$\text{Var}(\hat{R}(\mathbf{x}, \mathbf{y})) = \prod_{i=1}^n (\text{Var}(\hat{R}_i(\mathbf{x}, \mathbf{y})) + \hat{R}_i(\mathbf{x}, \mathbf{y})^2) - \prod_{i=1}^n (\hat{R}_i(\mathbf{x}, \mathbf{y}))^2, \tag{7}$$

$$\text{Var}(\hat{R}_i(\mathbf{x}, \mathbf{y})) = \left(\text{Var}(\hat{R}_{hw_i}) + \hat{R}_{hw_i}^2 \right) \left(\text{Var}(\hat{R}_{sw_i}) + \hat{R}_{sw_i}^2 \right) - (\hat{R}_{hw_i} \hat{R}_{sw_i})^2, \tag{8}$$

$$\hat{R}_{hw_i} = \sum_{j=1}^{m_i} \hat{R}_{hw_{ij}} x_{ij}, \quad \hat{R}_{sw_i} = \sum_{k=1}^{w_i} \hat{R}_{sw_{ik}} y_{ik},$$

$$\text{Var}(\hat{R}_{hw_i}) = \sum_{j=1}^{m_i} \text{Var}(\hat{R}_{hw_{ij}}) x_{ij},$$

$$\text{Var}(\hat{R}_{sw_i}) = \sum_{k=1}^{w_i} \text{Var}(\hat{R}_{sw_{ik}}) y_{ik},$$

$$\mathbf{x} = (x_{11}, x_{12}, \dots, x_{21}, x_{22}, \dots),$$

$$\mathbf{y} = (y_{11}, y_{12}, \dots, y_{21}, y_{22}, \dots),$$

$$x_{ij} = \begin{cases} 1 & \text{if hardware component } j \text{ is selected for module } i, \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{if software component } k \text{ is selected for module } i, \\ 0 & \text{otherwise,} \end{cases}$$

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. λ 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.

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 and may require redundancy. The problem formulation for this model is the same as in the previous model, except that each subsystem may selectively use 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 estimates and variances of those reliability estimates. 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.

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

$$\max \hat{R}(\mathbf{x}, \mathbf{y}) - \lambda \text{Var}(\hat{R}(\mathbf{x}, \mathbf{y}))$$

S. t.

$$\sum_{i=1}^n \sum_{j=1}^{m_i} C_{hw_{ij}} x_{ij} + \sum_{i=1}^n \sum_{k=1}^{w_i} C_{sw_{ik}} y_{ik} \leq \text{Cost},$$

$$\sum_{j=1}^{m_i} x_{ij} = 1,$$

$$\sum_{k=1}^{w_i} y_{ik} - 2z_i = 1,$$

$$x_{ij} \in \{0, 1\}, \quad y_{ik} \in \{0, 1\}, \quad z_i \in \{0, 1\},$$

where

$$\hat{R}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n (\hat{R}_i(\mathbf{x}, \mathbf{y})),$$

$\text{Var}(\hat{R}(\mathbf{x}, \mathbf{y}))$ is from Eq.(7),

$$\hat{R}_i(\mathbf{x}, \mathbf{y}) = \begin{cases} \sum_{j=1}^{m_i} \sum_{k=1}^{w_i} \hat{R}_{hw_{ij}} \hat{R}_{sw_{ik}} x_{ij} y_{ik} & \text{if } z_i = 0, \\ 1 - \hat{P} (\hat{P} \text{ is from Eq. (4) and Table 4}) & \text{if } z_i = 1, \end{cases}$$

$$\text{Var}(\hat{R}_i(\mathbf{x}, \mathbf{y})) = \begin{cases} \text{from Eq. (8)} & \text{if } z_i = 0, \\ \text{from Eq. (5) and Table 4} & \text{if } z_i = 1, \end{cases}$$

x_{ij} and y_{ik} are defined as in Model 1. z_i is a 0-1 decision variable. If z_i is equal to 0, no redundancy is used, and Model 2 is the same as Model 1. If z_i is equal to 1, three software components are selected as part of the NVP/0/1 architecture.

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.

$$\max \hat{R}(\mathbf{x}, \mathbf{y}) - \lambda \text{Var}(\hat{R}(\mathbf{x}, \mathbf{y}))$$

S. t.

$$\sum_{i=1}^n \sum_{j=1}^{m_i} C_{hw_{ij}} x_{ij} + \sum_{i=1}^n \sum_{k=1}^{w_i} C_{sw_{ik}} y_{ik} \leq \text{Cost},$$

$$\sum_{j=1}^{m_i} x_{ij} - 2z_i = 1,$$

$$\sum_{k=1}^{w_i} y_{ik} - 2z_i = 1,$$

$$x_{ij} x_{il} = 0 \quad \forall i, j, l \quad (j \neq l),$$

$$x_{ij} \in \{0, 1, 2, \dots\}, \quad y_{ik} \in \{0, 1\}, \quad z_i \in \{0, 1\},$$

where

$$\hat{R}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n (\hat{R}_i(\mathbf{x}, \mathbf{y})),$$

Var($\hat{R}(\mathbf{x}, \mathbf{y})$) is from Eq.(7),

$$\hat{R}_i(\mathbf{x}, \mathbf{y}) = \begin{cases} \sum_{j=1}^{m_i} \sum_{k=1}^{w_i} \hat{R}_{hw_{ij}} \hat{R}_{sw_{ik}} x_{ij} y_{ik} & \text{if } z_i = 0, \\ 1 - \hat{P} \quad (\hat{P} \text{ is from Eq. (4) and Table (5)}) & \text{if } z_i = 1, \end{cases}$$

$$\text{Var}(\hat{R}_i(\mathbf{x}, \mathbf{y})) = \begin{cases} \text{from Eq. (8)} & \text{if } z_i = 0, \\ \text{from Eq. (5) and Table (5)} & \text{if } z_i = 1, \end{cases}$$

z_i and y_{ik} are defined as before. For this model, x_{ij} is defined as the number of hardware components selected. If redundancy is to be used, there will be multiple software components each installed on a distinct hardware component. The software components are required to be different, but the hardware components will be of the same type, but that type is selected from m_i available options. The additional constraint ($x_{ij} x_{il} = 0$) assures that only one hardware type will be selected.

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 for the Recovery Block architecture.

$$\max \hat{R}(\mathbf{x}, \mathbf{y}) - \lambda \text{Var}(\hat{R}(\mathbf{x}, \mathbf{y}))$$

S. t.

$$\sum_{i=1}^n \sum_{j=1}^{m_i} C_{hw_{ij}} x_{ij} + \sum_{i=1}^n \sum_{k=1}^{w_i} C_{sw_{ik}} y_{ik} \leq \text{Cost},$$

$$\sum_{j=1}^{m_i} x_{ij} - z_i = 1,$$

$$\sum_{k=1}^{w_i} y_{ik} - z_i = 1,$$

$$x_{ij} x_{il} = 0 \quad \forall i, j, l \quad (j \neq l),$$

$$x_{ij} \in \{0, 1, 2, \dots\}, \quad y_{ik} \in \{0, 1\}, \quad z_i \in \{0, 1\},$$

where

$$\hat{R}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^n (\hat{R}_i(\mathbf{x}, \mathbf{y})),$$

Var($\hat{R}(\mathbf{x}, \mathbf{y})$) is from Eq.(7),

$$\hat{R}_i(\mathbf{x}, \mathbf{y}) = \begin{cases} \sum_{j=1}^{m_i} \sum_{k=1}^{w_i} \hat{R}_{hw_{ij}} \hat{R}_{sw_{ik}} x_{ij} y_{ik} & \text{if } z_i = 0, \\ 1 - \hat{P} \quad (\hat{P} \text{ is from Eq. (4) and Table (6)}) & \text{if } z_i = 1, \end{cases}$$

$$\text{Var}(\hat{R}_i(\mathbf{x}, \mathbf{y})) = \begin{cases} \text{from Eq. (8)} & \text{if } z_i = 0, \\ \text{from Eq. (5) and Table (6)} & \text{if } z_i = 1. \end{cases}$$

5. Genetic algorithm solution methodology

GA was used to determine recommended solutions to the problem. GA is a heuristic search algorithm. As a heuristic, it can not guarantee optimality, but it has been demonstrated to yield optimal and near-optimal solutions for many diverse problem domains.

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

$$H_1 S_1 | H_2 S_2 | H_3 S_3 | \dots | H_n S_n,$$

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

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

Model 1: H_i can be 1, 2, ..., m and S_i can be 1, 2, ..., w .

Model 2: H_i can be 1, 2, ..., m and S_i can be 1, 2, ...,

$$1, 2, \dots, \left(w + \frac{w!}{3!(w-3)!} \right).$$

If NVP/0/1 redundancy is selected for the subsystem, three different software versions and one hardware component are chosen. Different combinations of software choices are considered as different design alternatives as in the following example. Consider the following example with four different software versions available for a subsystem.

Let 1/2/3/4 = choose software version 1,2,3,4 individually

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 is the same. H_i can be 1, 2, ..., m and S_i can be

$$1, 2, \dots, \left(w + \frac{w!}{3!(w-3)!} \right).$$

However, with NVP/1/1 redundancy selected for the subsystem, three different software versions and one hardware type (three identical hardware components) must be chosen.

For Model 4, H_i can be 1, 2, ..., m and S_i can be

$$1, 2, \dots, \left(w + \frac{w!}{2!(w-2)!} \right).$$

If RB/1/1 redundancy is selected for the subsystem, two different software versions and one hardware type (two identical hardware components) must be chosen.

5.2. Initial population

The initial population was selected by randomly generating a set of chromosomes consisting of genes. Their fitness value was computed 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. Then, one subsystem is selected, and the two solutions exchange subsystem design information. The crossover positions P1 and P2 are labeled with bold font in the following example.

Example: $(P1)$ 1 2 | 1 3 | 1 1 | 3 4,
 $(P2)$ 1 1 | 2 3 | 3 5 | 4 4.
 Random subsystem = 3.
 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 and combine them with the best 85% from the crossover

operator results to form the population for the next generation.

5.5. Mutation

The mutation operator is defined as follows. First, the current population is sorted by fitness values. Then, each chromosome in the generation except the best 5% is mutated with a selected mutation rate which is usually selected to be less than 10%. The chromosomes resulting from mutation are combined with the current population generation.

5.6. Penalty function

A dynamic penalty function [21] was used to enforce feasibility, i.e., system cost constraint is satisfied. It is an effective approach to deal with constrained problems. 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. Otherwise, the cost penalty would be applied to the objective function. By doing this, the selected infeasible solution search space is explored and considered as local or temporary solutions which may lead to finding good feasible solutions. As the search progresses, the constraint violation penalty increases so that, by the end of the search, only feasible solutions are left. Details of the penalty function are given in Coit et al. [21].

6. Numerical example

The problem originally solved by Wattanapongsakorn et al. [16] was used to provide an example problem considering the reliability estimate and variance of the reliability estimate as multiple objectives. This example reliability optimization problem is a series system with six subsystems having $n = 6$, $m_i = 3$, and $w_i = 4$. 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 9 shows the reliability estimate and its variance as well as the cost of all the available components.

We apply this input data to optimize the four-optimization models with various system design cost constraints of 180, 320, 460, and also, with an unlimited cost constraint. The variance penalty value (λ), was set to 1.0 for all cost constraints, except when the system design cost constraint was 460 where λ was varied ($\lambda = 0.1, 1, 2, 3, 4, 5$ and 10). Other design considerations are failure correlations/dependencies which are $P_{rv_{ij}} = 0.004$ for all i and j , $P_{all} = 0.005$ and $P_d = 0.002$. Their corresponding variance-factors (η_i) are all equal to 20. In a few seconds of simulation time, we obtain the GA results from our four optimization models as presented in Tables 10–13.

From the GA results presented in Table 10 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 cost constraints. With a very tight cost constraint (Cost = 180), the best possible solutions are not as good as when the cost constraint is relaxed to 320 or 460. Models 2–4 cannot find solutions with component redundancy and still satisfy the constraint; thus, resulting in the same solution as obtained by Model 1. The selected component allocations are depicted in Table 11. A “better solution” means the solution has higher reliability estimate and lower variance of the reliability estimate.

Table 9
Available components and their parameters

Hardware				Software			
(i, j)	C_{hwij}	\hat{R}_{ij}	Variance-factor η_{ik}	(i, k)	C_{swij}	\hat{R}_{ik}	Variance-factor η_{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	0.992	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

Table 10
Results from GA with $\lambda = 1.0$

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

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 finds better solutions than those offered by Model 1, resulting in higher system reliability estimates and lower variances. For certain cost constraints, 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 12. Each solution cannot have a system cost that exceeds the constraint. Compared to all the models with no cost constraint, Model 3 with NVP/1/1 redundancy offers its solution (with cost of 810) which is much more expensive because more resources are required.

To see the effect and relationship of system reliability estimate and its variance, the variance penalty, λ , was changed to vary the relative importance of minimizing variance, i.e., uncertainty. The objective remains to find the best solution with the highest reliability estimate and the lowest variance of the reliability estimate, but the relative importance of the variance is changing.

The value of λ was varied, i.e., $\lambda = 0.1, 1, 2, 3, 4, 5$ and 10. Greater variance penalty indicates a higher effect to the obtained solutions, since a larger value is subtracted from the objective function to be maximized. Table 13 presents the GA results obtained for all the models with cost constraint 460 at each λ value. As expected, 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 a higher variance penalty, each model tends to seek solutions with smaller variances of the reliability estimates, but at the same time, unavoidably causing the system reliability estimates to be lower. This dependent relationship of reliability estimate and its variance is very interesting and meaningful. Different solutions or system designs are preferred when the decision-maker is more risk-averse, i.e., desires to avoid uncertainty. For some system design problems, where uncertainty can not be tolerated, we have to compromise by accepting lower reliability estimates.

Table 11
Component allocations for the results shown in Table 10

	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 12
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

Table 13
Results from GA with various variance penalty at Cost = 460

λ	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

form of reliability estimates and variance of the reliability estimate. The system design objectives are to maximize the system reliability estimate, and at the same time, minimize the associated variance. These multiple objectives are in

contrast to one another. When one objective is more important than another one, generally a distinct design choice is suggested. Therefore, the system design decision depends on the degree of importance of minimizing estimation uncertainty.

This is the first time that a technique is proposed to optimize reliability of system considering reliability estimation uncertainty using multiple software versions with different reliabilities and correlated failures. The correlated failures that we consider are from related faults between two software versions (P_{rvij}), related fault among all software versions (P_{all}), and decider or voter failure (P_d). The reliability optimization of redundant systems consists of realistic assumptions of failure correlation between/among software versions.

Four practical optimization models are provided for embedded system design considering no redundancy (Model 1) and with redundancy using NVP architectures (Models 2 and 3) and RB architectures (Model 4). A GA with dynamic penalty function was successfully applied to solve our case-study optimization problem, providing satisfying results.

Acknowledgment

This research was supported in part by the Thailand Research Fund, Thailand under Grant PDF 70-2544.

References

- [1] Rekab K. A sampling scheme for estimating the reliability of a series system. *IEEE Trans Reliab* 1993;R-42(2):217–30.
- [2] Rubinstein R, Levitin G, Lisnianski A, Ben-Haim H. Redundancy optimization of static series parallel reliability models under uncertainty. *IEEE Trans Reliab* 1997;R-46(4):503–11.
- [3] Jin T, Coit DW. Variance of system reliability estimates with arbitrarily repeated components. *IEEE Trans Reliab* 2001;R-50(4):409–13.
- [4] Coit DW, Jin T. Multi-criteria optimization: maximization of a system reliability estimate and minimization of the estimated variance. In *Proceedings of the European safety and reliability international conference (ESREL)*, 2001.
- [5] Coit DW, Jin T, Wattanapongsakorn N. System optimization with component reliability estimation uncertainty: a multi-criteria approach. *IEEE Trans Reliab* 2004;R-53(3):369–80.
- [6] Zafiroopoulos EP, Dialynas EN. Reliability and cost optimization of electronic devices considering the component failure rate uncertainty. *Reliab Eng Syst Saf* 2004;84:271–84.
- [7] Eckhardt DE, Lee LD. A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Trans Software Eng* 1985;11:1511–7.
- [8] Littlewood B. Comments on ‘reliability and performance analysis for fault-tolerant programs consisting of versions with different characteristics’ by Gregory Levitin [RESS 86 (2004)75–82]. *Reliab Eng Syst Saf* 2006;91:119–20.
- [9] Laprie J-C, Arlat J, Beounes C, Kanoun K. Definition and analysis of hardware- and software-fault-tolerant architectures. *IEEE Comput* 1990;39–51.
- [10] Lyu MR, He Y-T. Improving the N -version programming process though the evolution of a design paradigm. *IEEE Trans Reliab* 1993; R-42(2):179–89.
- [11] Ashrafi N, Berman O, Cutler M. Optimal design of large software systems using N -version programming. *IEEE Trans Reliab* 1994;344–50.
- [12] Dugan JB, Lyu MR. System reliability analysis of an N -version programming application. *IEEE Trans Reliab* 1994;R-43(4):509–13.
- [13] Lyu MR (Editor-in-Chief). *Handbook of software reliability engineering*. McGraw-Hill: IEEE Computer Society Press; 1996.
- [14] Gutjahr WJ, Uchida G. A branch-and-bound approach to the optimization of redundant software under failure correlation. *Comput Oper Res* 2002;29:1773–91.
- [15] Tsujimura Y, Yamamoto H, Takeno T, Yamazaki G. Evolutionary design of N -version fault tolerant software system. In: *Proceedings of the 33rd international conference on computers and industrial Engineering Jeju, Korea, 2004*.
- [16] Wattanapongsakorn N, Levitan SP. Reliability optimization models for embedded systems with multiple applications. *IEEE Trans Reliab* 2004;R-53(3):406–16.
- [17] Pullum LL. *Software fault tolerance techniques and implementation*. Norwood, MA: Artech House; 2001.
- [18] Kulturel-Konak S, Norman B, Coit D, Smith A. Exploiting tabu search memory in constrained problems. *INFORMS J Comput* 2004; 16(3) (Summer).
- [19] Tsujimura Y, Yamamoto H, Takeno T, Yamazaki G. A genetic algorithm for designing N -version program. In: *Proceedings of the Fifth Asia-Pacific industrial engineering and management systems, Gold Coast, Australia, 2004*.
- [20] Levitin G. Optimal version sequencing in fault-tolerant programs. *Asia-Pacific J Oper Res* 2005;22(1):1–18.
- [21] Coit DW, Smith A, Tate D. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS J Comput* 1996;8(2):173–82.