Reliability Optimization of Redundant Software with Correlated Failures

Walter J. Gutjahr *

Abstract: Some authors, as Belli, Jedrzejowicz, Berman, Ashrafi and Sarper, have addressed the problem of reliability optimization of modular software with redundancy. At present, these approaches seem to be limited by the fact that they use statistical independence assumptions for different program versions. In view of strong empirical evidence of positive failure correlations between program versions, it is desirable to generalize these reliability optimization techniques by taking such correlations explicitly into consideration. In the presented paper, this is done based on a multiversion software failure correlation model developed by Eckhard and Lee and further elaborated by Nicola and Goyal. We show how taking failure correlation into account extends the corresponding optimization problems, and how they can be treated computationally. In particular, in the (practically relevant) case of components with relatively high reliability, the optimization problems reduce to knapsack–type problems which can, for relevant problem sizes, be solved with rather low computational effort. Also for the general case, solution techniques are outlined. Typical differences between the optimal solutions on the independence assumption and on the assumption of high positive failure correlation, respectively, are demonstrated with the aid of an example.

Keywords: Combinatorial optimization, correlated failures, fault tolerance, multiversion programming, software redundancy, software reliability optimization.

1 Introduction

It is well–known that the reliability of a software system may be increased by the application of redundancy, i.e., by developing several versions of special modules and incorporating them into a fault-tolerant system. Such attempts lead immediately to two types of optimization problems (see [2], p. 310): (1) How should the system be designed to minimize costs under the constraint that a pre–defined reliability level is reached? (2) How should the system be designed to maximize reliability under a certain cost constraint? In their article [2], Belli and Jedrzejowicz investigate the second problem for two fundamental fault–tolerant system structures, namely recovery blocks and multiversion programming structures. Berman and Ashrafi [3] and later Sarper [14] treat a partially extended model, where modules can be used in (possibly) more than one program, for a kind of simplified recovery block scheme. All these articles assume *failure independence* between different versions of a module.

*Department of Statistics, Operations Research and Computer Science, University of Vienna, Universitaetsstrasse 5/9, A-1010 Austria. e-mail: gutjahr@smc.univie.ac.at, Tel.: (+43) 1 4076355 190, Fax: (+43) 4076355 88.

At least since the famous empirical results obtained by Knight and Leveson [9], the last–mentioned independence assumption is usually questioned. Most experts tend to assume a positive correlation between the failure behavior of redundant module versions. Consequently, Belli and Jedrzejowicz suggest in [2] to extend software reliability optimization approaches by taking failure correlations explicitly into account, for example based on the correlation model by Nicola and Goyal [10]. The present paper adopts this suggestion and elaborates some results on optimal redundancy distributions for module versions with *correlated* failures.

As the basic system structure models, we have chosen Model 2 and Model 4 in [3]. (The authors also treat two other models, which, however, do not use redundancy.) Essentially, these models refer to recovery blocks where the testing segments are assumed to be fault–free. Compared to the models in [2], this is a simplification which lends itself as a base for the generalization to the more complicated correlated–failures case.

Our failure correlation model has been taken from [10]. In this article, the authors specialize a general probabilistic approach of describing coincident errors, developed by Eckhardt and Lee [5], to the particular case of beta–distributed failure intensity functions. The "co–reliability" of redundant versions of a software module is then described by two parameters: the mean failure probability $\bar{p}$, and a so–called correlation level θ expressing the degree to which the versions are similar in their failure behavior.

The plan of the paper is the following: In section 2, the investigated problem is defined in a formally precise way. In section 3, two solution techniques are described: an approach based on Dynamic Programming (generalizing the approach in [3]), and a new and computationally more promising technique based on continuous relaxation of the considered discrete optimization problem. Section 4 treats a special boundary case which is nevertheless of vital interest for practical applications: the case where the used module versions are already thoroughly tested, such that their reliability is near to one. On this condition, the optimization problems simplify to some extent. Section 5 contains concluding remarks.

2 Problem Formulation

2.1 The Models by Berman and Ashrafi

We start by recapitulating Model 2 and 4 in [3]. (As already mentioned, the authors also investigate two other models, which are not relevant in our context because they are not concerned with software redundancy.) Compared with [3], our formulation of the models will be more specific insofar as we assume that different versions of one and the same software module have the same reliability level and can be developed or bought at the same costs. The first assumption is an inherent part of the used Nicola–Goyal approach; the second assumption is a consequence of the first since different prices of versions with the same functionality would only be justified in the case of different reliability levels.

Berman and Ashrafi assume a system structure where a program can be divided into several modules. If (as it is done in Model 4) several programs are considered, one module can also be called by more than one program. For each module, functionally equivalent and independently developed versions are available. It is assumed that the reliability of such a version can be estimated, and its costs are known. The optimization process determines the most effective redundancy levels of the modules under a budget constraint: in total, not more than C currency units can be spent.

Similarly as in [3], the following notations are used:

K	Number of programs.
n	Number of modules within the software system.
F_k	Frequency of use of program k $(k = 1, \ldots, K)$.
x_i	Number of versions used for module i $(i = 1, \ldots, n)$.
r_i	Estimated reliability of a version of module i $(i = 1, \ldots, n)$.
R_i	Estimated reliability of module i $(i = 1, \ldots, n)$.
R	Estimated reliability of the software system.
c_i	Cost of developing a version for module i $(i = 1, \ldots, n)$.
C	Available budget.

Model 2 of [3]:

A single program, requiring module $1, \ldots, n$, is considered. In view of the independence of the failure behavior of all module versions, the problem of optimizing the system reliability under the given budget constraint can be formulated as follows:

$$\text{Maximize } R = \prod_{i=1}^{n} R_i$$

$$\text{subject to} \tag{$P1$}$$

$$R_i = 1 - (1 - r_i)^{x_i} \quad (i = 1, \ldots, n),$$

$$\sum_{i=1}^{n} c_i x_i \leq C, \quad x_i \in N = \{1, 2, \ldots\} \quad (i = 1, \ldots, n).$$

As it can be seen, the reliability of module i is defined here as the probability that at least one of the x_i versions of module i performs correctly. This reflects a recovery block system with correctly working test segments.

Model 4 of [3]:

In this model, the system consists of *several* programs. Let S_k denote the set of modules required by program k $(k = 1, \ldots, K)$. If all the modules to be called by the programs are numbered from 1 to n, then $S_k \subseteq \{1, \ldots, n\}$. Program k is used with frequency F_k, so the (expected) reliability of the total system is the weighted average of the reliability values of the single programs, with weights F_k. This leads to the following optimization problem:

$$\text{Maximize } R = \sum_{k=1}^{K} F_k \prod_{i \in S_k} R_i \tag{$P2$}$$

subject to the constraints of (P1).

2.2 Generalization to the Correlated Case

In this subsection, Models 2 and 4 described in subsection 2.1 are generalized to the situation where the failures of different versions of a module are *correlated*. Let us emphasize that (versions of) *different* modules are still assumed to fail independently from each other, which is a realistic assumption since the functional behavior of different modules is different. For modeling the failure correlation of versions of one and the same module, we apply the technique presented in [10], which is based on the theoretical framework developed in [5]. Let us briefly recapitulate the technique:

Let $p(x)$ be the fraction of programs, out of all possible programs written to a given specification, that fail on input x, or, in other words, the probability that a randomly chosen program out of this population would fail on x. If $p(x)$ has the same value for all inputs x, then the failure behavior of the program versions is independent of the input; otherwise, it depends on the input, which induces a correlated failure behavior between the program versions. Now consider the input x as randomly chosen. [1] Then the probability of a program version to fail on this random input is a random variable, denoted by P, which has a certain probability distribution $\mathcal{D}(P)$. Nicola and Goyal suggest to model $\mathcal{D}(P)$ by a *Beta distribution*, a probability distribution on the interval $[0, 1]$ specified by two parameters $\alpha, \beta > 0$. The density of the Beta distribution is given by

$$f(p) = \frac{p^{\alpha-1}\,(1-p)^{\beta-1}}{B(\alpha,\beta)} \quad (0 \le p \le 1),$$

where $B(\alpha, \beta)$ is the Beta function with parameters α and β (see [10], p. 351). Nicola and Goyal transform the parameters α and β by setting

$$\pi = \alpha\,/\,(\alpha+\beta), \quad \theta = 1\,/\,(\alpha+\beta) \tag{1}$$

to two other parameters π and θ, which can be interpreted in the following way: $\pi = \bar{p}$ is the expected value of the random variable P distributed according to $\mathcal{D}(P)$, i.e., the expected fraction of programs failing on a random input, or, in shorter terms, the *expected unreliability* $1-r$. The parameter $\theta > 0$ is a *correlation level:* If $\theta = 0$, the Beta distribution degenerates to a point mass, such that we obtain the case of independent failure behavior of the program versions. On the other hand: if θ is large, then there is a high (positive) correlation between the failures of the versions, the extreme case being $\theta = \infty$, which (as it can be shown) represents identical failure behavior of all program versions. Via the inverse transformation to (1),

$$\alpha = \pi\,/\,\theta, \quad \beta = (1-\pi)\,/\,\theta, \tag{2}$$

the parameters α and β are determined by π and θ.

The resulting distribution of the number of failing program versions out of k given versions is the so–called Beta–binomial distribution, given by the probabilities

$$b_k(i) = \binom{k}{i} \frac{B(\alpha+i, k+\beta-i)}{B(\alpha,\beta)} \quad (i = 0, \ldots, k). \tag{3}$$

[1] Note that the technique works with two different probability models: one for the programs, the other for the inputs. As to other applications of this approach, cf. [6] and [7].

For the purpose of this paper, only the probability that *all* k versions fail is of interest. From (3) one obtains (cf. [3], eq. (2.2)):

$$b_k(k) = \frac{B(\alpha+k,\beta)}{B(\alpha,\beta)} = \frac{(\alpha+k-1)(\alpha+k-2)\ldots\alpha}{(\alpha+\beta+k-1)(\alpha+\beta+k-2)\ldots(\alpha+\beta)}. \tag{4}$$

Thus, if $k = x_i$ versions are developed for module i, and the parameters of the corresponding Beta distribution are α_i and β_i, then the reliability of module i gets

$$R_i = b_{x_i}(x_i) = 1 - \frac{B(\alpha_i+x_i,\beta_i)}{B(\alpha_i,\beta_i)} = 1 - \frac{(\alpha_i+x_i-1)(\alpha_i+x_i-2)\ldots\alpha_i}{(\alpha_i+\beta_i+x_i-1)(\alpha_i+\beta_i+x_i-2)\ldots(\alpha_i+\beta_i)}. \tag{5}$$

It is immediately seen that for $\theta_i \to 0$, i.e., for $\alpha_i \to \infty, \beta_i \to \infty$, the right hand side of (5) tends to

$$1 - (\alpha_i \,/\, (\alpha_i + \beta_i))^{x_i} = 1 - \pi_i^{x_i} = 1 - (1 - r_i)^{x_i}, \tag{6}$$

which yields the representation of R_i in $(P1)$, i.e., the special case of independence.

We are now able to formulate our generalized models for the correlated case:

Model 2':

$$\text{Maximize } R = \prod_{i=1}^{n} R_i$$

$$\text{subject to} \tag{P3}$$

$$\sum_{i=1}^{n} c_i x_i \le C, \quad x_i \in N = \{1,2,\ldots\}\ (i = 1,\ldots,n),$$

$$\text{where } R_i \text{ is given by (5).}$$

Model 4':

$$\text{Maximize } R = \sum_{k=1}^{K} F_k \prod_{i \in S_k} R_i \tag{P4}$$

$$\text{subject to the constraints of } (P3).$$

The reader should notice that different versions of a module i are assumed to have the same parameters α_i and β_i in the failure model. As demonstrated in [10], the parameters α_i and β_i can be *estimated*, based on failure statistics of the modules under consideration. In section 4, it will be shown that, at least on the assumption of high and constant failure correlation between the versions, only one type of parameters has to be estimated for the application of our approach, namely the reliabilities r_i, such that no additional estimations are required, compared to the approach in [2] or [3].

3 Solution Techniques

In this section we turn to the question how to solve the optimization problems ($P3$) and ($P4$) numerically. It should be noticed that in view of the integrality constraints and the nonlinear objective functions, they are "hard" problems (by the way, their NP-hardness can be formally shown), so the application of standard general–purpose optimization software is only successful (within reasonable runtime) for small problem instances. Solution techniques based on the special form of ($P3$) and ($P4$) have to be developed. We start by generalizing the Dynamic Programming (DP) approach of [3] to the correlated–failures case.

3.1 Dynamic Programming

This approach can be applied to Model 2'. Let $R_i(S)$ denote the optimal reliability of the system consisting of the modules $i, i+1, \ldots, n$, if a (remaining) budget of S currency units can be spent for the corresponding module versions. Obviously, $R_1(C)$ is then the optimal reliability of the overall system. Because of the special form (5) of R_i, the following recurrence relation holds:

$$R_i(S) = \max \left(\left[1 - \frac{B(\alpha_i + x_i, \beta_i)}{B(\alpha_i, \beta_i)} \right] \cdot R_{i+1}(S - c_i x_i) \right) \quad (i = 1, \ldots, n),$$

where the maximum is to be taken over all $x_i \in N$ with

$$c_i x_i \leq S - \sum_{k=i+1}^{n} c_k,$$

since the budget S is still available, and for each of the modules $i+1, \ldots, n$, at least one version will be required. Therefore,

$$1 \leq x_i \leq \left\lfloor \left(S - \sum_{k=i+1}^{n} c_i \right) / c_i \right\rfloor .$$

For $i = n$, we obtain as the base of the recursion:

$$R_n(S) = \max \left[1 - \frac{B(\alpha_n + x_n, \beta_n)}{B(\alpha_n, \beta_n)} \right],$$

where the maximum is to be taken over all $x_n \in N$ with

$$1 \leq x_n \leq \lfloor S/c_n \rfloor .$$

Now, by computing the values $R_i(S)$ for each $S \in N$ with

$$\sum_{k=i}^{n} c_k \leq S \leq C - \sum_{k=1}^{i-1} c_k \tag{7}$$

and for $i = n, n-1, \ldots, 1$, the optimal numbers x_i can be determined successively.

If the "granularity" of the cost values c_i and C is coarse, i.e., if they all are small integers or at least small integer multiples of some fixed base value, the described approach may be computationally efficient. Otherwise, the runtime will be rather high, even if n and the resulting values x_i are only small integers. The reason is that in view of (7), optimization must be done for many different values S at each iteration step. In such cases of fine currency granularity, even complete enumeration of all feasible solutions may be faster than the described DP approach. For Model 4', the DP approach is not applicable anymore (cf. [3], p. 1121). These are the reasons why we develop a new, alternative solution technique in the next section.

3.2 An Alternative Solution Technique

For a more efficient solution of the optimization problems $(P3)$ and $(P4)$, we use a technique based on continuous relaxation of the discrete problem formulation; a similar approach has been successfully applied in [8] for the solution of another hard reliability optimization problem.

3.2.1 Continuous Relaxation

Let us omit, for the moment, the integrality constraints on the variables x_i in $(P3)$ and $(P4)$, i.e., we replace $x_i \in N$ by x_i real, $x_i > 0$. It can be observed that the objective function R of the two problems is increasing in the variables x_i: From the well–known representation of the Beta function as an integral it follows that $B(\alpha, \beta)$ is decreasing in α, so the reliability R_i given by (5) is increasing in x_i, which entails that both $\prod R_i$ and $\sum F_k \prod_{i \in S_k} R_i$ are increasing in each variable x_i. Therefore, the constraint $\sum c_i x_i \leq C$ can be replaced by the stronger constraint $\sum c_i x_i = C$ without changing the optima. We obtain in this way from $(P3)$ and $(P4)$ the following continuous, nonlinear optimization problems:

$$\begin{gathered}\text{Minimize } -\prod_{i=1}^{n}\left[1 - \frac{B(\alpha_i + x_i, \beta_i)}{B(\alpha_i, \beta_i)}\right] \\ \text{subject to} \\ \sum_{i=1}^{n} c_i x_i = C, \quad x_i > 0 \ (i = 1, \ldots, n),\end{gathered} \tag{$P5$}$$

resp.

$$\text{Minimize } -\sum_{k=1}^{K} F_k \prod_{i \in S_k}\left[1 - \frac{B(\alpha_i + x_i, \beta_i)}{B(\alpha_i, \beta_i)}\right] \tag{$P6$}$$

$$\text{subject to the constraints of } (P5),$$

which have been formulated as minimization problems according to a convention in nonlinear optimization.

The problems $(P5)$ and $(P6)$ can be solved numerically by appropriate techniques of Nonlinear Optimization. In general, the objective functions of both problems may be non–convex. It can be shown, however, that for values r_i sufficiently close to one, both objective functions get convex functions (we omit the proof for the sake of brevity). Note that in practical applications, the reliabilities of pre–tested module versions are always near one, otherwise the versions would not have passed the tests successfully. Since also the constraints are convex (the are even linear), we obtain convex optimization problems which can be solved in a very efficient way. For the solution of $(P5)$ and $(P6)$ under convexity, Wolfe's algorithm can be applied (see [1], p. 460 f).

3.2.2 The Special Case of Independence

In the special case where $\theta_i = 0$, i.e., for independent or uncorrelated failures, the numerical solution of $(P5)$ gets particularly simple. Let us briefly deal with this situation, although the emphasis of the present paper is on the correlated case. Considering that, in the independent case, R_i is given by (6), the objective function in $(P5)$ reduces to

$$-\prod_{i=1}^{n}[1-(1-r_i)^{x_i}] = -\prod_{i=1}^{n}\left[1-e^{x_i \ln(1-r_i)}\right].$$

For abbreviation, set $\rho_i = -\ln(1-r_i) \quad (i = 1, \ldots, n)$. This yields the problem

$$\begin{gathered}\text{Minimize } -\prod_{i=1}^{n}\left(1-e^{-\rho_i x_i}\right) \\ \text{subject to the constraints of } (P5),\end{gathered} \tag{$P7$}$$

which can be solved by the well–known Lagrange function method with the following result: The solution $(x_1, \ldots, x_n)$ is given by

$$x_i = \frac{c_i}{\rho_i}\ln\left(\frac{\rho_i}{c_i z^*}+1\right) \quad (i = 1, \ldots, n), \tag{8}$$

where z^* is the (unique) root of the function

$$f(z) = \sum_{i=1}^{n}\frac{c_i}{\rho_i}\ln\left(\frac{\rho_i}{c_i z}+1\right) - C. \tag{9}$$

Short computation shows that f is decreasing and convex, so its root z^* can be determined efficiently by Newton's method.

3.2.3 Finding Integer Solutions

Now we return to the general case of arbitrary θ_i. The numerical solution of $(P5)$ or $(P6)$ yields *real* numbers $x_1, \ldots, x_n$; for the application, however, *integers* are required. The simplest way of transforming the real solution into an integer solution is by *rounding*. This may be done as follows: First, round each value $x_i \leq 1$ to the value 1. Then, round each value $x_j > 1$ to the nearest neighbor integer value $x_j^{(r)}$. Check whether the budget constraint remains satisfied. If yes, accept the resulting integer solution. Otherwise, reduce that $x_j^{(r)}$ that has maximum difference $x_j^{(r)} - x_j$ by 1. Repeat until the budget constraint is satisfied.

It is well–known that by rounding the (exact) solution of the continuous relaxation of an integer optimization problem to an integer solution, one only obtains an *approximate* solution of the integer problem. Of course, it may be conjectured that the quality of the approximate solution obtained as described above is in general sufficient for practical purposes. Nevertheless, there may remain doubts about this point. Therefore, in the sequel a technique for improving the rounded solution to the optimal solution, resp. for verifying that the rounded solution is already optimal, shall be presented.

The technique relies on the observation that knowledge of any high–quality solution $(x_1, \ldots, x_n)$ of the considered problem $(P3)$ or $(P4)$ allows the derivation of good lower bounds $x_1^{(l)}, \ldots, x_n^{(l)}$ for the components $x_1, \ldots, x_n$ of the *optimal* solution. After these bounds have been computed, only solution vectors satisfying $x_1 \geq x_1^{(l)}, \ldots, x_n \geq x_n^{(l)}$ *and* the budget constraint have still to be examined. This can be done by complete enumeration or by DP (as described in subsection 3.2.1) with a drastically reduced computational effort. As the example in subsection 3.2.4 will show, the lower bounds may even completely determine the optimal solution, such that no additional computations are necessary.

Model 2':

For Model 2', we obtain the bounds as follows. Let R_i and R denote the reliability of module i and of the overall system, respectively, in the *optimal* solution, and let $\tilde{R}$ denote the reliability of the overall system in an *approximate* solution (e.g., the solution obtained by rounding from the solution of $(P5)$). Then, for optimal $(x_1, \ldots, x_n)$,

$$R_i = 1 - \frac{B(\alpha_i + x_i, \beta_i)}{B(\alpha_i, \beta_i)} \geq R \geq \tilde{R}$$

or

$$B(\alpha_i + x_i, \beta_i) \leq (1 - \tilde{R})\, B(\alpha_i, \beta_i). \tag{10}$$

As mentioned in subsection 3.2.1, the Beta function is decreasing in its first argument, so the function $\phi_\beta(\alpha) = B(\alpha, \beta)$ (β fixed) may be inverted. We denote its inverse by ψ_β. Of course, ψ_β is also decreasing. Applying ψ_{β_i} to both sides of (10) yields

$$x_i \geq \psi_{\beta_i}\left((1 - \tilde{R})\, B(\alpha_i, \beta_i)\right) - \alpha_i. \tag{11}$$

The critical value of x_i is the value $\bar{x}_i$ satisfying (11) with equity. It is characterized by

$$B(\alpha_i + \bar{x}_i, \beta_i) = (1 - \tilde{R})\, B(\alpha_i, \beta_i). \tag{12}$$

The expression on the left hand side is decreasing in $\bar{x}_i$, and its derivative to $\bar{x}_i$ is given by $B_\alpha(\alpha_i + \bar{x}_i, \beta_i)$, where $B_\alpha(\alpha, \beta)$ is the partial derivative of the Beta function to its first argument (it can be computed numerically from the integral representation of the Beta function). Therefore, $\bar{x}_i$ can be determined by Newton's method, such that finally from (11) the bound

$$x_i \geq \lceil \bar{x}_i \rceil \tag{13}$$

on the ith component of the optimal solution is obtained.

Let us remark that in the special case $\theta_i = 0$ of independence, (12) takes the form

$$(1 - r_i)^{\bar{x}_i} = 1 - \tilde{R}$$

(cf. (5) – (6)), which has the solution

$$\bar{x}_i = \frac{\ln(1-\tilde{R})}{\ln(1-r_i)} = -\frac{\ln(1-\tilde{R})}{\rho_i}. \tag{14}$$

Model 4':

The derivation of bounds for the components of the optimal solution of $(P4)$ is more complicated insofar as $R_i \geq R$ needs not to hold for Model 4'. Another basic connection relating R_i with $\tilde{R}$ is required in this context. We get a lower bound for R_i as follows:

Let $T_i = \{k \mid i \in S_k\}$. Then for fixed i,

$$\begin{aligned}\tilde{R} \leq R &= \sum_{k \in T_i} F_k R_i \prod_{j \in S_k, j \neq i} R_j + \sum_{k \notin T_i} F_k \prod_{j \in S_k} R_j \\ &\leq R_i \sum_{k \in T_i} F_k + \sum_{k \notin T_i} F_k,\end{aligned}$$

hence

$$R_i \geq \left(\tilde{R} - \sum_{k \notin T_i} F_k\right) / \sum_{k \in T_i} F_k. \tag{15}$$

The derivation of a lower bound for optimal x_i can now be based upon the last inequality, following the same line as for Model 2': All we have to do is to replace $\tilde{R}$ by the right hand side of (15).

3.2.4 Numerical Example I

We illustrate the technique of subsections 3.2.1 – 3.2.3 by a small numerical example. In order to keep the mathematical computations as transparent as possible, $\theta_i = 0$ (independent failure behavior) has been assumed. (In subsection 4.3, an example with high θ_i values will be presented.) Already in this example with coarse currency granularity, the computational effort required by our relaxation technique turns out to be significantly smaller than that required by the DP approach.

Input: Model 2 with

$$r_1 = 0.99, \quad r_2 = 0.95, \quad r_3 = 0.999; \quad c_1 = 1, \quad c_2 = 1, \quad c_3 = 1; \quad C = 10.$$

Solution:

One obtains:

$$\rho_1 = -\ln 0.01 = 4.6051..., \quad \rho_2 = -\ln 0.05 = 2.9957..., \quad \rho_3 = -\ln 0.001 = 6.9077...$$

The function $f(z)$ defined by (9) now reads

$$f(z) = \frac{1}{4.605} \ln\left(\frac{4.605}{z} + 1\right) + \frac{1}{2.996} \ln\left(\frac{2.996}{z} + 1\right) + \frac{1}{6.908} \ln\left(\frac{6.908}{z} + 1\right) - 10.$$

Computing the root of $f(z)$ by Newton's method yields

$$z^* = 2.331 \cdot 10^{-6}.$$

Inserted into (8), this leads to

$$x_1 = 3.148 \approx 3, \quad x_2 = 4.698 \approx 5, \quad x_3 = 2.157 \approx 2.$$

The rounding $(3, 5, 2)$ satisfies the budget constraint, so it is accepted. This solution produces a system reliability of

$$\tilde{R} = (1 - 0.01^3)\,(1 - 0.05^5)\,(1 - 0.001^2) = 0.9999977.$$

From (14) and (13),

$$x_1 \geq \lceil 2.82 \rceil = 3, \quad x_2 \geq \lceil 4.33 \rceil = 5, \quad x_3 \geq \lceil 1.88 \rceil = 2$$

is obtained, which verifies the solution $x_1 = 3$, $x_2 = 5$, $x_3 = 2$ as already optimal, since it is the only solution satisfying both the bounds above and the budget constraint.

4 Highly Reliable Software

As already argued in section 3.2.1, the reliability r_i of a (intensely pre–tested) module version can be assumed to be close to one. Even for software on a moderate reliability level, only one failure in $10^3 - 10^7$ hours is expected (see [4]). Using the rough estimate that a program run takes one hour in the average, this results in an unreliability $1 - r_i$ of less than 0.001. So it makes sense to investigate how the models and optimization problem formulations of sections 2 – 3 behave in the limiting case where the reliabilities r_i tend to one.

4.1 Approximation in the Uncorrelated Case

As a warming up, we start with Model 2 (i.e., with Model 2' under independence). It will be shown that in an approximation for $r_i \approx 1$, problem $(P1)$ simplifies to a *knapsack problem* (see, e.g., [13]).

Let $\pi_i = 1 - r_i$ denote the unreliability of a version of module i. We suppose $\pi_i \ll 1$ for $i = 1, \ldots, n$ and $\pi_i^{m+1} \ll \pi_j^m$ for $i \neq j$. Then, the objective function $R = \prod R_i$, with R_i given by (6), can be approximated as follows:

$$R = \prod_{i=1}^{n} (1 - \pi_i^{x_i}) \approx 1 - \sum_{i=1}^{n} \pi_i^{x_i} \approx 1 - \sum_{i \in I} \pi_i^{x_i} = 1 - \sum_{i \in I} \pi_i^k,$$

where I is the set of all indices i for which x_i is minimal among $x_1, \ldots, x_n$, and k is the minimal value of x_i.

Taking instead of R its approximate value $1 - \sum_{i \in I} \pi_i^k$ as the objective function to be maximized, we obtain the problem

$$\text{Minimize} \sum_{i \in I} \pi_i^k$$

$$\text{subject to} \tag{P8}$$

$$k = \min(x_1, \ldots, x_n), \quad I = \{i \,|\, x_i = k\}, \quad \sum_{i=1}^{n} c_i x_i \leq C, \quad x_i \in N \ (i = 1, \ldots, n).$$

It can be observed that only those components i enter into the objective function of $(P8)$ that occur with the least redundancy in the solution. Therefore, as short reflexion shows, an optimal solution can only be of the following type:

$$x_i = k \quad \text{or} \quad x_i = k + 1 \quad (i = 1, \ldots, n). \tag{16}$$

It is now easy to determine the optimal solution: Considering (16), it is clear that k is given by

$$k = \left\lfloor C \,/ \sum_{i=1}^{n} c_i \right\rfloor. \tag{17}$$

Now set $y_i = x_i - k$, such that $y_i = 0$ for $x_i = k$ and $y_i = 1$ for $x_i = k + 1$. Then $(P8)$ may be rewritten as

$$\text{Maximize} \sum_{i=1}^{n} \pi_i^k y_i$$

$$\text{subject to} \tag{P9}$$

$$\sum_{i=1}^{n} c_i y_i \leq C - k \sum_{i=1}^{n} c_i, \quad y_i \in \{0, 1\} \ (i = 1, \ldots, n).$$

This is a knapsack problem. For solution techniques, see [13], [11] or [12].

4.2 Approximation in the Strongly Correlated Case

Now let us do the high–reliability approximation for failure correlations of medium or high amount: We suppose $\theta_i \gg \pi_i$, i.e., the values θ_i are of a larger order of magnitude than the unreliabilities $\pi_i = 1 - r_i$. As in subsection 4.1, the values π are assumed to be small compared to one. So the following approximations hold in cases of highly reliable module versions with non–negligible failure correlation. Expressing the parameters α_i and β_i in (5) via (2) by the parameters π_i and θ_i, we obtain

$$R_i = 1 - \frac{(\pi_i + \theta_i(x_i - 1))\,(\pi_i + \theta_i(x_i - 2)) \ldots \pi_i}{(1 + \theta_i(x_i - 1))\,(1 + \theta_i(x_i - 2)) \ldots 1}. \tag{18}$$

Now, as $\pi_i \ll \theta_i$, the denominator on the right hand side gets $\approx \pi_i \, \theta_i^{x_i - 1} \, (x_i - 1)! \ll 1$. Therefore, the objective function R of $(P3)$ can be approximated as

$$R = \prod_{i=1}^{n} R_i \approx 1 - \sum_{i=1}^{n} \frac{\pi_i \, \theta_i^{x_i - 1} \, (x_i - 1)!}{(1 + \theta_i(x_i - 1))\,(1 + \theta_i(x_i - 2)) \ldots 1},$$

which yields the problem

$$\text{Minimize} \quad \sum_{i=1}^{n} \frac{\pi_i \, \theta_i^{x_i-1} \, (x_i-1)!}{(1+\theta_i(x_i-1))\,(1+\theta_i(x_i-2))\ldots 1}$$

subject to the constraints of $(P3)$.

For *high* failure correlations, this formulation further simplifies. Assume now even $\theta_i \gg 1$. Setting $\delta_i = 1/\theta_i \ll 1$, the fraction in the objective function of the problem above gets

$$\frac{\pi_i \, (x_i-1)!}{((x_i-1)+\delta_i)\,((x_i-2)+\delta_i)\ldots(1+\delta_i)} \approx \frac{\pi_i \, (x_i-1)!}{(x_i-1)! \, \left[1+\delta_i \sum_{j=1}^{x_i-1} \frac{1}{x_i-j}\right]}$$

$$\approx \pi_i \, \left[1-\delta_i \sum_{j=1}^{x_i-1} \frac{1}{x_i-j}\right] = \pi_i \, \left[1-\delta_i \sum_{j=1}^{x_i-1} \frac{1}{j}\right].$$

Since $\pi_i \delta_i = \pi_i/\theta_i = \alpha_i$, the following problem is obtained:

$$\text{Maximize} \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{x_i-1} \frac{1}{j} \tag{P10}$$

subject to the constraints of $(P3)$.

The term $\sum_{j=1}^{x_i-1}(1/j)$ may also be written as $H(x_i-1)$, where $H(n) = \sum_{j=1}^{n}(1/j)$ is the nth harmonic number.

In the special case where the correlation level θ_i is constant for all modules, we have $\alpha_i =$ const $\cdot \pi_i$. Then the objective of $(P10)$ is replaced by

$$\text{Maximize} \sum_{i=1}^{n} \pi_i \sum_{j=1}^{x_i-1} \frac{1}{j}. \tag{19}$$

In this case of high, constant failure correlation, only the reliabilities r_i enter (via $\pi_i = 1-r_i$) into the optimization problem, i.e., no other parameters have to be estimated.

The problem $(P10)$ may be interpreted as follows: The mth version of module i, shortly denoted by $V_i(m)$, achieves an increase of the objective function by $\alpha_i/(m-1)$ at additional costs of c_i currency units. This is a knapsack problem with the following additional constraint: $V_i(m)$ is only allowed to be selected, if also $V_i(1)$, $V_i(2)$, … $V_i(m-1)$ have been selected. Without going into details, let us mention that this problem can be solved by a modification of the Branch–and–Bound algorithm for the knapsack problem described in [12]. The additional constraint even reduces the runtime, since it restricts the solution space.

Without proof we remark that an analogous approximation for Model 4' leads to an optimization problem of the same form as $(P10)$ and can therefore be treated by the same technique.

4.3 Numerical Example II

Using the methods of the last sections, we analyze an example with high module reliabilities on two opposite assumptions: (a) no failure correlation, (b) high failure correlation.

The results will illustrate how failure correlations influence the solution of the reliability optimization problem.

Input: Model 2' with

$$r_1 = 0.9995,\ r_2 = 0.99995,\ r_3 = 0.99998,\ r_4 = 0.99999;$$

$$c_1 = 50,\ c_2 = 50,\ c_3 = 100,\ c_4 = 100;$$

$$C = 700.$$

Solution:

(a) Uncorrelated case: By (17), one finds $k = 2$. Then, $(P9)$ yields

$$\text{Maximize} \quad 0.0005^2 y_1 + 0.00005^2 y_2 + 0.00002^2 y_3 + 0.00001^2 y_4$$

$$\text{subject to}$$

$$50y_1 + 50y_2 + 100y_3 + 100y_4 \le 100, \quad y_i \in \{0, 1\}\ (i = 1, \dots, 4).$$

The solution of this knapsack problem is $(1, 1, 0, 0)$, which leads to $x_1 = 3$, $x_2 = 3$, $x_3 = 2$, $x_4 = 2$ as the solution of the original problem.

(b) Highly correlated case with constant correlation level: $(P10)$ with objective function (19) yields

$$\text{Maximize} \quad 0.0005\, H(x_1{-}1){+}0.00005\, H(x_2{-}1){+}0.00002\, H(x_3{-}1){+}0.00001\, H(x_4{-}1)$$

$$\text{subject to}$$

$$50x_1 + 50x_2 + 100x_3 + 100x_4 \le 700, \quad x_i \in N\ (i = 1, \dots, 4),$$

where $H(n)$ denotes again the nth harmonic number. Setting $y_i = x_i - 1$ and multiplying objective function and constraints by suitable constants, we may rewrite this as

$$\text{Maximize} \quad 50H(y_1) + 5H(y_2) + 2H(y_3) + H(y_4)$$

$$\text{subject to}$$

$$y_1 + y_2 + 2y_3 + 2y_4 \le 8, \quad y_i \in N \cup \{0\}\ (i = 1, \dots, 4).$$

Applying the interpretation suggested at the end of subsection 4.2, one finds that even the 8th additional version of module 1 still achieves an increase of the objective function of $(1/8) \cdot 50 = 6.25$, while the first additional version of module 2 would only achieve an increase of 5. So the optimal solution is $x_1 = 9$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$.

A comparison with the optimal solution under independence (see (a)) shows that failure independence leads here to rather balanced redundancy levels in the optimal solution, while under high failure correlation, the optimal solution rather tends to concentrate redundancy on cheap and/or unreliable modules.

5 Conclusion

We have presented nonlinear integer programming formulations for the problem of maximizing the reliability of two particular types of redundant software systems under fixed budget constraints. Redundant versions of modules were not considered anymore as independent in their failure behavior (as in prior investigations); instead, the Nicola–Goyal model of failure correlations has been used for the computation of the system reliability. Solution techniques for the resulting optimization problems have been presented.

Future work has to be done mainly in two directions: First, the described numerical solution techniques should be implemented in a black–box manner such that they can also be applied by users who are not familiar with nonlinear and combinatorial optimization. Experiments with this kind of software will certainly further improve the techniques.

Secondly, more complex redundant system structures should be made accessible to reliability optimization along the lines of the presented approach. In particular, the two types of fault–tolerant system structures investigated in [2] on the assumption of failure independence, namely recovery blocks (with possibly faulty test segments) and voting–system–based multiversion programming structures, deserve a treatment by optimization models that include failure correlation.

References

[1] Bazaraa, M. S., Sherali, H. D., Shetty, C. M., *Nonlinear Programming: Theory and Algorithms*, Wiley (1993).

[2] Belli, F., Jedrzejowicz, P., "An approach to the reliability optimization of software with redundancy", *IEEE Trans. Software Eng.*, vol. SE-17, 310 – 312 (1991).

[3] Berman, O., Ashrafi, N., "Optimization models for reliability of modular software systems", *IEEE Trans. Software Eng.*, vol. SE-19, 119 – 1123 (1993).

[4] Butler, R. W., Finelli, G. B., "The infeasibility of quantifying the reliability of life–critical software", *IEEE Trans. Software Eng.*, vol. SE-19, 3 – 2 (1993).

[5] Eckhardt, D. E., Lee, L. D., "A theoretical basis for the analysis of multiversion software subject to coincident errors", *IEEE Trans. Software Eng.*, vol. SE-11, 1511 – 1517 (1985).

[6] Gutjahr, W. J., "Optimal test distributions for software failure cost estimation", *IEEE Trans. Software Eng.*, vol. SE-21, 219 – 228 (1995).

[7] Gutjahr, W. J., "Importance sampling of test cases in Markovian software usage models", *Probability in the Engineering and Informational Sciences*, vol. 11, 19 – 26 (1997).

[8] Gutjahr, W. J., Pflug, Ch., Ruszczynski, A., "Configurations of series–parallel networks with maximum reliability", *Microelectron. Reliab.*, vol. 36, 247 – 253 (1996).

[9] Knight, J. C., Leveson, N. G., "An experimental evaluation of the assumption of independence in multiversion programming", *IEEE Trans. Software Eng.*, vol. SE-12, 96 – 109 (1986).

[10] Nicola, V. F., Goyal, A., "Modeling of correlated failures and community error recovery in multiversion software", *IEEE Trans. Software Eng.*, vol. SE-16, 350 – 359 (1990).

[11] Parker, R. G., Rardin, R. L., *Discrete Optimization*, Academic Press (1988).

[12] Martello, S., Toth, P., *Knapsack Problems: Algorithms and Computer Implementation*, Wiley (1990).

[13] Papadimitriou, H., Steiglitz, I., *Combinatorial Optimization: Algorithms and Complexity*, Prentice–Hall (1982).

[14] Sarper, H., "No special schemes are needed for solving software reliability optimization problems", *IEEE Trans. Software Eng.*, vol. SE-21, 701 – 702 (1995).