# Reliability Allocation

## Introduction

Many systems are implemented by using a set of interconnected subsystems. While the architecture of the overall system may often be fixed, individual subsystems may be implemented differently. A designer needs to either achieve the target reliability while minimizing the total cost, or maximize the reliability while using only the available budget. Intuitively, some of the lowest-reliability components may need special attention to raise the overall reliability level. Such an optimization problem may arise while designing a complex software or a computer system. Such problems also arise in mechanical or electrical systems. A number of studies since 1960 have examined such problems [1].

In a nonredundant system, all the subsystems are essential. Often, however, an individual subsystem can be made more reliable by using a more costly implementation. This additional cost may represent wider columns in a building or more thorough testing of software. In redundant implementations, higher reliability can sometimes be achieved by using several copies of a subsystem, such that the system forms a parallel or ***k*-out-of-*n*** configuration.

The next section considers the problem formulation, followed by approaches used for setting up an optimization problem. As an example, software reliability allocation is examined in detail with two numerical illustrations. The last section considers reliability allocation in complex systems.

## Problem Formulation

We assume that a system has been designed at a higher level as an assembly of appropriately connected subsystems. In general, the functionality of each subsystem can be unique; however, there can be several choices for many of the subsystems providing the same functionality, but differently reliability levels.

Here we consider the problem formulation for a common and widely applicable case. Let there be $n$ subsystems $\mathrm{SS}_i, i = 1, \ldots, n$, each with reliability $R_i$ and cost $C_i$. Let the cost $C_i$ be a function of the reliability given by $f_i(R_i)$, it is referred to as the *cost function* below. Let $C_\mathrm{s}$ and $R_\mathrm{s}$ represent the total system cost and the overall reliability and $R_\mathrm{ST}$ be the specified target reliability. If all the subsystems are essential to the system and if their failures are statistically independent, the system can be modeled as a *series system.* The cost minimization problem can be stated as follows:

$$\text{Minimize } C_\mathrm{s} = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(R_i) \qquad (1)$$

Subject to

$$R_\mathrm{ST} \leq R_\mathrm{s}$$

$$i.e.\ R_\mathrm{ST} \leq \prod_{i=1}^{n} R_i \text{ since } R_\mathrm{S} = \prod_{i=1}^{n} R_i \qquad (2)$$

Note that equation (1) assumes that the cost of interconnecting the subsystems is negligible. An alternative problem would be to maximize $\mathrm{R}_s$ while keeping $\mathrm{C}_s$ less than or equal to the allocated cost budget.

The $i$th subsystem $\mathrm{SS}_i$ may have several implementation choices with different reliability values:

1. By extending a continuous attribute (for example, diameter of a column in building or time spent for software testing) the subsystem can be made more reliable.
2. Different vendors may offer their own implementations of $\mathrm{SS}_i$ at different costs.
3. It may be possible to use multiple copies of $\mathrm{SS}_i$ (for example, double wheels of a truck) to achieve higher reliability. Often the number of copies is constrained between a minimum (often one) and a maximum number because of implementation issues.

Note that in the first case, both the cost and the reliability can be varied continuously, whereas in the other two cases, the choices are discrete. In the first case, we can define a continuous cost function. In the second case too, the market forces may impose a cost function. In the third case, the subsystem may be modeled as a *parallel* or *k-out-of-n system* for reliability evaluation, provided the failures are statistically independent. A number of publications on reliability allocation consider only the third case, where the optimization problem becomes an integer optimization problem. It becomes a 0–1 optimization problem when choices are discrete and a component from a given list of candidates is either used or not used [2].

## Approaches for Problem Setup

It is reasonable to assume that the cost function $f_i$ would satisfy the following three conditions [3]:

1. $f_i$ is a positive function
2. $f_i$ is nondecreasing
3. $f_i$ increases at a higher rate for higher values of $R_i$.

The third condition suggests that it can be very expensive to achieve the reliability value of one. In fact, for software, it has to been shown that under some assumptions, it is infeasible to achieve ultrahigh reliability [4].

In some cases, the cost function can be derived from basic considerations, as we will do below for **software reliability**. In other cases, it may be derived empirically by compiling data for different choices. The cost function is often stated in terms of the reliability, for example, the cost function proposed by Mettas [3] is given by

$$C_i(R_i, p_i, R_{i,\min}, R_{i,\max}) = \exp\left((1-p_i)\frac{R_i - R_{i,\min}}{R_{i,\max} - R_i}\right) \tag{3}$$

where $R_{i,\min}$ and $R_{i,\max}$ are the minimum and maximum values of $R_i$ and $p_i$ is parameter ranging between zero and one that represents the relative difficulty of increasing a component's reliability. The cost function can also be given in terms of the failure rate as illustrated below for software reliability allocation.

A useful transformation of equation (2) can be obtained by taking the logarithms of both sides of the equation [5–7].

$$\ln(R_{\mathrm{ST}}) \le \sum_{i=1}^{n} \ln(R_i) \tag{4}$$

The transformation in equation (4) can sometimes reduce the problem to a linear optimization problem. The term $\ln(R_i)$ can also have a well-defined physical significance, in some cases, as the failure rate. When the failure rate of a subsystem $\mathrm{SS}_i$ is constant, its reliability is given by an exponential relationship $R_i(t) = \exp(-\lambda_i t)$, the system failure rate is given by the summation of the subsystem failure rates and hence equation (4) can be restated as

$$\lambda_{\mathrm{ST}} \ge \sum_{i=1}^{n} \lambda_i \tag{5}$$

The failure rate itself is a major reliability attribute. In some cases, such as in software reliability engineering, it is the failure rate that is often specified [8, 9]. The cost function of a subsystem can also be given in terms of its failure rate. If the cost $C_i$ is given by the function $f_i(\lambda_i)$, equation (1) can be restated as

$$\text{Minimize } C_{\mathrm{S}} = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(\lambda_i) \tag{6}$$

For example, let us consider software reliability. When a software is tested, defects in it are found and removed by debugging. A program tested more thoroughly will have fewer bugs and hence higher reliability. Several models relating software **reliability growth**, with the time spent in testing, have been proposed and validated. These are termed *software reliability growth models* (SRGMs). For the popular *exponential software reliability growth model* [8–10], the failure rate as a function of testing time $d$ is given by

$$\lambda_i(d) = \lambda_{0i} \exp(-\beta_i d) \tag{7}$$

where $\lambda_0$ and $\beta_1$ are the SRGM parameters. If we assume that the cost is dominated by the testing time, the cost is given by the following function, which satisfies the three conditions mentioned above.

$$d(\lambda_i) = \frac{1}{\beta_i} \ln\left(\frac{\lambda_{0i}}{\lambda_i}\right) \tag{8}$$

## Reliability Allocation Approaches for Basic Series and Parallel Systems

The reliability allocation problem for two basic reliability structures *series* and *parallel* can be solved by linearizing the constraints [1, 2, 7]. In a *series* system, the constraint is given in equation (2) above, which can be linearized by rewriting it as

$$\ln(R_{\mathrm{ST}}) \le \sum_{i=1}^{n} \ln(R_i) \tag{9}$$

which may then be solved relatively easily. An example is given below for software reliability allocation.

In a *parallel system*, functionally identical subsystems are configured such that correct operation of at least one of them assures a correctly functioning system. It is assumed that any overhead in implementing such a system is negligible. In real systems, the overhead involved will result in a lower level of reliability. In a number of studies, the problem assumes that the reliability of a subsystem can be increased by using a functionally identical component in parallel [2, 7]. For **parallel systems**, the constraint is given by

$$R_{\mathrm{ST}} \leq 1 - \prod_{i=1}^{n} (1 - R_i) \tag{10}$$

The constraint can be linearized by using logarithms of the complements of reliability. Therefore, equation (10) can be rewritten as

$$\ln(1 - R_{\mathrm{ST}}) \geq \sum_{i=1}^{n} \ln(1 - R_i) \tag{11}$$

Elegbede *et al.* have recently shown [7] that if the cost function satisfies the three properties given above, the cost is optimal if all the parallel components have the same cost. For software, computer hardware, and mechanical systems, the number of discrete parallel component is likely to be very small.

## Reliability Allocation for Software Systems

As a detailed example, we examine the problem of software reliability allocation [6]. Typically, a software consists of sequentially executed blocks, such that only one is under execution at a time. Each block can be independently tested and debugged to reduce the failure rate below a target value. In some cases, the reliability of a block can be further increased by replication. For replication to be effective, each replicated version must be developed independently such that the failures are relatively independent. The impact of replication can be evaluated by assuming statistical independence. However, it has been shown that sometimes the statistical **correlation** can be significant, requiring more complex analysis.

Here, in this example we consider the common case, a nonredundant implementation of software, divided into $n$ sequential blocks [6]. Let us assume that a block $i$ is under execution for a fraction $x_i$ of the time where $\Sigma x_i = 1$ [5]. Then the reliability allocation problem can be written as

$$\text{Minimize } C = \sum_{i=1}^{n} \frac{1}{\beta_i} \ln\left(\frac{\lambda_{0i}}{\lambda_i}\right) \tag{12}$$

$$\text{Subject to } \lambda_{\mathrm{ST}} \leq \sum_{i=1}^{n} x_i \lambda_i \tag{13}$$

**Solution:** Let us solve the problem posed by equations (12) and (13) by using the Lagrange multiplier approach by finding the minimum of

$$F(\lambda_1, \ldots, \lambda_n) = C + \theta(\lambda_1 + \cdots + \lambda_n) - \lambda_{\mathrm{ST}} \tag{14}$$

where $\theta$ is the Lagrange multiplier. The necessary conditions for the minimum to exist are (a) the partial derivatives of the function $F$ are equal to zero, (b) $\theta > 0$, and (c) $x_1\lambda_1 + x_2\lambda_2 + \cdots + x_n\lambda_n = \lambda_{\mathrm{ST}}$ [6]. Equating the partial derivatives to zero and using the third condition, the solutions for the optimal failure rates are found as follows:

$$\lambda_1 = \frac{\dfrac{\lambda_{\mathrm{ST}}}{x_1}}{\displaystyle\sum_{i=1}^{n} \frac{\beta_1}{\beta_i}} \quad \lambda_2 = \frac{\beta_1 x_1}{\beta_2 x_2}\lambda_1 \quad \cdots \quad \lambda_n = \frac{\beta_1 x_1}{\beta_n x_n}\lambda_1 \tag{15}$$

The testing times of the individual modules are given by equation (12). The optimal values of $d_1$ and $d_i, i \neq 1$ are given by

$$d_1 = \frac{1}{\beta_1} \ln\left(\frac{\lambda_{10} x_1 \displaystyle\sum_{i=1}^{n} \frac{\beta_1}{\beta_i}}{\lambda_{\mathrm{ST}}}\right) \quad \text{and}$$

$$d_i = \frac{1}{\beta_i} \ln\left(\frac{\lambda_{i0}\beta_i x_i}{\lambda_1 \beta_1 x_1}\right) \tag{16}$$

Note that $d_i$ is positive if $\lambda_i \leq \lambda_{i0}$. The testing time for a block must be nonnegative. If any of the testing times in equation (16) is negative, the optimization problem must be solved iteratively [6].

In software reliability engineering, the assumptions involved in formulation of the exponential model imply that the parameter $\beta_i$ is inversely proportional to the software size [8, 11], when measured in terms of the lines of code. The value of $x_i$ can be

**Table 1** Input data and optimal values for Example 1

| Block | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ |
|---|---|---|---|---|---|
| $\beta_i$ | $7 \times 10^{-3}$ | $3.5 \times 10^{-3}$ | $2.333 \times 10^{-3}$ | $7 \times 10^{-4}$ | $3.5 \times 10^{-4}$ |
| $\lambda_{i0}$ | 0.14 | 0.14 | 0.14 | 0.175 | 0.21 |
| $x_i$ | 0.028 | 0.056 | 0.083 | 0.278 | 0.556 |
| Optimal $\lambda_i$ | 0.06 | 0.06 | 0.06 | 0.06 | 0.06 |
| Optimal $d_i$ | 121.043 | 242.085 | 363.128 | $1.529 \times 10^3$ | $3.579 \times 10^3$ |

reasonably assumed to be proportional to the code size. The values of $\lambda_i$ and $\lambda_{i0}$ do not depend on size but depend on the initial defect densities [11]. Therefore, if the exponential model indeed holds, the equation (15) states that the optimal values of the post-test failure rates $\lambda_1, \ldots, \lambda_n$ are equal. In addition, if all the initial defect densities are also equal for all the blocks, then the optimal test times for each module is proportional to its size.

**Example 1** A software system uses five functional blocks B1–B5. We construct this example assuming sizes 1, 2, 3, 10, and 20 KLOC (thousand lines of code) respectively, and the initial defect densities of 20, 20, 20, 25, and 30 defects per KLOC respectively. Let us assume that measured parameter values are given in the top three rows, which are the inputs to the optimization problem. The solution obtained using equations (15) and (16) are given in the two bottom rows. Let us now minimize the test cost such that the overall failure rate is less than or equal to 0.06 per unit time. Here the time units can be hours of testing time or hours of CPU time used for testing. (See Table 1).

Note that the optimal values of $\lambda_i$ for the five modules are equal, even though they start with different initial values. This requires a substantial part of the test effort allocated to largest blocks. The total cost in terms of testing is $5.835 \times 10^3$ h.

**Example 2** This is identical to the previous example with one difference, the numbers in the second row for $\lambda_{i0}$ are obtained assuming all five modules have the same defect density value of 20 per KLOC. (See Table 2).

We note that for this example, the optimal testing time is exactly proportional to the software size, block $B_5$ is tested for 20 times the time used for $B_1$. The final failure rates are identical for the five blocks.

The above discussion suggests that some preliminary rules may be used for obtaining initial apportionments. Some apportionment rules have been suggested in the literature [5].

**Equal reliability apportionment** For example, one can test a set of software blocks, such that at the end they all individually have the failure rate equal to the target failure rate for the system.

**Complexity-based apportionment** For example, the software size itself is a complexity metric. Therefore, the available test time can be apportioned in proportion to the software size.

**Impact-based apportionment** A block that is executed more frequently, or is more critical in terms of failures, should be assigned more resources.

## Reliability Allocation for Complex Systems

In practice, many cases can be complex and may require an iterative approach [1, 2, 7, 12]. Such an approach is also needed if the objective function has multiple objectives and includes both the total cost

**Table 2** Input data and optimal values for Example 2

| Block | $B_1$ | $B_2$ | $B_3$ | $B_4$ | $B_5$ |
|---|---|---|---|---|---|
| $\beta_i$ | $7 \times 10^{-3}$ | $3.5 \times 10^{-3}$ | $2.333 \times 10^{-3}$ | $7 \times 10^{-4}$ | $3.5 \times 10^{-4}$ |
| $\lambda_{i0}$ | 0.14 | 0.14 | 0.14 | 0.14 | 0.14 |
| $x_i$ | 0.028 | 0.056 | 0.083 | 0.278 | 0.556 |
| Optimal $\lambda_i$ | 0.06 | 0.06 | 0.06 | 0.06 | 0.06 |
| Optimal $d_i$ | 121.043 | 242.085 | 363.128 | $1.21 \times 10^3$ | $2.421 \times 10^3$ |

and the system reliability [1]. These steps are based on [12]:

1. Design the system using functional subsystems.
2. Perform an initial apportionment of cost or reliability attributes based on suitable apportionment rules or preliminary computation.
3. Predict system reliability.
4. Determine if reallocation is feasible and will enhance the objective function. If so, perform reallocation.
5. Repeat until optimality is achieved.
6. See if this meets the objectives. If not, consider returning to step 1 and revising the design.
7. Finalize the design with recommended reliability allocation and the cost projections.

The optimization methods used in steps 2–5 above can be classified into three approaches [1].

1. **Exact methods** When the problem is not large, exact methods can be desirable. In general, the problem can be a nonlinear optimization problem. In a few cases, the problem can be transformed into a linear problem, as shown in the example above.
2. **Heuristics-based methods** Several heuristics for reliability allocation have been developed. Many of them are based on identifying the variable to which the solution is most sensitive and incrementing its value.
3. **Metaheuristic algorithms** These algorithms are based on artificial reasoning. The best known of them are ***genetic algorithms***, *simulated annealing*, and *tabu-search*. These algorithms can be useful when the search space is large and approximate results are sought.

Some general purpose [13] and special purpose [6] software tools have been developed that can simplify setting up and solving the optimal allocation problem. The reliability allocation problem can also be formulated to address other reliability attributes like availability [14] or maintainability.

## References

[1] Kuo, W. & Prasad, V.R. (2000). An annotated overview of system-reliability optimization, *IEEE Transactions on Reliability* **49**, 176–187.

[2] Majety, S.R.V., Dawande, M. & Rajgopal, J. (1999). Optimal reliability allocation with discrete cost-reliability data for components, *Operations Research* **47**, 899–906.

[3] Mettas, A. (2000). Reliability allocation and optimization for complex systems, in *Proceedings Annual Reliability and Maintainability Symposium*, Los Angeles, January 2000, pp. 216–221.

[4] Butler, R.W. & Finelli, G.B. (1993). The infeasibility of quantifying the reliability of life-critical real-time software, *IEEE Transactions on Software Engineering* **19**, 3–12.

[5] Lakey, P.B. & Neufelder, A.M. (1996). *System and Software Reliability Assurance Notebook*, Rome Laboratory, Rome, pp. 6.1–6.24.

[6] Lyu, M.R., Rangarajan, S. & van Moorsel, A.P.A. (2002). Optimal allocation of test resources for software reliability growth modeling in software development, *IEEE Transactions on Reliability* **51**, 183–192.

[7] Elegbede, A.O.C., Chengbin, C., Adjallah, K.H. & Yalaoui, F. (2003). Reliability allocation through cost minimization, *IEEE Transactions on Reliability* **52**, 106–111.

[8] Musa, J.D., Iannini, A. & Okumoto, K. (1897). *Software Reliability, Measurement, Prediction, Application*, McGraw-Hill.

[9] Lyu, M.R. (ed) (1995). *Handbook of Software Reliability Engineering*, McGraw-Hill.

[10] Goel, A.L. & Okumoto, K. (1979). Time-dependent error detection rate model for software and other performance measures, *IEEE Transactions on Reliability* **28**, 206–211.

[11] Malaiya, Y.K. & Denton, J. (1997). What do the software reliability growth model parameters represent? in *International Symposium on Software Reliability Engineering*, Albuquerque, pp. 124–135.

[12] NASA Document. (2004). Reliability allocation, Oct 1, 2004, http://www.foia.msfc.nasa.gov/docs/NAS8-00 179/S&MAOI/R-Reliability/QD-R-008.pdf.

[13] ReliaSoft. (2003). Reliability importance and optimized reliability allocation (analytical), http://www.weibull.com/SystemRelWeb/blocksimtheory.htm.

[14] Gurov, S.V., Utkin, L.V. & Shubinsky, I.B. (1995). Optimal reliability allocation of redundant units and repair facilities by arbitrary failure and repair distributions, *Microelectronics Reliability* **35**(12), 1451–1460.

## Related Articles

***k*-out-of-*n* Systems**; **Parallel, Series, and Series–Parallel Systems**.

Yashwant K. Malaiya