54. Optimal System Design

The first section of this chapter describes various applications of optimal system design and associated mathematical formulations. Special attention is given to the consideration the randomness associated with system characteristics. The problems are described from the reliability engineering point of view. It includes a detailed state-of-the-art presentation of various spares optimization models and their applications.

The second section describes the importance of optimal cost-effective designs. The detailed formulations of cost-effective designs for repairable and nonrepairable systems are discussed. Various cost factors such as failure cost, downtime cost, spares cost, and maintenance cost are considered. In order to apply these methods for real-life situations, various constraints including acceptable reliability and availability, weight and space limitations, and budget limitations are addressed.

The third section describes the solution techniques and algorithms used for optimal system-design problems. The algorithms are broadly classified as exact algorithms, heuristics, meta-heuristics, approximations, and hybrid methods. The merits and demerits of these algorithms are described. The importance of bounds on the optimal solutions are described.

The fourth section describes the usefulness of hybrid methods in solving large problems in a realistic time frame. A detailed description of the latest research findings relating to hybrid methods and their computational advantages are provided. One of the major advantages of these algorithms is finding the near-optimal solutions as quickly as possible and improving the solution quality iteratively. Further, each iteration improves the search efficiency by reducing the search space as a result of the improved bounds. The efficiency of the proposed method is demonstrated through numerical examples.

54.1 Optimal System Design

In everyday life, we come across various kinds of decision-making problems, ranging from personal decisions related to investment, travel, and career development to business decisions related to procuring equipment, hiring staff, product design, and modifications to existing design and manufacturing procedures. Decision analysis involves the use of a rational process for selecting the best of several alternatives. The solution to decision-making problem requires the identification of three main components.

1. What are the decision alternatives?
   Examples: Should I select vendor X or vendor Y? Should I keep an additional spare component or not?
2. Under what restrictions (constraints) is the decision to be made?

Examples: Do not spend more than $ 10 000 for procuring new equipment. On average, downtime of the system should not exceed two days in a year.

3. What is an appropriate objective criterion for evaluating the alternatives?
   Examples: Overall profit is maximum. Overall availability is maximum. Overall cost is minimum.

Generally, the alternatives of the decision problem may take the form of unknown variables. The variables are then used to construct the restrictions and the objective criterion in appropriate mathematical functions. The end results is a mathematical model relating the variables, constraints, and objective function. The solution of the model then yields the values of the decision variables that optimize (maximize or minimize) the value of the objective function while satisfying all the constraints [54.1]. The resulting solution is referred to as the optimum feasible solution (or simply optimal solution). A typical mathematical model for optimal decision-making is organized as follows:

Maximize or minimize (objective function)

subject to (constraints) .

In the mathematical models for optimization, the decision variables may be integer or continuous, and the objective and constraint functions may be linear and nonlinear. It should be noted that the discrete variables, such as component reliability choices that are restricted to the set {0.6, 0.75, 0.9}, as well as categorical variables such as names can be converted into equivalent integer variables (by using mapping). The optimization problems posed by these models give rise to variety of problem formulations. Each category of problem formulation (or the model) can be solved using a class of solution methods, each designed to account for the special mathematical properties of the model.

1. Linear programming problem: where all objective and constraint functions are linear, and all the variables are continuous.
2. Linear integer programming: is a linear programming problem with the additional restriction that all the variables are integers.
3. Nonlinear programming: where the objective or at least one constraint function is nonlinear, and all the variables are continuous.
4. Nonlinear integer programming: is a nonlinear programming problem where some variables are integers and other variables are continuous.
5. Nonlinear mixed integer programming: is a nonlinear programming problem where some variables are integers and other variables are continuous.

Problems with integer variables and nonlinear functions are difficult to solve. The difficulty further increases if there exist both integer and continuous variables.

54.1.1 System Design

System design is one of the important applications of optimal decision-making problems. One of the important goals of system design is to build the system such that it performs its functions successfully. When a system is unable to perform its functions, this is called a system failure. Several factors related to system design as well as external events influence the system functionality. In most cases, the effects of these factors are random, which means that they cannot be determined precisely but can only be explained through probability distributions. Therefore, failure events or the time to system failure are random variables. The engineering discipline that deals with the successful and unsuccessful (failure) operations of systems is known as reliability engineering. Reliability is one of the important system characteristics and is defined as the probability that the system performs its intended (or specified) functions successfully over a specified period of time under the specified environment. One of the goals of reliability engineering is to illustrate how high reliability, through careful design and analysis, can be built into systems within the limits of economical and physical constraints. Some important principles for enhancing system reliability are [54.2–4]:

1. Keep the system as simple as compatible with performance requirements. This can be achieved by minimizing the number of components in series and their interactions.
2. Increase the reliability of the components in the system. This can be achieved by reducing the variations in the components' strength and applied load (better quality control and controlled operational environment), increasing the strength of the components (better materials), and reducing the applied loads (derating). Alternatively, to some extent, this can be achieved by using large safety factors or management programs for product improvement.
3. Use burn-in procedures for components that have high infant mortality to eliminate early failures in the field.

4. Use redundancy (spares) for less-reliable components; this can be achieved by adding spares in the parallel or standby redundancy.
5. Use fault-tolerant design such that system can continue its functions even in the presence of some failures. This can be achieved using sparing redundancy, fault-masking, and failover capabilities.

In addition to this, if system or its components are repairable, the availability of the system should be considered as a system performance index. The availability of the system is the probability that the system is operational at a specified time. In the long run, the system availability estimate reaches an asymptotic value called the steady-state availability [54.5]. Therefore, in most cases, we focus our attention on improving the steady-state availability of the system. The system availability can be increasing by reducing the downtime. Some important principles for enhancing system reliability are:

1. Use methods that increase the reliability of the system.
2. Decrease the downtime by reducing delays in performing the repair. This can achieved by keeping additional spares on site, providing better training to the repair personnel, using better diagnosis procedures, and increasing the size of the repair crew.
3. Perform preventive maintenance such that components are replaced by new ones whenever they fail, or at some regular time intervals or age, whichever comes first.
4. Perform condition-based maintenance such that downtime related to either preventive or corrective maintenance is minimal.
5. Use better arrangement for exchangeable components.

Implementation of the above steps often consume some resources. The resources to improve system performance (reliability or availability) may be limited. This resources limitation may include available budget, space to keep components, and weight limitations. In such cases, the objective should be to obtain the maximum performance within the utilization of available resources. However, in some cases, achieving high performance may not lead to high profit (or low cost). In such cases, optimal designs should be performed to achieve the most cost-effective solution that strikes a balance between system failure cost and the cost of efforts for reducing the system failures.

54.1.2 System Design Objectives

Depending on the situation, the objective of optimal system design can be one of the following.

1. Maximize system performance.
   There are a number of measures that indicate the performance of a system. For nonrepairable systems, reliability is an important performance measure. For repairable systems, the availability or total uptime is important. When the system has several levels of performance (multi-state systems), the average capacity or throughput is important.
2. Minimize the losses associated with unwanted system behaviors.
   We can also design the system such that the losses associated with downtime can be minimized. Therefore, we can focus on reducing the unreliability, unavailability, downtime, or number of failures.
3. Maximize the overall profit (or minimize the overall cost) associated with the system.

In general, the optimal design corresponding to the maximum system performance (or minimum unwanted behavior) may not exactly coincide with the optimal design that maximizes system profit (or minimum overall cost). In such cases, the objective should be minimizing the overall cost associated with the system that meets the acceptable system performance as well as resource consumption.

In order to solve these optimization problems, we should specify the objectives in a mathematical form. Therefore, we should express the objective functions in a mathematical form.

54.1.3 Notation

m	number of subsystems in the system,
n_i	number of components in subsystem i; $i \in [1, \cdots, m]$,
k_i	minimum number of good components required for successful operation of the subsystem i; $1 \le k_i \le n_i$,
s_i	number of spares in subsystem i; $s_i = n_i - k_i$,
$\boldsymbol{n}$	vector of components; $\boldsymbol{n} = [n_1, \cdots, n_m]$,
$\boldsymbol{n^K}$	vector of components with $n_i = k_i$; $\boldsymbol{n^K} = [k_1, \cdots, k_m]$,
n^*	optimal value of n,
$n^{\mathrm{L}}, n^{\mathrm{U}}$	[lower, upper] bound on the optimal value of n; $n^{\mathrm{L}} \le n^* \le n^{\mathrm{U}}$; n_i^{L} and n_i^{U} are the

	lower and upper bounds on n_i^*; $\boldsymbol{n}^{\mathrm{L}}$ and $\boldsymbol{n}^{\mathrm{U}}$ are the lower and upper bounds on $\boldsymbol{n}^*$,
$\phi_i, \psi_i, \varphi_i$	[mean time to failure = MTTF, mean time to repair = MTTR, mean logistic delay time = MLDT] of a component in subsystem i,
γ_i	fixed miscellaneous cost per each repair of a component in subsystem i,
δ_i	variable cost per unit repair time for a component in subsystem i,
ν_i	frequency of failures for a component in subsystem i,
c_i	maintenance cost per unit time for each component in subsystem i,
c_{f}	cost of system downtime per unit time,
r_i or R_i	reliability of subsystem i,
Q_i	unreliability of subsystem i; $Q_i = 1 - R_i$,
R_{s}	reliability of the system,
Q_{s}	unreliability of the system; $Q_{\mathrm{s}} = 1 - R_{\mathrm{s}}$,
$f(.)$	is a function; $f(r_1, \cdots, r_m)$ is a function in r_i, which is used to represent system reliability in terms of component reliabilities,
$g_i(.)$	is a function; $g_i(r_1, \cdots, r_m)$ is a function in r_i, which is used to represent the constraints in terms of component reliabilities,
r_j^{l}	explicit lower limit on r_j,
r_j^{u}	explicit upper limit on r_j,
$T_{\mathrm{d}}(\boldsymbol{n})$	average downtime cost per unit time with component vector $\boldsymbol{n}$,
$T_{\mathrm{m}}(\boldsymbol{n})$	average maintenance cost per unit time with component vector $\boldsymbol{n}$,
$T(\boldsymbol{n})$	average system cost per unit time with component vector $\boldsymbol{n}$,
C_K	total maintenance cost with $\boldsymbol{n}^{\boldsymbol{K}}$; $C_K = T_{\mathrm{m}}(\boldsymbol{n}^{\boldsymbol{K}}) = \sum_{i=1}^{m} k_i c_i$
C_{U}	total maintenance cost at the upper bound $\boldsymbol{n}^{\mathrm{U}}$; $C_{\mathrm{U}} = T_{\mathrm{m}}(\boldsymbol{n}^{\mathrm{U}}) = \sum_{i=1}^{m} n_i^{\mathrm{U}} c_i$,
C_{L}	total maintenance cost at the lower bound $\boldsymbol{n}^{\mathrm{L}}$; $C_{\mathrm{L}} = T_{\mathrm{m}}(\boldsymbol{n}^{\mathrm{L}}) = \sum_{i=1}^{m} n_i^{\mathrm{L}} c_i$,
p_i, q_i	[availability, unavailability] or [reliability, unreliability] of a component in subsystem i,
A_i, U_i	[availability, unavailability] of subsystem i when there are n_i components in that subsystem; $A_i \equiv A_i(n_i)$; $U_i \equiv U_i(n_i)$,
$A(\boldsymbol{n}), U(\boldsymbol{n})$	[availability, unavailability] of the system with component vector $\boldsymbol{n}$,
h_i^1, h_i^0	Pr{system is operating | subsystem i is [operating, failed]},
binf$(k; p, n)$	cumulative distribution function of binomial distribution; binf$(k; p, n) \equiv \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$; binfc$(k; p, n) \equiv 1 - \mathrm{binf}(k; p, n)$,
gilb$(\cdot)$	greatest integer lower bound; floor$(\cdot)$

54.1.4 System Reliability

If the objective is maximization of system reliability, then we should express the system reliability in a mathematical form. The form of the reliability expression varies with system configuration.

Series Configuration

Series configuration is the simplest and perhaps one of the most common configurations. Let m be the number of subsystems (or components) in the system. In this configuration, all m subsystems (components) must be operating to ensure system operation. In other words, the system fails when any one of the m subsystems fails. Therefore, the reliability of the series system, when all subsystems are independent, is the product of reliabilities of its systems

$$R_{\mathrm{s}} = \prod_{i=1}^{m} R_i \,. \tag{54.1}$$

Parallel Configuration

In the parallel configuration, several paths (subsystems) perform the same operation simultaneously. Therefore, the system fails if all of the m subsystems fail. This is also called an active redundant configuration. The parallel system may occur as a result of the basic system structure or may be produced as a results of additional spares included to improve the system reliability. The parallel system configuration is successful if any one of the m subsystems is successful. In other words, the system fails when all m subsystems fail. Thus the unreliability of the parallel system, when all subsystems are independent, is the product of the unreliabilities of its components

$$Q_{\mathrm{s}} = \prod_{i=1}^{m} Q_i \,,$$

$$R_{\mathrm{s}} = 1 - Q_{\mathrm{s}} = 1 - \prod_{i=1}^{m} Q_i = 1 - \prod_{i=1}^{m} (1 - R_i) \,. \tag{54.2}$$

Standby Configuration

Standby components are used to increase the reliability of the system. There are three types of redundancy [54.6, 7]: (1) cold standby, (2) warm standby, and (3) hot

standby. In the cold-standby redundancy configuration, a component (or subsystem) does not fail before it is put into operation. In warm-standby redundancy configuration, the component can fail in standby mode. However, the chances of failure (failure rate in standby) are less than the chances of failure in operation (failure rate in operation). If the failure pattern of a standby component does not depend on whether the component is idle or in operation, then it is called hot standby. In this chapter, the analysis is provided for cold- and hot-standby components only.

If the redundant components operate simultaneously from time zero, even though the system needs only one of them at any time, such an arrangement is called parallel (or active) redundancy. This arrangement is essential when switching or starting a good component following a component failure is ruled out. The mathematical models for hot-standby and parallel redundancy arrangements are equivalent. In the cold-standby redundancy configuration, the redundant components are sequentially used in the system at component failure times. The system fails when all components fails. Cold-standby redundancy provides longer system life than hot-standby redundancy.

For a cold-standby system with a total of m components where initially the first component is in operation and the rest of the $m-1$ components are in standby, the system reliability at time t is

$$\begin{aligned} &R_\mathrm{s}(t) \\ &= \Pr(\text{system operates successfully until time } t) \\ &= \Pr(X_1 + X_2 + \cdots + X_\mathrm{m} \geq t)\,, \end{aligned} \tag{54.3}$$

where X_i is the random variable that represents the failure time of component i. The final expression for the system reliability is a function of the parameters of the component failure-time distribution. If all components are identical and follow exponential failure-time distributions with hazard rate λ, then the reliability of the cold-standby system with m components is [54.7, 8]

$$\begin{aligned} R_\mathrm{s}(t) &= \exp(-\lambda t)\left(\sum_{i=0}^{m-1}\frac{(\lambda t)^i}{i!}\right) \\ &= p\left(\sum_{i=0}^{m-1}\frac{[-\ln(p)]^i}{i!}\right)\,, \end{aligned} \tag{54.4}$$

where $p = \exp(-\lambda t)$ is the reliability of each component.

k-out-of-n Active Standby Configuration

In this configuration, the system consists of n active components in parallel. The system functions successfully when at least k of its n components function. This is also referred to as a k-out-of-n:G parallel (or hot or active standby) system or simply a k-out-of-n system. A parallel system is a special case of a k-out-of-n system with $k = 1$, i. e., a parallel system is a 1-out-of-n system. Similarly, a series system is a special case of a k-out-of-n system with $n = 1$, i. e., a series system is an n-out-of-n system.

When all components (or subsystems) are independent and identical, the reliability of a k-out-of-n system with component reliability equal to p is [54.3, 8, 9]:

$$\begin{aligned} R_\mathrm{s} &= \sum_{i=k}^{n}\binom{n}{i}p^i(1-p)^{n-i} \\ &= 1 - \sum_{i=0}^{k-1}\binom{n}{i}p^i(1-p)^{n-i} \\ &= \mathrm{binfc}(k-1;\, p, n)\,. \end{aligned} \tag{54.5}$$

k-out-of-n Cold-Standby Configuration

In this configuration, the system consists of a total of n components. Initially, only k components are in operation and the remaining (n-k) redundant components are kept in cold standby. The redundant components are sequentially used to replace the failed components. When all components (or subsystems) are identical and the failure distribution is exponential, the reliability of a k-out-of-n cold-standby system is [54.8]:

$$\begin{aligned} R_\mathrm{s}(t) &= \exp(-k\lambda t)\left(\sum_{i=0}^{k-1}\frac{(k\lambda t)^i}{i!}\right) \\ &= p^k\left(\sum_{i=0}^{k-1}\frac{[-k\ln(p)]^i}{i!}\right)\,. \end{aligned} \tag{54.6}$$

General System Configuration

It is well known that the reliability of a system is a function of the component reliabilities or its parameters. Hence, we have [54.8]

$$R_\mathrm{s} = h(R_1, \cdots, R_\mathrm{m})\,. \tag{54.7}$$

Here, h is the function that represents the system reliability in terms of the component reliabilities. The actual formula for $h(R_1, \cdots, R_\mathrm{m})$ depends on the system configuration. Several algorithms are available [54.7] to compute (54.7). Recent literature [54.10] has shown

that, in many cases, algorithms based on binary decision diagrams are more efficient in computing (54.7).

54.1.5 System Availability

If the objective is to maximize the system availability, then we should express the system availability in a mathematical form. The form of the availability expression varies with system configuration. In this section, we provide the availability expressions for some specific configurations. These expressions are valid under the following assumptions.

1. The failure- and repair-time distributions of all components are independent.
2. There are sufficient repair resources such that repair of a failed component starts immediately.

Series Configuration

The availability of the series system is [54.7]

$$A_s = \prod_{i=1}^{m} A_i \,, \tag{54.8}$$

where A_i is the availability of component (subsystem) i and m is the number of components.

Parallel Configuration

The availability of the parallel system is [54.7]:

$$A_s = 1 - \prod_{i=1}^{m} A_i = 1 - \prod_{i=1}^{m} (1 - A_i) \,, \tag{54.9}$$

where A_i is the availability of component (subsystem) i and m is the number of components.

k-out-of-n Active-Standby Configuration

The availability of a k-out-of-n system with identical components is [54.7, 9]

$$\begin{aligned} A_s &= \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i} \\ &= \text{binfc}(k-1;\, p, n) \,, \end{aligned} \tag{54.10}$$

where p represents the component availability.

General System Configuration

When failure and repair processes of all components or subsystems are independent, then we can represent the system availability as a function of component availabilities. Hence, we have

$$R_s = h(A_1, \cdots, A_m). \tag{54.11}$$

Here, h is the function that represents the system availability in terms of component availabilities. The actual formula for $h(A_1, \cdots, A_m)$ depends on the system configuration. In this cases, the same algorithms used for system reliability can be used for computing system availability.

54.1.6 Other Objective Functions

Similarly, we can also find the other objective functions analytically [54.11–17]. When closed-form expression are not available the objective function can be calculated using either numerical methods or simulation [54.7]. In such cases, the methods for finding the optimal solutions should not depend on the form of the objective function. The generic nature of these solution methods may lose the advantages associated with the specific form of the objective function. In this section, we provide the expressions for some of the objective functions.

Unreliability

In some systems, the objective would be minimization of unreliability. The unreliability of a system can be obtained from its reliability [54.7]

$$\begin{aligned} &\text{Unreliability} = 1 - \text{Reliability}, \\ &Q_s = 1 - R_s \,. \end{aligned} \tag{54.12}$$

Unavailability

In some systems, the objective would be minimization of unavailability. We have [54.7]

$$\begin{aligned} &\text{Unavailability} = 1 - \text{Availability}, \\ &U_s = 1 - A_s \,. \end{aligned} \tag{54.13}$$

Total Uptime

In some systems, the objective would be the maximization of total uptime (or operational time) over a period of time. We have [54.7]

$$TUT_s(t) = \int_0^t A_s(x)\,\mathrm{d}x \,. \tag{54.14}$$

Total Downtime

In some systems, the objective would be minimization of total downtime during a specified period of time. We have [54.7]

$$TDT_s(t) = \int_0^t U_s(x)\,\mathrm{d}x = t - TUT_s(t) \,. \tag{54.15}$$

Mean Availability

In some systems, the objective would be maximization of system mean availability over a period of time. We have [54.7]

$$A_{\mathrm{M}}(t) = \frac{TUT_{\mathrm{s}}(t)}{t} \,. \tag{54.16}$$

Mean Unavailability

In some systems, the objective would be minimization of mean unavailability over a period of time. We have [54.7]

$$U_{\mathrm{M}}(t) = \frac{TDT_{\mathrm{s}}(t)}{t} = 1 - A_{\mathrm{M}}(t) \,. \tag{54.17}$$

Average cost

In the majority of applications, the objective of system design is to minimize the overall cost associated with the system. The total cost is the sum of several cost factors. These include:

1. System failure cost,
2. Cost of components and spares,
3. Cost of maintenance (repair, replacement, and inspection costs),
4. Warranty costs,
5. Cost associated with downtime (or loss of production).

The actual formulas and cost factors vary with the applications. For details, see [54.11–15, 17].

Decision Variables

These are the values that we are interested to find such that the specified objective is minimized or maximized [54.1]. The decision variables include:

1. Type of system configuration,
2. Types of components or spares,
3. Number of spares in a specific application (or subsystem),
4. Number of repair personnel.

The type of a component is applicable if there are several alternative components for the same application with various costs, weights, volumes, and failure rates (or reliabilities).

Constraints

The optimal solutions should be obtained within the allowed resource restrictions. These are also called constraints of the optimization problem. The constraints include:

1. Desired reliability,
2. Desired availability,
3. Desired mean time to failure (MTTF) or mean time between failures (MTBF),
4. Allowed downtime,
5. Allowed unavailability,
6. Allowed budget (for spares or repair resources),
7. Allowed weight,
8. Available space (or volume).

54.1.7 Existing Optimization Models

As described in the previous sections, there are several possibilities for objective functions, constraints, and decision variables. Furthermore, the diversity of the system configurations and their special properties lead to several optimization models. In this section, we present some well-studied optimization models and associated mathematical formulations. A detailed treatment of these problems is presented in [54.6].

In all these models, it is assumed that the system consists of several stages (subsystems or modules). In most cases, the objective is to maximize the system reliability by optimally assigning the component reliabilities and/or redundancy levels at various stages, subject to resources constraints.

Allocation of Continuous Component Reliabilities

In this formulation, the system reliability can be improved through the selection of component reliabilities at stages subject to resource constraints. Therefore, the decision variables are the reliabilities of the stages $(r_1, \cdots, r_m)$. The problem of maximizing system reliability through the selection of component reliabilities subject to resource constraints can be expressed as:

$$\begin{aligned} &\text{Maximize } R_{\mathrm{s}} = f(r_1, \cdots, r_m) \\ &\quad \text{subject to:} \\ &g_i(r_1, \cdots, r_m) \le b_i \qquad \text{for } i = 1, \cdots, k \,, \\ &r_j^{\mathrm{l}} \le r_j \le r_j^{\mathrm{u}} \qquad \text{for } j = 1, \cdots, m \,. \end{aligned} \tag{54.18}$$

This is a nonlinear programming problem.

Allocation of Discrete and Continuous Component Reliabilities

In this formulation, we have u_j discrete choices for component reliability at stage j for $j = 1, \cdots, s$ and the choice for the component reliability at stages $s+1, \cdots, m$ is on a continuous scale. Therefore, the decision variables are the reliabilities of stages $(r_1, \cdots, r_m)$.

Because we have discrete choices for the reliability at stage j for $j = 1, \cdots, s$, selecting the reliability at each stage is equivalent to selecting the related choice, which can be expressed as an integer. Hence, the decision variables are: $(x_1, \cdots, x_s, R_{s+1}, \cdots, R_m)$. Alternatively, they are equivalent to specifying $[R_1(x_1), \cdots, R_s(x_s), R_{s+1}, \cdots, R_m]$. The problem of maximizing system reliability through the selection of component reliabilities subject to resource constraints can be expressed as:

$$\begin{aligned}
&\text{Maximize} \\
&R_s = f(R_1(x_1), \cdots, R_s(x_s), R_{s+1}, \cdots, R_m) \\
&\text{subject to:} \\
&g_i[R_1(x_1), \cdots, R_s(x_s), R_{s+1}, \cdots, R_m] \le b_i \\
&\qquad\qquad \text{for } i = 1, \cdots, k\,, \\
&x_j \in \{1, 2, \cdots, u_j\} \qquad \text{for } j = 1, \cdots, s\,, \\
&r_j^{\mathrm{l}} \le r_j \le r_j^{\mathrm{u}} \qquad \text{for } j = 1, \cdots, m\,.
\end{aligned} \tag{54.19}$$

This problem is called the reliability allocation problem. This problem can be simplified when the separability assumption is applicable. With the separability assumption, we have

$$g_i(R_1, \cdots, R_m) = \sum_{j=1}^{m} g_{ij}(r_i)\,. \tag{54.20}$$

This problem is a nonlinear mixed integer programming problem.

Redundancy Allocation

System reliability can be improved through the selection of redundancy levels at stages, subject to resource constraints. Therefore, the decision variables are the redundancy levels of stages $(x_1, \cdots, x_m)$. Alternatively, they are equivalent to specifying $[R_1(x_1), \cdots, R_m(x_m)]$. The problem of maximizing the system reliability through the selection of optimal redundancy levels, $x_1, \cdots, x_n$, subject to resource constraints can be expressed as:

$$\begin{aligned}
&\text{Maximize } R_s = f(x_1, \cdots, x_n) \\
&\text{subject to:} \\
&g_i(x_1, \cdots, x_n) \le b_i \qquad \text{for } i = 1, \cdots, k\,, \\
&l_j \le x_j \le u_j \qquad \text{for } j = 1, \cdots, m\,, \\
&x_j \qquad \text{is an integer}\,.
\end{aligned} \tag{54.21}$$

This problem is called the redundancy allocation problem. It is a nonlinear integer programming problem. As in other cases, this problem can be simplified with the separability assumption, which is often applicable in real life. With the separability assumption, we have

$$g_i(x_1, \cdots, x_m) = \sum_{j=1}^{m} g_{ij}(x_i)\,. \tag{54.22}$$

This problem is thoroughly discussed in the literature [54.3, 4, 6, 18–21].

Reliability–Redundancy Allocation

In this formulation, the system reliability can be improved through the selection of component reliabilities as well as redundancy levels at stages, subject to resource constraints. Therefore, the decision variables are the pair of values containing the reliability and level of redundancy for each stage $s(r_s, x_s)$. Hence, the decision variables are both $(x_1, \cdots, x_n)$ and $(r_1, \cdots, r_n)$. The problem of finding simultaneously the optimal redundancy levels $(x_1, \cdots, x_n)$ and the optimal component reliabilities $(r_1, \cdots, r_n)$ that maximize system reliability subject to the resource constraints can be expressed as:

$$\begin{aligned}
&\text{Maximize} \\
&R_s = f(x_1, \cdots, x_n; r_1, \cdots, r_n) \\
&\text{subject to:} \\
&g_i(x_1, \cdots, x_n; r_1, \cdots, r_n) \le b_i \quad \text{for } i = 1, \cdots, k\,, \\
&l_j \le x_j \le u_j \qquad \text{for } j = 1, \cdots, m\,, \\
&r_j^{\mathrm{l}} \le r_j \le r_j^{\mathrm{u}} \qquad \text{for } j = 1, \cdots, m\,, \\
&x_j \qquad \text{is an integer}\,.
\end{aligned} \tag{54.23}$$

This problem is called the reliability-redundancy allocation problem. It is a nonlinear mixed integer programming problem. This problem can also be simplified when the separability assumption is applicable. With the separability assumption, we have:

$$g_i(x_1, \cdots, x_m; r_1, \cdots, r_n) = \sum_{j=1}^{m} g_{ij}(x_i, r_i)\,. \tag{54.24}$$

Redundancy Allocation for Cost Minimization

In this formulation, the objective is cost minimization. In traditional models, the overall cost of the system is expressed as the sum of the cost of all components (stages). The cost of each stage is a function of the redundancy level of the stage. Therefore, the decision variables are the redundancy levels at states $(x_1, \cdots, x_n)$. The problem of finding simultaneously optimal the redundancy

levels $(x_1, \cdots, x_n)$ that minimize the system cost subject to resource constraints can be expressed as:

$$\begin{aligned} \text{Minimize } C_s &= \sum_{j=1}^{m} C_j(x_j) \\ \text{subject to:} & \\ g_i(x_1, \cdots, x_n) &\le b_i \quad \text{for } i = 1, \cdots, k\,, \\ l_j \le x_j &\le u_j \quad \text{for } j = 1, \cdots, m\,, \\ x_j &\quad \text{is an integer}\,. \end{aligned} \tag{54.25}$$

Here, $c_j(x_j)$ is the cost of x_j components at stage j.

Cost-minimization problems are less studied in the literature. Furthermore, there is a lot of scope to improve this formulation by incorporating various cost factors associated with system.

Other Formulations

Other formulations include:

1. Allocation of discrete component reliabilities and redundancies. In this formulation, depending on the type of stage, we can select either a discrete component reliability or the redundancy level (which is also discrete). Therefore, it is a nonlinear integer programming problem. A generalization to this formulation could be by allowing a combination of discrete and continuous choices for component reliability along with choosing the redundancies levels at each stage. Hence, this generalization problem becomes a mixed integer nonlinear programming problem.
2. Component assignment problem. This is applicable when components at various stages can be interchangeable
3. Multi-objective optimization problem. This is applicable when there are multiple simultaneous objectives such as maximization of reliability and minimization of cost, volume, weight, etc.

For more details on these formulations, see [54.6].

54.2 Cost-Effective Designs

In most of the problems studied in the literature, the objective is to maximize the system reliability. However, as we discussed earlier, the optimal solution that maximizes reliability may not necessarily minimizes the overall cost associated with the system. Even though some formulations allow the specification of cost factors in the constraints, they do not minimize the cost. Instead they provide optimal solutions within the specified cost constraints, such as an allowed budget. Further, the well-known cost-minimization problem minimizes the total cost of components subject to other constraints such as desired reliability and allowed volume, weight, etc. However, the cost of the system not only includes the cost of components but also includes various other factors, which are discussed in Sect. 54.1.6. The optimal solution should strike a balance between various competing cost factors such that the overall cost of the system is minimized and at the same time all constraints are satisfied. The cost factors varies with the system type, failure mode, and other details related to a specific system.

54.2.1 Nonrepairable Systems

General Cost Structure

For nonrepairable systems, the major cost factors are the cost of spares and the cost of failure. The system failure cost can be minimized by using reliable system designs. System reliability can be improved by using the redundancy of spares, which can be kept in hot- or cold-standby mode (or in some cases in warm-standby mode). Although the reliability of a system increases with the addition of spares, the cost of the system also increases due to the number of redundant components added to the system. Thus, it is desirable to derive a cost-effective solution that strikes a balance between the system failure cost and the cost of spares in the system. Therefore, the objective is to minimize the average cost associated with the system

$$\begin{aligned} &\text{Average cost} \\ &= \text{Cost of minimum required components} \\ &\quad + \text{Cost of spares} + \text{Cost of failure}\,. \end{aligned} \tag{54.26}$$

Because the cost of the minimum required components is a fixed initial cost, this cost can be eliminated. Alternatively, minimizing the cost using (54.26) and (54.27) are mathematically the same and produce the same optimal solution for the spares. Therefore, depending on convenience, we can use one of these formulations

$$\text{Average cost} = \text{Cost of spares} + \text{Cost of failure}\,. \tag{54.27}$$

The cost of failure can be modeled in several ways. There can be a cost (fixed or randomly distributed with a fi-

nite mean) for not completing the mission successfully. Therefore,

$$\begin{aligned}&\text{Average cost of failure}\\&= \text{Cost of failure of a mission} \times \text{Unreliability}\,.\end{aligned} \tag{54.28}$$

For example, this kind of cost is applicable for missions such as landing on a planet or a moon. Finally, the average system cost is

$$\begin{aligned}\text{Average cost} = {}&\text{Cost of spares}\\&+\text{Cost of failure of a mission}\\&\times\text{Unreliability}\,.\end{aligned} \tag{54.29}$$

In some scenarios the cost of system failure may depend on the time at which the failure occurs. If the system fails at the beginning of the mission, the losses are high compared to if the system fails almost at the end of the mission. If the cost is linearly proportional to the remaining mission time, then we can express the average system cost in terms of the average remaining mission time. Therefore,

$$\begin{aligned}&\text{Average cost of failure}\\&= \text{Cost of failure of a mission per unit time}\\&\quad\times\text{Remaining mission}\\&= \text{Cost of failure of a mission per unit time}\\&\quad\times\text{Mission duration}\\&\quad\times\text{Average unreliability or unavailability}\\&= \text{Cost of failure of a mission per unit time}\\&\quad\times\text{Mean downtime}\,.\end{aligned} \tag{54.30}$$

In most cases, the problem of minimizing the average cost of the system can be converted into the problem of maximizing the average profit of the system. When there is a fixed profit for each successful operation of the mission, we can easily show that these two problems are identical.

$$\begin{aligned}&\text{Average system profit}\\&= \text{Profit during success} \times \text{Reliability}\\&\quad-\text{Losses during failure}\times\text{Unreliability}\\&\quad-\text{Cost of components}\\&= \text{Profit during success}\\&\quad-(\text{Profit during success}-\text{Losses during failure})\\&\quad\times\text{Unreliability}\\&\quad-\text{Cost of components}\,.\end{aligned} \tag{54.31}$$

Because the *profit during success* is fixed, it is a constant. Therefore, the problem of maximizing the system profit is reduced to maximization of Function1 shown in (54.32):

$$\begin{aligned}&\text{Function1}\\&= -(\text{Profit during success}-\text{Losses during failure})\\&\quad\times\text{Unreliability}\\&\quad-\text{Cost of components}\,.\end{aligned} \tag{54.32}$$

Because maximizing $[f(x)]$ is equivalent to minimizing $[-f(x)]$, the problem is equivalent to minimizing the Function2 in (54.33). For details, see [54.1].

$$\begin{aligned}&\text{Function2}\\&= (\text{Profit when success}-\text{Losses during failure})\\&\quad\times\text{Unreliability}\\&\quad+\text{Cost of components}\,.\end{aligned} \tag{54.33}$$

The objective functions shown in (54.29) and (54.33) have the same form. Therefore, the same methods that are used for cost-minimization problems can also be used for profit-maximization problems.

A specific Model for Nonrepairable Systems

In this section, we provide a detailed model for a specific class of nonrepairable systems. The system consists of several subsystems, which are arranged in a network configuration. Each subsystem requires a minimum number of identical components to perform its functions. The additional spare components in each subsystem can be kept online or in cold- or hot-standby mode. Therefore, each subsystem behaves like a k-out-of-n:G cold/hot-standby system. The objective of the problem considered in this chapter is to find the optimal cost-effective design of the overall system. This means that the optimal number of components in each subsystem must be found to minimize the overall cost associated with the system.

Assumptions.

1. The system consists of m subsystems. The subsystems can be arranged in a complex network configuration [54.22].
2. System structure function is coherent with respect to the subsystems.
3. All subsystems are statistically independent.
4. Each subsystem consists of $n_i \equiv k_i + s_i$ identical components.
5. Subsystem i requires k_i components for its successful operation.

6. Subsystem i can have s_i spares. Spares of a subsystem can be kept in hot- or cold-standby mode.
7. The failure rate of an operational component is constant. The failure rate of a hot-standby component is equivalent to the failure rate of an operational component. The failure rate of a cold-standby component is zero.

Problem Formulation. Because the subsystems are independent, we can compute the reliability of each subsystem separately and use those results to compute the overall system reliability. The reliability calculation of a subsystem depends on the type of spares used.

1. Hot standby

$$R_i = \text{binfc}(k_i - 1;\, p_i, n_i)\,. \tag{54.34}$$

2. Cold standby

$$\begin{aligned} R_i &= \exp(-k_i\lambda_i t)\left(\sum_{j=0}^{s_i}\frac{(k_i\lambda_i t)^j}{j!}\right) \\ &= p_i^{k_i}\left(\sum_{j=0}^{s_i}\frac{[-k_i\ln(p_i)]^j}{j!}\right)\,. \end{aligned} \tag{54.35}$$

System reliability is a function of the subsystem reliabilities

$$\begin{aligned} R(\boldsymbol{n}) &= h(R_1, \cdots, R_m) \\ &= R_i h_i^1 + (1 - R_i)h_i^0 \\ &= \left(h_i^1 - h_i^0\right)R_i + h_i^0\,. \end{aligned} \tag{54.36}$$

The average cost of the system, $T_s(n)$, is the cost incurred when the system has failed, plus the cost of all components in the system

$$T(\boldsymbol{n}) = \sum_{i=1}^{m} c_i n_i + c_f\left[1 - R(\boldsymbol{n})\right]\,. \tag{54.37}$$

The objective is to find the optimal $\boldsymbol{n}$ that minimizes $T(\boldsymbol{n})$. The problem can be further refined by considering minimum acceptable reliability (R_a), acceptable upper limit on total weight & volume, and allowable budget for spares acquisition. Therefore, the constraints are

$$\begin{aligned} \text{Budget constraint:} &\quad \sum_{i=1}^{m} s_i.c_i \le B \\ \text{Volume constraint:} &\quad \sum_{i=1}^{m} v_i \le V \\ \text{Weight constraint:} &\quad \sum_{i=1}^{m} w_i \le W \\ \text{Reliability constraint:} &\quad R(\boldsymbol{n}) \ge R_a\,, \end{aligned} \tag{54.38}$$

where B is the allowed budget, V is the maximum allowed volume, W is the maximum allowed weight, and R_a is the minimum acceptable reliability.

In addition, we can also add explicit constraints on the number of components in each subsystem. However, in general, the problem is difficult to solve when no explicit constraints on the decision variables are specified. The solution can be simplified by finding the explicit bounds for each decision variable, i. e., the number of components. It is a nonlinear integer programming problem.

54.2.2 Repairable Systems

General Cost Structure

For repairable systems, the major cost factors are:

- initial cost of spares,
- cost of failures,
- cost of repair and replacement,
- cost of storage.

However, in the long run, the cost of the initial cost of spares is negligible compared to the other costs. The cost of failure can be minimized by minimizing the system unreliability, which in turn can be achieved by increasing the number of spares in the system. However, the increase in the number of spares can also increase the maintenance and operational costs of spares, which include repair, replacement and storage costs. Thus, it is desirable to derive a cost-effective solution that strikes a balance between the system failure cost and the cost of maintenance and operation

$$\begin{aligned} &\text{Average system cost} \\ &= \text{Cost of maintenance} \\ &\quad + \text{Cost of failure} \times \text{Unavailability}\,. \end{aligned} \tag{54.39}$$

However, for short-duration systems, we have

$$\begin{aligned}&\text{Average system cost}\\&= \text{Initial cost of components and spares}\\&\quad + \text{Cost of set up}\\&\quad + \text{Cost of maintenance}\\&\quad + \text{Cost of failure} \times \text{Unavailability}\\&\quad + \text{Cost of disposal}\,.\end{aligned} \tag{54.40}$$

The cost of disposal can be positive or negative (for resale value it is negative). When we can calculate the cost of each component including the cost of procurement, set up, and resale values. The problem can be reduced to

$$\begin{aligned}&\text{Average system cost}\\&= \text{Effective cost of components and spares}\\&\quad + \text{Cost of maintenance}\\&\quad + \text{Cost of failure} \times \text{Unavailability}\,.\end{aligned} \tag{54.41}$$

A Specific Model for Repairable Systems

The majority of engineering and manufacturing systems consist of several subsystems [54.22], which are usually nonidentical. In general, all subsystems need not be in series, but can be arranged in any configuration. The success logic of these types of systems can be represented using networks or reliability block diagrams, which are collectively known as combinatorial reliability models [54.23]. Each subsystem can have one or more functionally similar components [54.24]. For successful operation of a subsystem, there must be at least a specified number of components in operation. Such subsystems, known as k-out-of-n subsystems [54.9], have a wide range of applications [54.9, 12, 13, 25]. A special case of a k-out-of-n subsystem with $k = 1$ is called a parallel subsystem [54.26].

In general, systems and components are repairable. Therefore, systems and components undergo several failure–repair cycles that include logistic delays while performing repairs [54.27]. In the long run, the downtime costs are directly related to the asymptotic unavailability of the system. System unavailability can be reduced by increasing the availability of its subsystems, which in turn can be increased by additional spares for each subsystem. Although the availability of a system increases with the addition of spares, the cost of the system also increases due to the added operational and maintenance costs. Thus, it is desirable to derive a cost-effective solution that strikes a balance between the system downtime cost and the operational and maintenance costs of spares in the system. The objective of the cost-effective solution is to find the optimal number of spares in each subsystem that minimizes the overall system cost.

Assumptions.

1. The system consists of m subsystems. All subsystems are statistically independent.
2. Subsystem i consists of $n_i \equiv k_i + s_i$ components, where subsystem i requires at least k_i components for its successful operation.
3. Failure, repair, and logistic delay times of all components are independent and can follow any distribution.
4. System failure cost is proportional to the downtime.
5. There is a fixed miscellaneous cost for each repair of a component. In addition to the fixed cost, the cost of repair has a variable cost, which is proportional to the amount of repair time. The downtime of a component is the sum of the repair time and logistic delay time.

Problem Formulation. The average system cost is the sum of the average cost of downtime plus the average cost of maintenance

$$T(\boldsymbol{n}) = T_{\mathrm{d}}(\boldsymbol{n}) + T_{\mathrm{m}}(\boldsymbol{n})\,. \tag{54.42}$$

The cost of downtime can be calculated from the percentage of downtime within a unit time duration and the loss (cost) per unit downtime. As time progresses, the system reaches a steady-state condition. Under steady-state conditions, the percentage of downtime is equivalent to the steady-state unavailability. Hence,

$$T_{\mathrm{d}}(\boldsymbol{n}) = c_{\mathrm{f}} U(\boldsymbol{n}) = c_{\mathrm{f}}\left[1 - A(\boldsymbol{n})\right]\,. \tag{54.43}$$

System availability is a function of the subsystem availabilities

$$A(\boldsymbol{n}) = h(A_1, \cdots, A_{\mathrm{m}})\,. \tag{54.44}$$

The actual formula for $h(A_1, \cdots, A_m)$ depends on the system configuration. For example, if the system configuration is series, then

$$A(\boldsymbol{n}) = \prod_{i=1}^{m} A_i(n_i)\,. \tag{54.45}$$

Several algorithms are available [54.7] to compute (54.44). Recent literature [54.10] has shown that, in many cases, algorithms based on binary decision diagram are more efficient in computing (54.44).

If the system is under steady-state conditions, and the failure and repair processes of all components are

independent, the availability of each component can be calculated using its mean time to failure (MTTF), mean time to repair (MTTR), and mean logistic delay time (MLDT). Because each subsystem consists of identical components and the configuration of each subsystem is k_i-out-of-n_i, we have:

$$\begin{aligned} A_i = A_i(n_i) &= \text{binf}(n_i - k_i; q_i, n_i) \\ &= \text{binfc}(k_i - 1; p_i, n_i) \end{aligned} \tag{54.46}$$

where p_i and q_i are, respectively, the operational availability and unavailability of a component in subsystem i. Furthermore, given that $q_i = 1 - p_i = (\psi_i + \varphi_i)/(\phi_i + \psi_i + \varphi_i)$, where ϕ_i is the MTTF, ψ_i is the MTTR, and φ_i is the MLDT of a component in subsystem i.

The cost of maintenance is proportional to the cost associated with the repairs of individual components. The cost of repair of a failed component includes the miscellaneous fixed cost as well as the variable cost based on the repair time. Therefore, if the repair of a failed component in the subsystem i takes on average ψ_i units of time, then the average cost of repair for each instance of repair is

$$R_i = \gamma_i + \psi_i \delta_i \,. \tag{54.47}$$

Let ν_i be the failure frequency, i. e., expected number of failures per unit time, of a component in subsystem i. Because all components in a subsystem are identical, on average, we have $N_i \equiv n_i f_i$ failures per unit time. Under steady-state conditions, we have

$$\begin{aligned} \nu_i &= \frac{1}{\phi_i + \psi_i + \varphi_i} \\ N_i &= \frac{n_i}{\phi_i + \psi_i + \varphi_i} \,. \end{aligned} \tag{54.48}$$

Hence, the cost of maintenance of a subsystem per unit time is $\theta_i(n_i)$

$$\theta_i(n_i) = N_i R_i = n_i \nu_i R_i \,. \tag{54.49}$$

The cost of maintenance of the entire system is

$$T_{\text{m}}(\boldsymbol{n}) = \sum_{i=1}^{m} \theta_i(n_i) = \sum_{i=1}^{m} n_i \nu_i R_i \,. \tag{54.50}$$

It should be noted that, in the long run, the initial cost of the spares per unit time is negligible and need not be considered.

Therefore, the average cost of the system is

$$\begin{aligned} T(\boldsymbol{n}) &= \sum_{i=1}^{m} c_i n_i + c_{\text{f}} U(\boldsymbol{n}) \,, \\ c_i &= \frac{\gamma_i + \psi_i \delta_i}{\phi_i + \psi_i + \varphi_i} \,, \\ U(\boldsymbol{n}) &= 1 - A(\boldsymbol{n}) \,, \\ A(\boldsymbol{n}) &= h(A_1, \cdots, A_m) \,, \\ A_i &\equiv A_i(n_i) = \text{binf}(n_i - k_i; q_i, n_i) \,, \\ q_i &= \frac{\psi_i + \varphi_i}{\phi_i + \psi_i + \varphi_i} = 1 - \frac{\phi_i}{\phi_i + \psi_i + \varphi_i} \,. \end{aligned} \tag{54.51}$$

The objective is to find the optimal $\boldsymbol{n}$ that minimizes $T(\boldsymbol{n})$. The problem can be further refined by considering the maximum acceptable unavailability (U_{a}) and the acceptable upper limit on total weight and volume. Therefore, the constraints are

$$\begin{aligned} \text{Volume constraint:} \quad & \sum_{i=1}^{m} v_i \le V \,; \\ \text{Weight constraint:} \quad & \sum_{i=1}^{m} w_i \le W \,; \\ \text{Unavailability constraint:} \quad & U(\boldsymbol{n}) \le U_{\text{a}} \,. \end{aligned} \tag{54.52}$$

Similarly, we can also add explicit constraints on the number of components in each subsystem. This is a nonlinear integer programming problem. More specific forms of average cost functions are studied in [54.11–15, 17].

54.3 Optimal Design Algorithms

54.3.1 An Overview

In this section, we discuss various methods for solving optimal system problems. We demonstrate these methods for solving the problems associated with cost-effective designs. The same algorithms can also be applied for the other problem formulations. The algorithms can be broadly classified as follows:

1. *Exact methods*. These methods produce exact optimal solutions. It is generally difficult to develop efficient exact methods for spares optimization problems. This is because the objective function, which is the function to be minimized, is nonlinear and involves several integer variables, which are the components to be optimized. In addition, the objective function may have several peaks and may

not possess monotonic increasing or decreasing characteristics. Therefore, exact methods involve more computational effort and usually require large amounts of computer memory.

One exact method is exhaustive searching, where the objective function is computed for all possible combinations of the decision variables, which would be the quantities of spares. However, exhaustive searching is infeasible even for a moderately size problem for optimizing the number of spares. Other algorithms in this category involve dynamic programming, branch-and-bound methods, and nonlinear mixed integer programming techniques. Some researchers have developed exact methods that have closed-form or computationally efficient solutions for some well-structured systems such as k-out-of-n systems, parallel–series, and series–parallel systems.

2. *Approximation methods*. Some of the difficulties of spares optimization problem are due to the presence of an integer variable (the number of spares), the variable forms of cost functions (the number of terms in the cost function is based on the number of spares), and non-polynomial cost functions. In addition, the cost function may have several peaks and valleys. By using approximation methods, these complex problems can be modeled in a simpler form by considering continuous variables for spares quantities and approximate forms for the cost function. This approach produces near-optimal solutions, or solutions that are very close to the exact solution result. However, approximation methods cannot guarantee the global optimal solutions, which are exact solutions that actually minimize the objective functions.
3. *Heuristic methods*. Finding exact optimal solutions for large complex systems is not always feasible because such problems involve resource-intensive computation efforts and usually require large amounts of computer memory. For these reasons, researchers in reliability optimization have placed more emphasis on heuristic approaches, which are methods based on rules of thumb or guidelines that generally find the solutions but do not guarantee an exact solution. In most of these approaches, the solution is improved in each iteration. One of the simple heuristic approaches is the greedy method, where the quantity of spares is incremented for the subsystem where the maximum reduction in cost is achieved. This iterative process is stopped when a point is reached where adding spares to any component increases the cost. Heuristic methods typically produce local optimal solutions within a short time. However, they may not guarantee the global optimal solutions and may produce local optimal solutions, which is an optimal solution in a local neighborhood. For spares optimization, there may exist several local optimal solutions, where changing any variable slightly (increasing or decreasing any spare) increases the total cost. The global optimal solution corresponds to the minimal solution out of all such local optimal solutions.
4. *Meta-heuristic methods*. These methods are based more on artificial reasoning than classical mathematics-based optimization. They include genetic algorithms (GA) and simulated annealing (SA). GAs seek to imitate the biological phenomenon of evolutionary production through parent–child relationships. SA is based on a physical process in metallurgy. Most of these methods use stochastic searching mechanisms. Meta-heuristic methods can overcome the local optimal solutions and, in most cases, they produce efficient results. However, they also cannot guarantee the global optimal solutions.
5. *Hybrid methods*. These methods use combinations of different optimization techniques. An example of a hybrid method would be to find the initial

Table 54.1 Exhaustive search results

n_1	n_2	R_s	C_s	Remarks
1	1	0.600 000	–	C5 violated
1	2	0.720 000	–	C5 violated
1	3	0.744 000	–	C5 violated
1	4	0.748 800	–	C5 violated
2	1	0.750 000	–	C5 violated
2	2	0.900 000	106.000 000	
2	3	0.930 000	78.000 000	
2	4	0.936 000	74.000 000	
3	1	0.787 500	–	C5 violated
3	2	0.945 000	62.000 000	
3	3	0.976 500	32.500 000	
3	4	0.982 800	28.200 000	
4	1	0.796 875	–	C5 violated
4	2	0.956 250	51.750 000	
4	3	0.988 125	21.875 000	
4	4	0.994 500	17.500 000	
5	1	0.799 219	–	C5 violated
5	2	0.959 062	49.937 500	
5	3	0.991 031	19.968 750	
5	4	0.997 425	–	C3, C4 violated

solution with different heuristic approaches or approximations and then apply meta-heuristic methods to search for better solutions.

54.3.2 Exact Methods

In order to find an exact solution to the optimization problem, we should either compute the objective function for all possible combinations of decision variables or systematically identify all potential combinations of decision variables. Therefore, exact methods are applicable only when the problem size is very small or it posses a special properties that can be used to identify the potential optimal solutions. Many exact methods were developed before 1980 and documented in *Tillman* et al. [54.4].

Exhaustive Searching

In this method, compute the objective function for all possible combination of decision variable. In order to achieve this goal, we should know the lower and upper limits on the decision variables. This method is demonstrated for a nonrepairable system consisting of two subsystems in series. The objective is to find the optimal number of online spares in each subsystem that minimizes the average cost associated with the system

$$\begin{aligned}
&\text{Minimize } C_s = n_1c_1 + n_2c_2 + c_f[1 - R_s(n_1, n_2)] \\
&\quad \text{where,} \qquad (54.53)\\
&R_s(n_1, n_2) = R_1(n_1)R_2(n_2)\,, \\
&R_1(n_1) = 1 - (q_1)^{n_1}\,, \\
&R_2(n_2) = 1 - (q_2)^{n_2}\,, \\
&c_f = 1000,\, c_1 = 1,\, c_2 = 2\,, \\
&p_1 = 0.75,\, p_2 = 0.8
\end{aligned}$$

subject to:

$$\begin{aligned}
C1: &\quad 1 \le n_1 \le 5 \\
C2: &\quad 1 \le n_2 \le 6 \\
C3: &\quad n_1 + n_2 \le 8 \quad \text{total number constraint} \\
C4: &\quad 2.n_1 + 3.n_2 \le 20 \quad \text{weight constraints} \\
C5: &\quad R_s(n1, n2) \ge 0.9\,.
\end{aligned}$$

Because we know the explicit limits on n_1 and n_2, we can find all possible combinations of the possible solutions. For each possible solution combination, we check the other constraints. If any constraint is violated, we discard that combination and go for the next combination. For all valid combinations, we compute the objective function. The combination that correspond to the minimum value for the objective function is the optimal solution. This is described in the Table 54.1.

The optimal solution is ($n_1 = 4$, $n_2 = 4$) and the corresponding cost is 17.5. If there are no constraints for this problem, the optimal solution is ($n_1 = 5, n_2 = 4$) and the corresponding cost is 15.575.

Although it is easy to understand and program exhaustive searching methods, they are infeasible for large systems. This is because the search space increases exponentially with the problem size. Therefore, researchers proposed methods that can only search within a reduced space. *Misra* [54.20] has proposed an exact algorithm for optimal redundancy allocation problem with a reliability maximization objective based on a search near the boundary of the feasible region. This method was later implemented by *Misra* and *Sharma* [54.28], and *Misra* and *Misra* [54.29] to solve various redundancy allocation problems. This does not always gives an exact optimal solution. *Prasad* and *Kuo* [54.30] recently developed a partial enumeration method based on lexicographic search with an upper bound on system reliability. This method was demonstrated for both small and large problems. An overview of other exact methods can be found in [54.6]. With some minor modifications, the same concepts can also be used for cost-effective design.

Dynamic Programming

Dynamic programming (DP) is a solution methodology based on Richard Bellman's principle of optimality. DP is an approach for solving a wide spectrum of decision-making problems and is highly effective at solving certain types of deterministic and stochastic optimization problems. In this approach, a decision-making problem involving n variables is reduced to a set of n single-variable problems, which are derived sequentially in such a way that the solution of each problem is obtained using the preceding answer. The DP approach transforms an n-variable optimization problem into a multi-stage decision-making process. Such a transformation is not always straightforward and it quite often requires a lot of ingenuity. For this reason, it is not easy to describe the general DP approach as an algorithm. An outline of DP is given for various optimization problems in [54.6].

The basic DP approach is generally used to solve optimization problems with, at most, one constraints and with or without bounds on the decision variables. In this section, we demonstrate the DP approach solving a three-unit series system without constraints and

bounds.

Minimize

$$C_s = n_1c_1 + n_2c_2 + n_3c_3 + c_f[1 - R_s(n_1, n_2, n_3)]$$

where, (54.54)

$$\begin{aligned} R_s(n_1, n_2, n_3) &= R_1(n_1)R_2(n_2)R_3(n_3)\,, \\ R_1(n_1) &= 1 - (q_1)^{n_1}\,, \\ R_2(n_2) &= 1 - (q_2)^{n_2}\,, \\ R_3(n_3) &= 1 - (q_3)^{n_3}\,, \\ c_f &= 10,\ c_1 = 1,\ c_2 = 1,\ c_3 = 0.2\,, \\ p_1 &= 0.75,\ p_2 = 0.5,\ p_3 = 0.33\,. \end{aligned}$$

For example assume that the optimal configurations for stages 2 and 3 are fixed, i. e., n_2 and n_3 are fixed. Therefore, $n_2c_2 + n_3c_3$ and $R_2(n_2)R_3(n_3)$ are also fixed. Let us define v_i as the probability that the stages $i, \cdots, 3$ are working. Hence, the problem is reduced to:

$$\text{Minimize } C_s = n_1c_1 + c_f[1 - v_2R_1(n_1)]\,. \quad (54.55)$$

The solution to this problem depends on the value of v_2. For variable values of v_2, we find optimal value of n_1, n_1^*, using simple search techniques. We can also find the optimal solution analytically. Let the corresponding value for C_s be $f_1(v_2)$. The results are provided in Table 54.2. Now the remaining problem is to find the value of v_2 corresponding to the optimal values of n_2 and n_3. Let us assume that we know the value of n_3. Therefore, we also know $v_3 \equiv R_3(n_3)$. Hence, the problem is reduced to:

Minimize

$$\begin{aligned} C_s &= n_1c_1 + n_2c_2 + c_f[1 - v_3R_1(n_1)R_2(n_2)] \\ &= n_2c_2 + f_1[v_3R_2(n_2)]\,. \end{aligned} \quad (54.56)$$

For any fixed v_3, we can find the value of C_s for varying values of n_2. In this process, we need to compute $f_1(v)$ for given value of x_2. This can be done through interpolation. For example, the value of $f_1(v)$ for $v = 0.88$ is determined by interpolation of $f_1(0.9)$ and $f_1(0.8)$ obtained in the previous stage (see Table 54.2). Once we compute the C_s for various values of n_2, we can find the n_2^* that minimize C_s. Let the corresponding value for C_s for a fixed v_3 be $f_2(v_3)$.

Now the problem is reduced to find the value of v_3 corresponding to the optimal value of n_3. The cost function can be rewritten as

Minimize

$$\begin{aligned} C_s &= n_1c_1 + n_2c_2 + n_3c_3 + c_f[1 - R_s(n_1, n_2, n_3)] \\ &= n_3c_3 + f_2(R_3(n_3)) \end{aligned} \quad (54.57)$$

Table 54.2 Dynamic programming solution

v_2	$n_1^*(v_2)$	$f_1(v_2)$	$n_2^*(v_3)$	$f_2(v_3)$
1.0	2	2.625	1.0	6.997
0.9	2	3.563	0.9	7.617
0.8	2	4.500	0.8	8.375
0.7	2	5.438	0.7	9.031
0.6	2	6.375	0.6	9.625
0.5	1	7.250	0.5	10.125
0.4	1	8.000	0.4	10.500
0.3	1	9.750	0.3	10.875
0.2	1	9.500	0.2	11.250
0.1	1	10.250	0.1	11.125

We can find the value of $R_3(n_3)$ for varying values of n_3. Hence, we can compute $f_2(v)$ and C_s for varying values of n_3. Hence, we can find the value n_3^* that minimizes C_s. The optimal value of n_3 is $n_3^* = 7$, and the corresponding minimum cost is 8.6787. This is the minimum cost for the overall system. From this we can calculate the corresponding $v_3 = R_3(n_3^*) = 0.94$. Using the backtracking method, we can compute $n_2^*(v_3) = n_2^*(0.94) = 3$. Similarly, we can compute $v_2 = 0.82$. The optimal choice for n_1 is $n_1^*(v_2) = n_1^*(0.82) = 2$ and the corresponding v_1 is 0.77.

Therefore, the optimal solution is $(n_1^*, n_2^*, n_3^*) = (2,3,7)$ and the corresponding minimum average cost is 8.6787.

It should be noted that the whole process can be expressed using the following recursive relationship. Let $f_i(v_{i+1})$ denote the minimum expected average profit due to redundancy at stages $1, \cdots, i$, where v_{i+1} is the probability that the stages $i+1, \cdots, 3$ work. Let $v_4 = 1$. Then, for $i = 1, \cdots, 3$, we have the following recursive relationship

$$f_i(v_i + 1) = \min \arg_{n_i}\{f_{i-1}[v_{i+1}R_i(x_i)] + c_ix_i\} \quad (54.58)$$

for $i = 2, 3$. For $i = 3$, it is enough to consider only the value 1.0 for v_{i+1}. Let $n_i{}^*(v_{i+1})$ denote the value of n_i that maximizes $f_{i-1}[v_{i+1}R_i(n_i)] + c_in_i$, then

$$\begin{aligned} f_1(v_2) &= \min \arg_{n_1}\{c_fv_2[1 - (1 - r_1)^{n_1}] + c_1n_1\}, \\ f_2(v_3) &= \min \arg_{n_2}\{f_1(v_3[1 - (1 - r_2)^{n_2}]) + c_2n_2\}, \\ f_3(1.0) &= \min \arg_{n_3}\{f_2[1 - (1 - r_3)^{n_3}] + c_3n_3\}\,. \end{aligned} \quad (54.59)$$

Although recursive relations are available, it is not computationally feasible to evaluate $f_i(v)$ for all v in the

continuous interval [0, 1]. Thus, $f_i(v)$ is evaluated on a grid of v for $i = 1$ and 2. Linear interpolation is done when $f_i(v)$ is needed, but not available, for a required value of v.

54.4 Hybrid Optimization Algorithms

54.4.1 Motivation

The problem of finding the optimal number of spares is known as a redundancy allocation problem. It is a nonlinear programming problem involving integer variables [54.31]. *Chern* [54.18] has shown that the problem is NP-hard even for series–parallel systems. A summary of approaches to determine optimal or near optimal solutions in this area is presented in [54.2, 3]. In most of the published articles, the objective is the maximization of a system performance measure such as reliability or availability that satisfies a set of given constraints. In a cost-effective solution, the objective should be the minimization of the total costs associated with the system, which is the sum of the cost of initial spares, cost of maintenance, and cost of downtime. The publications on cost-effective designs are limited [54.9, 12, 32, 33]. Furthermore, even though the majority of systems are repairable, few publications exist on the optimal design issues of repairable systems [54.12, 21, 33, 34].

In general, it is difficult to obtain the optimal solution for complex systems. There exist several heuristics algorithms [54.35–37] in the literature to find near-optimal solutions. However, the efficiency and computational effort of those methods depend on the lower and upper bounds of the decision variables [54.2]. In this chapter we present a new method to find tighter bounds for obtaining the optimal number of spares. The main contributions of this research is a development of a new method to find tighter bounds for the optimal number of spares for each subsystem. The efficiency of the proposed bounds is demonstrated through several examples.

54.4.2 Rationale for the Hybrid Method

Under some mild assumptions the cost-effective solutions of both repairable and nonrepairable systems can be represented using the same mathematical formulation. For example, the cost-effective formulations used in Sects. 54.1 and 54.2 are in the same form. Therefore, the same optimization methods can used for solving these two problems. Although, the underlying problem is NP-hard, when there is only one subsystem in the system, we can find the exact solution in an efficient way even when spares are in cold- or hot-standby mode. Furthermore, we can find the exact solution even if the subsystem adopt the k-out-of-n configuration. Using our latest research findings, we can show that, when the cost of failure is the same, optimal value obtained for the subsystem in a stand-alone analysis is in fact the upper bound in the entire analysis. The upper bound can be used find the lower bounds. Further, using these lower and upper bounds, we can find better bounds in an iterative way until we reach stable bounds. This new way of finding bounds reduces the search space considerably and hence improves the solution quality.

We first describe this method for repairable systems. Then we apply the same method for the nonrepairable case.

54.4.3 Repairable Systems

Cost Minimization with One Subsystem

Before solving the actual problem, first consider a simpler problem that contains only one subsystem, i. e., $m = 1$. Say it is the subsystem i. Therefore, the cost of the system is

$$T(n_i) = c_i n_i + c_f [1 - A_i(n_i)] \ ,$$
$$A_i(n_i) \equiv \text{binf}(n_i - k_i; q_i, n_i) \ . \qquad (54.60)$$

It should be noted that the exact solution for this problem cannot be found with simple searching methods, where adding spares is stopped once a spares quantity that increases the overall system cost is reached. This is because there can exist multiple discontinuous regions where the average system cost increases with additional spares. For details, refer to [54.32]. Therefore, we need efficient algorithms to solve these optimization problems. Optimal design policies for k-out-of-n systems, i. e., systems with single subsystems, are studied in [54.9, 33].

Definition

$$f(n_i) \equiv A_i(n_i + 1) - A_i(n_i)$$
$$= \binom{n_i}{k_i - 1} p_i^{k_i} q_i^{n_i - k_i + 1} \ ,$$

$$m_0 \equiv \max\left[k_i, \text{gilb}\left(\frac{k_i-1}{p_i}\right)\right] ,$$
$$m_1 \equiv \frac{T(k_i)}{c_i} ,$$
$$m_2 \equiv \inf\{n_i \in [m_0, m_1] : f(n_i) < c_i/c_f\} . \quad (54.61)$$

It should be noted that m_2 can be evaluated efficiently using binary searching methods. When sequential searching is performed, we can use the following relationship to reduce the computational efforts

$$f(n_i) = f(n_i-1)\frac{n_i q_i}{n_i - k_i + 1} ,$$
$$f(k_i - 1) = 1 . \quad (54.62)$$

Theorem 54.1

For fixed k_i, p_i, c_i, and c_f, the optimal value of n_i that minimizes $T(n_i)$ is n_i^*:

if $f(m_0) < c_i/c_f$, then $n_i^* = k_i$
else if $f(k_i) \geq c_i/c_f$, then $n_i^* = m_2$
else if $T(k_i) > T(m_2)$, then $n_i^* = m_2$
else $n_i^* = k_i$.

Proof: The form of the cost function for this problem is equivalent to the form of the cost function used in references [54.9, 33] for minimizing the total cost of a nonrepairable k-out-of-n system. Therefore, the underlying mathematical problem is the same. For details, refer to [54.9, 33]. ■

Corollary 54.1

If the configuration of the subsystem is parallel, i. e., $k_i = 1$, then for fixed p_i, c_i, and c_f, the optimal value of n_i that minimizes $T(n_i)$ is $n_i^* = e_1$

$$e_0 \equiv \frac{\ln\left(\frac{c_i}{c_f \cdot p_i}\right)}{\ln(q_i)} ,$$
$$e_1 \equiv \max[1, \text{gilb}(e_0) + 1] . \quad (54.63)$$

Proof: For parallel systems, $f(n) = q^n p$ [54.33]. Therefore, the rest of the proof is straightforward from Theorem 54.1. It should be noted that Corollary 54.1 is a special case of Theorem 54.1. ■

Corollary 54.2

For fixed k_i, p_i, and c_i, the optimal value of n_i that minimizes $T(n_i)$ is nondecreasing with an increase in c_f.

Proof: Because c_i/c_f is decreasing, $k_i \leq m_2$, and $f(n_i)$ is decreasing in $[m_0, m_1]$, the proof is straightforward. ■

Cost Minimization with Multiple Subsystems

The system consists of several subsystems. Except for one subsystem (say subsystem i), the configurations of these subsystems are fixed. The system availability can be expressed using conditional probabilities based on Shannon's decomposition formula [54.10]. The theorem related to this formula is known as the total probability theorem [54.7]. Let us consider that subsystem i is the pivotal element (also called the key element) in the decomposition; then

$$\begin{aligned} A(\boldsymbol{n}) &= h(A_1, \cdots, A_m) \\ &= A_i h_i^1 + (1 - A_i) h_i^0 \\ &= \left(h_i^1 - h_i^0\right) A_i + h_i^0 . \end{aligned} \quad (54.64)$$

Here h_i^1 and h_i^0 are the conditional availabilities of the system given that subsystem i is operating and nonoperating, respectively. Therefore, the total system cost with n_i components in subsystem i is

$$\begin{aligned} T(n_i) &= \sum_{j=1}^{m} c_j n_j + c_f\left[1 - \left(h_i^1 - h_i^0\right) A_i(n_i) - h_i^0\right] \\ &= c_0 + c_f\left(1 - h_i^1\right) + c_i n_i \\ &\quad + c_f\left(h_i^1 - h_i^0\right)\left[1 - A_i(n_i)\right] , \\ c_0 &= \sum_{j=1; j\neq i}^{m} c_j n_j . \end{aligned} \quad (54.65)$$

Because $\left[c_0 + c_f\left(1 - h_i^1\right)\right]$ is independent of n_i, minimizing $T(n_i)$ in (54.65) is equivalent to minimizing $T_1(n_i)$ in (54.66)

$$T_1(n_i) = c_i n_i + c_f\left(h_i^1 - h_i^0\right)\left[1 - A_i(n_i)\right] . \quad (54.66)$$

The cost equations (54.60) and (54.66) are similar. Therefore, we can find the optimal n_i using Theorem 54.1. However, instead of using c_f, we need to use the modified system cost $c_f\left(h_i^1 - h_i^0\right)$ in Theorem 54.1. The results are presented in Corollary 54.3.

Definition

$$m_2' \equiv \inf\left\{n_i \in [m_0, m_1] : f(n_i) < \frac{c_i}{c_f\left(h_i^1 - h_i^0\right)}\right\} . \quad (54.67)$$

Corollary 54.3

For fixed configurations of all subsystems (except subsystem i) and for fixed k_i, p_i, c_i, and c_f, the optimal value of n_i that minimizes $T(n_i)$ is n_i^*:

$$\begin{aligned}
&\text{if } f(m_0) < \frac{c_i}{c_f\left(h_i^1 - h_i^0\right)}, \text{ then } n_i^* = k_i \\
&\text{else if } f(k_i) \geq \frac{c_i}{c_f\left(h_i^1 - h_i^0\right)}, \text{ then } n^* = m_2' \\
&\qquad \text{else if } T(k_i) > T(m_2), \text{ then } n_i^* = m_2' \\
&\qquad\qquad \text{else } n_i^* = k_i .
\end{aligned}$$

Proof: The proof is straightforward from Theorem 54.1. ■

Corollary 54.4

If all subsystems are in series and the configuration of each subsystem is parallel, then for fixed configurations of all subsystems (except subsystem i) and for fixed h_i^1, p_i, c_i, and c_f, the optimal value of n_i that minimizes $T(n_i)$ is $n_i^* = e_3$:

$$e_2 \equiv \frac{\ln\left(\frac{c_i}{c_f h_i^1 p}\right)}{\ln(q)} ,$$

$$e_3 \equiv \max[1, \text{gilb}(e_2) + 1] . \tag{54.68}$$

Proof: For series systems: $h_i^0 = 0$ and $A_i(n_i) = 1 - q_i^{n_i}$. Hence, the rest of the proof follows from Corollaries 1 and 3. ■

It should be noted that Corollary 54.4 is also applicable even if all subsystems are not in series. The only requirement is that subsystem i is in series with the rest of the system.

Corollary 54.5

The optimal n that minimizes the average system cost is always less than or equal to the n that minimizes average cost of the subsystem with the same failure cost.

Proof: Because $c_f\left(h_i^1 - h_i^0\right) \leq c_f$, from Corollary 54.2, the proof is straightforward. ■

Simultaneous Optimization

It is well known that simultaneous optimization of the components in all subsystems is a nonlinear integer programming problem consisting of a vector of decision variables. It is important to know the bounds on the decision variables for applying general-purpose optimization algorithms such as GA or SA. The efficiency of optimization algorithms, in terms of computational time as well as solution quality, can be improved by reducing the search space [54.6].

It is reasonable to assume that all subsystems must contain at least the minimum number of components that are required for its successful operation. This assumption is valid as long as we are not allowed to change the system configuration by removing some subsystems. In this chapter, we assume that we are not allowed to change the system configuration and subsystem i must have at least k_i components. Therefore, $n_i^L = k_i$; i. e., the default lower bound is $\boldsymbol{n}^L = \boldsymbol{n}^K$.

Furthermore, it is well known that the maintenance cost of all components should be less than or equal to the cost of downtime. Therefore,

$$\sum_{i=1}^{m} n_i c_i \leq c_f . \tag{54.69}$$

In the worst case, the above equation can lead to the following upper bound on n_i^* [54.33]

$$n_i^U = \text{gilb}\left(\frac{c_f}{c_i}\right) . \tag{54.70}$$

However, a better upper bound [54.33] could be:

$$n_i^U = k_i + \text{gilb}\left(\frac{c_f[1 - A(\boldsymbol{n}^K)]}{c_i}\right) . \tag{54.71}$$

Within a given set of lower and upper bounds, the total number of solution vectors in the search space is equivalent to M

$$M = \prod_{i=1}^{m} \left(n_i^U - n_i^L + 1\right) . \tag{54.72}$$

In general, these lower and upper bounds are very wide (refer to the examples). At the same time, there seems to be no method that finds the better bounds in a systematic way. In this chapter, we show that the results presented in the previous sections can be used to find tighter bounds that reduce the search space and hence increase the efficiency of the solution procedure.

Upper Bound. According to Corollary 54.5, the upper bound on n_i^* (i. e., n_i^U) that minimizes the average system cost is always less than or equal to the n_i^* that minimizes the average cost of the subsystem with the same failure cost. Therefore, the optimal value obtained using Theorem 54.1 is an upper bound for n_i^*. An improved upper bound can be found using Theorem 54.2.

Theorem 54.2

If $\boldsymbol{n}^L$ is a lower bound for $\boldsymbol{n}^*$ and $\boldsymbol{n}^U$ is an upper bound for $\boldsymbol{n}^*$, then the improved upper bound on n_i^* is equal

to $n_i^{U'}$.

$$
\begin{aligned}
n_i^{U'} &\equiv \text{optimal value obtained using Corollary 54.3 with the following parameters,}\\
h_i^1 &\equiv h_i^1(\boldsymbol{n}^{\mathrm{U}}) = \text{conditional availability with } \boldsymbol{n}^{\mathrm{U}} \text{ given that subsystem } i \text{ is operating}\\
&\equiv h\left(A_1^{\mathrm{U}}, \cdots, A_i^{\mathrm{U}} = 1, \cdots, A_m^{\mathrm{U}}\right) ,\\
A_j^{\mathrm{U}} &\equiv \mathrm{binf}(n_j - k_j; q_j, n_j^{\mathrm{U}}) ,\\
h_i^0 &\equiv h_i^0(\boldsymbol{n}^{\mathrm{L}}) = \text{conditional availability with } \boldsymbol{n}^{\mathrm{L}} \text{ given that subsystem } i \text{ is nonoperating}\\
&\equiv h\left(A_1^{\mathrm{L}}, \cdots, A_i^{\mathrm{L}} = 0, \cdots, A_m^{\mathrm{L}}\right) ,\\
A_j^{\mathrm{L}} &\equiv \mathrm{binf}\left(n_j - k_j; q_j, n_j^{\mathrm{L}}\right) .
\end{aligned}
\tag{54.73}
$$

Proof: Because the system is coherent, $h_i^1(\boldsymbol{n}^{\mathrm{U}})$ is greater than or equal to $h_i^1(\boldsymbol{n}^*)$, and $h_i^0(\boldsymbol{n}^{\mathrm{L}})$ is less than or equal to $h_i^0(\boldsymbol{n}^*)$. Therefore, $(h_i^1 - h_i^0)$ at the optimal point is always less than or equal to $[h_i^1(\boldsymbol{n}^{\mathrm{U}}) - h_i^0(\boldsymbol{n}^{\mathrm{L}})]$. From Corollary 54.2, the optimal value is decreasing with a decrease in c_{f} (the effective c_{f}). Therefore, n_i^{U}, obtained from Theorem 54.2 using Corollary 54.3, is the better upper bound. ■

Lower Bound. The default lower bound is $\boldsymbol{n}^{\mathrm{L}} = \boldsymbol{n}^{\boldsymbol{K}}$. The upper bound $\boldsymbol{n}^{\mathrm{U}}$ can be computed using the procedure presented in Sect. 54.4.3. In this section, we provide an improved lower bound for a given set of previous lower and upper bounds: $\boldsymbol{n}^{\mathrm{L}}$ and $\boldsymbol{n}^{\mathrm{U}}$.

Theorem 54.3

If $\boldsymbol{n}^{\mathrm{L}}$ is a lower bound for $\boldsymbol{n}^*$ and $\boldsymbol{n}^{\mathrm{U}}$ is an upper bound for $\boldsymbol{n}^*$, then the improved lower bound on n_i^* is equal to $n_i^{L'}$

$$
\begin{aligned}
n_i^{L'} &\equiv \inf\{n_i \geq n_i^{\mathrm{L}} : A_i(n_i) \geq B_i\}\\
B_i &\equiv \frac{A(\boldsymbol{n}^{\mathrm{U}}) - h_i^0(\boldsymbol{n}^{\mathrm{U}}) - \frac{(C_{\mathrm{U}} - C_{\mathrm{L}})}{c_{\mathrm{f}}}}{h_i^1(\boldsymbol{n}^{\mathrm{U}}) - h_i^0(\boldsymbol{n}^{\mathrm{L}})} .
\end{aligned}
\tag{54.74}
$$

Proof: It should be noted that the average cost of the system at the optimal point should always be less than or equal to the cost of the system at any other $\boldsymbol{n}$. Hence,

$$
T(\boldsymbol{n}^*) \leq T(\boldsymbol{n}^{\mathrm{U}}) = C_{\mathrm{U}} + c_{\mathrm{f}} U(\boldsymbol{n}^{\mathrm{U}}) . \tag{54.75}
$$

Maintenance cost is a strictly increasing function in $\boldsymbol{n}^*$. Therefore, $C_{\mathrm{L}} \leq T_{\mathrm{m}}(\boldsymbol{n}^*)$. Hence,

$$
\begin{aligned}
&C_{\mathrm{L}} + c_{\mathrm{f}} U(\boldsymbol{n}^*) \leq C_{\mathrm{U}} + c_{\mathrm{f}} U(\boldsymbol{n}^{\mathrm{U}}) ,\\
&\left[h_i^1(\boldsymbol{n}^*) - h_i^0(\boldsymbol{n}^*)\right] A_i(n_i^*) + h_i^0(\boldsymbol{n}^*)\\
&\qquad \geq A(\boldsymbol{n}^{\mathrm{U}}) - \frac{(C_{\mathrm{U}} - C_{\mathrm{L}})}{c_{\mathrm{f}}} .
\end{aligned}
\tag{54.76}
$$

Because the system is coherent, we have

$$
\begin{aligned}
&h_i^0(\boldsymbol{n}^{\mathrm{U}}) \geq h_i^0(\boldsymbol{n}^*) ,\\
&h_i^1(\boldsymbol{n}^{\mathrm{U}}) - h_i^0(\boldsymbol{n}^{\mathrm{L}}) \geq h_i^1(\boldsymbol{n}^*) - h_i^0(\boldsymbol{n}^*) .
\end{aligned}
\tag{54.77}
$$

Hence,

$$
A_i(n_i^*) \geq \frac{A(\boldsymbol{n}^{\mathrm{U}}) - h_i^0(\boldsymbol{n}^{\mathrm{U}}) - \frac{(C_{\mathrm{U}} - C_{\mathrm{L}})}{c_{\mathrm{f}}}}{h_i^1(\boldsymbol{n}^{\mathrm{U}}) - h_i^0(\boldsymbol{n}^{\mathrm{L}})} \equiv B_i . \tag{54.78}
$$

Because $A_i(n_i^*)$ is an increasing function n_i^*, the new lower bound is

$$
n_i^{L'} \equiv \inf[n_i : A_i(n_i) \geq B_i] . \tag{54.79}
$$

■

Corollary 54.6

If $\boldsymbol{n}^{\mathrm{L}}$ is a lower bound for $\boldsymbol{n}^*$, $\boldsymbol{n}^{\mathrm{U}}$ is an upper bound for $\boldsymbol{n}^*$, and the configuration of subsystem i is parallel, then the improved lower bound on n_i^* is equal to $n_i^{L'}$.

$$
\begin{aligned}
n_i^{L'} &\equiv \max[1, \mathrm{gilb}(L_i) + 1] ,\\
L_i &\equiv \frac{\ln(1 - B_i)}{\ln(q_i)} .
\end{aligned}
\tag{54.80}
$$

B_i is defined in (54.74).

Proof: For parallel subsystems, $A_i(n_i^*) = 1 - q_i^{n_i^*}$. Hence, the rest of the proof is straightforward from Theorem 54.3.

If the subsystem i is in series with the rest of the system, then the equation for B_i can be simplified as shown in (54.81).

$$
B_i \equiv A_i\left(n_i^{\mathrm{U}}\right)\left(1 - \frac{(C_{\mathrm{U}} - C_{\mathrm{L}})}{c_{\mathrm{f}} A(\boldsymbol{n}^{\mathrm{U}})}\right) . \tag{54.81}
$$

■

Algorithm

1. Find the default lower bound. It is $\boldsymbol{n}^{\mathrm{L}} = \boldsymbol{n}^{\boldsymbol{K}} = [k_1, \cdots, k_m]$.
2. Find the optimal value for each subsystem using Theorem 54.1, considering that there is only one subsystem in the system and the cost of failure is c_{f}. Let the optimal value for subsystem i be n_i^{U}. From Corollary 54.5, it is the upper bound for n_i^*.
3. Find $A_i(n_i^{\mathrm{L}})$ and $A_i(n_i^{\mathrm{U}})$ for each subsystem.
4. Find $A(\boldsymbol{n}^{\mathrm{L}})$ and $A(\boldsymbol{n}^{\mathrm{U}})$.
5. Find $h_i^0(n_i^{\mathrm{L}})$, $h_i^0(n_i^{\mathrm{U}})$, $h_i^1(n_i^{\mathrm{L}})$, and $h_i^1(n_i^{\mathrm{U}})$ for each subsystem. For series configurations, we

have: $h_i^0(n_i^{\mathrm{L}}) = 0$, $h_i^0(n_i^{\mathrm{U}}) = 0$, $h_i^1(n_i^{\mathrm{L}}) = \frac{A(\boldsymbol{n}^{\mathrm{L}})}{A_i(n_i^{\mathrm{L}})}$, and $h_i^1(n_i^{\mathrm{U}}) = \frac{A(\boldsymbol{n}^{\mathrm{U}})}{A_i(n_i^{\mathrm{U}})}$.

6. Using Theorem 54.2, find the updated $\boldsymbol{n}^{\mathrm{U}}$.
7. Using Theorem 54.3, find the updated $\boldsymbol{n}^{\mathrm{L}}$.
8. If stable lower and upper bounds are reached, i. e., if there are no change in the lower and upper bounds, then continue to Step 9. Otherwise (if there is a change), repeat from Step 3.
9. Using n_i^{L} and n_i^{U} as the bounds for optimal n_i, find the optimal vector that minimizes $T(\boldsymbol{n})$.
   - We can use any search algorithm for optimization including GA [54.38, 39], SA [54.40], or any other general-purpose optimization algorithms [54.35–37, 41].
   - In this chapter, we use an exhaustive searching algorithm to illustrate the example problems.

Numerical Examples

Series System with Parallel Subsystems. Consider a system consisting of five parallel subsystems arranged in a series configuration. The reliability block diagram of the system is shown in Fig. 54.1.

The failure-time distribution of a component in subsystem i is Weibull with characteristic life η_i and shape parameter β_i. There exist several forms of probability density function (PDF) for the Weibull distribution. The form of the Weibull distribution considered in this chapter is

$$f_i(t) = \frac{\beta_i}{\eta_i}\left(\frac{t}{\eta_i}\right)^{\beta_i - 1} \exp\left[-\left(\frac{t}{\eta_i}\right)^{\beta_i}\right] . \qquad (54.82)$$

Hence, the MTTF of each component (ϕ_i) is

$$\phi_i = \eta_i \Gamma(1/\beta_i + 1) . \qquad (54.83)$$

The repair-time distribution of a component in subsystem i is lognormal with parameters μ_i and σ_i. Hence,

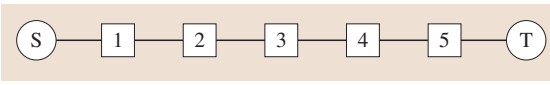

Fig. 54.1 Reliability block diagram for a series system

Table 54.4 Parameters for optimization of a series system

i	p_i	q_i	ν_i	R_i	c_i
1	0.90886	0.09114	0.00091	109.98298	0.10026
2	0.86842	0.13158	0.00174	110.67862	0.19272
3	0.97501	0.02499	0.00049	131.81396	0.06471
4	0.96746	0.03254	0.00129	83.27819	0.10706
5	0.79644	0.20356	0.00159	393.22117	0.62662

the MTTR of each component (ψ_i) is

$$\psi_i = \exp\left(\mu_i + \frac{\sigma_i^2}{2}\right) . \qquad (54.84)$$

In this example, the MLDT associated with repairs is considered to be zero for all components. Using the system parameters in Table 54.3, we can find the other parameters of the components in each subsystem. These parameters include availabilities (p_i), unavailabilities (q_i), failure frequencies (ν_i), cost of each repair (R_i), and repair cost per unit time (c_i). Values for these parameters are provided in Table 54.4.

1. Because the configuration of the system is series, the system availability expression can be obtained using (54.1).
2. Without the results of this chapter, $\boldsymbol{n}^{\mathrm{L}} = \{1, 1, 1, 1, 1\}$ and $\boldsymbol{n}^{\mathrm{U}} = \{4060, 2113, 6291, 3802, 650\}$. Therefore, the search space contains more than 1.3×10^{17} solution vectors. It is unrealistic to search for an exact solution within such a large solution space.
3. Using common-sense reasoning, we may guess that the actual optimal solution will be somewhere between the following bounds: $\boldsymbol{n}^{\mathrm{L}} = \{1, 1, 1, 1, 1\}$ and

Table 54.3 Parameters for a series system

Subsystem ID	Failure distribution Weibull			Repair distribution Lognormal			Repair cost Parameters	
i	η_i	β_i	ϕ_i	μ_i	σ_i	ψ_i	γ_i	δ_i
1	1125	2	997.01	4.00	1.10	99.98	10.00	1.00
2	540	1.3	498.73	3.20	1.50	75.57	20.00	1.20
3	2200	1.5	1986.04	3.75	0.60	50.91	30.00	2.00
4	800	1.2	752.52	3.20	0.25	25.31	20.00	2.50
5	475	0.9	499.79	4.35	1.00	127.74	10.00	3.00

Cost per unit downtime (c_{f}) = 1000 cost units

$\boldsymbol{n}^{\mathrm{U}} = \{100, 100, 100, 100, 100\}$. Even this search space contains 10^{10} solution vectors. To reduce the search space, we may guess that the upper bound might be $\boldsymbol{n}^{\mathrm{U}} = \{10, 10, 10, 10, 10\}$. This contains 10^5 solution vectors. To reduce the search space even further, we might guess that $\boldsymbol{n}^{\mathrm{U}} = \{4, 4, 4, 4, 4\}$. This search space contains 1024 solution vectors. Now, if we can compute the objective function (cost function) for each vector, we can find the exact solution. However, if the optimal vector is not within the assumed lower and upper bounds, then we will not obtain correct results. Therefore, a systematic procedure to find the bounds is essential.

4. Using Theorem 54.1 (Corollary 54.1) and Corollary 54.5, $\boldsymbol{n}^{\mathrm{U}} = \{4, 5, 3, 3, 5\}$. Even without applying the proposed lower bound, i.e, even with $\boldsymbol{n}^{\mathrm{L}} = \{1, 1, 1, 1, 1\}$, we have 900 solution vectors. The number of solution vectors using this upper bound is less than the number of solution vectors of the above wild guess (which leads to incorrect results).
5. Further, using Theorem 54.2 (Corollary 54.6), we have $\boldsymbol{n}^{\mathrm{L}} = \{3, 3, 2, 2, 4\}$. Therefore, now there are only 48 solution vectors within the bounds.
6. With exhaustive searching, the optimal vector of components that minimizes $T(\boldsymbol{n})$ is $\boldsymbol{n}^* = \{4, 5, 3, 3, 5\}$. For this example, $\boldsymbol{n}^* = \boldsymbol{n}^{\mathrm{U}}$. The corresponding system availability is 0.99949, and the corresponding minimum cost per unit time is 5.521.

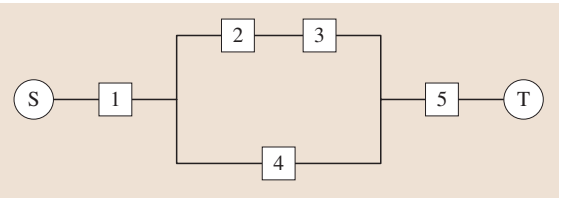

Fig. 54.2 A hypothetical reliability block diagram

Hypothetical Reliability Block Diagram. Consider a hypothetical example where subsystems are connected using the reliability block diagram shown in Fig. 54.2. The failure and repair times of each component follow exponential distributions. The logistic delay follows a deterministic time distribution. The parameters of the system are shown in Table 54.5. The parameters required for optimization are provided in Table 54.6.

1. The expression for the availability as a function of the subsystems availabilities can be obtained as follows.

$$\begin{aligned} A(\boldsymbol{n}) &= h(A_1, A_2, A_3, A_4, A_5) \\ &= A_1 A_5 \left[1 - (1 - A_2 A_3)(1 - A_4)\right] , \end{aligned} \tag{54.85}$$

   where A_i can be calculated using (54.51).
2. The default lower bound for n_i^* is $n_i^{\mathrm{L}} = k_i$. Therefore, $\boldsymbol{n}^{\mathrm{L}} = \{2, 3, 2, 3, 2\}$.
3. Without using the results of this chapter, $\boldsymbol{n}^{\mathrm{U}} = \{2860, 1472, 1889, 7765, 1386\}$. This pro-

Table 54.5 Parameters for a hypothetical reliability block diagram

Subsystem ID	Required #	MTTF/MTTR/MLDT			Repair cost	
i	k_i	ϕ_i	ψ_i	φ_i	γ_i	δ_i
1	2	1000	90	10	10.00	1.20
2	3	500	50	25	20.00	2.00
3	2	2000	40	10	30.00	2.50
4	3	750	20	5	20.00	3.00
5	2	500	100	25	10.00	3.50
Cost per unit downtime (c_{f}) = 1000 cost units						

Table 54.6 Parameters for the optimization of a hypothetical reliability block diagram

i	k_i	p_i	q_i	v_i	R_i	c_i
1	2	0.90886	0.09114	0.00091	109.98298	0.10026
2	3	0.86842	0.13158	0.00174	110.67862	0.19272
3	2	0.97501	0.02499	0.00049	131.81396	0.06471
4	3	0.96746	0.03254	0.00129	83.27819	0.10706
5	2	0.79644	0.20356	0.00159	393.22117	0.62662

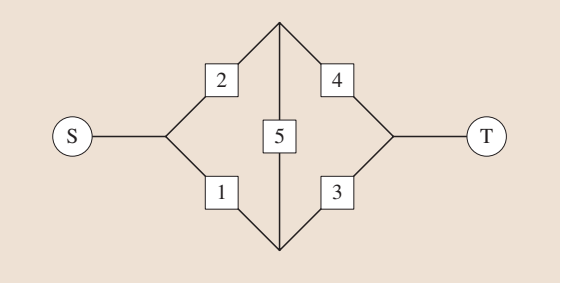

Fig. 54.3 A repairable bridge network

duces more than 8.5×10^{16} solution vectors (points of the search space).
4. Using Theorem 54.1 and Corollary 54.5, $\boldsymbol{n}^{\mathrm{U}} = \{6, 8, 5, 5, 5\}$. At present, the search space contains 1440 solution vectors.
5. Within two iterations, we obtain the stable $\boldsymbol{n}^{\mathrm{L}} = \{5, 3, 2, 3, 4\}$ and $\boldsymbol{n}^{\mathrm{U}} = \{6, 6, 4, 5, 5\}$. Now the search space contains 144 solution vectors.
6. With exhaustive searching, the optimal vector of components that minimizes $T(\boldsymbol{n})$ is $\boldsymbol{n}^* = \{6, 3, 2, 5, 5\}$. The corresponding system availability is 0.9998, and the corresponding minimum cost is 3.1065.

We observe that, in most cases, if a subsystem is in series with the rest of the system, then its optimal value is equivalent or very close to its upper bound. Therefore, using this information, we can further reduce the search space. Hence, the effective search space contains only 36 solution vectors.

Bridge Network. Consider a hypothetical example where subsystems are connected as a bridge network as shown in Fig. 54.3. The parameters used in Table 54.5 and Table 54.6 are considered for this example.

1. The expression for the availability as a function of the subsystem availabilities can be obtained using (54.86).

$$\begin{aligned} A(\boldsymbol{n}) &= h(A_1, A_2, A_3, A_4, A_5) \\ &= A_1A_3 + A_2A_4(1 - A_1A_3) \\ &\quad + A_1A_4A_5U_2U_3 + A_2A_3A_5U_1U_4 \,, \end{aligned} \tag{54.86}$$

where A_i can be calculated using (54.51).
2. The default lower bound for n_i^* is $n_i^{\mathrm{L}} = k_i$. Therefore, $\boldsymbol{n}^{\mathrm{L}} = \{2, 3, 2, 3, 2\}$.
3. Using Theorem 54.1 and Corollary 54.5, $\boldsymbol{n}^{\mathrm{U}} = \{6, 8, 5, 5, 5\}$. At present, the search space contains 1440 solution vectors.
4. Within one iteration, we obtain the stable $\boldsymbol{n}^{\mathrm{L}} = \{2, 3, 2, 3, 2\}$ and $\boldsymbol{n}^{\mathrm{U}} = \{5, 7, 4, 5, 4\}$. Now the search space contains 540 solution vectors.
5. With exhaustive searching, the optimal vector of components that minimizes $T(\boldsymbol{n})$ is $\boldsymbol{n}^* = \{5, 3, 4, 3, 2\}$. The corresponding system availability is 0.9998, and the corresponding minimum cost is 2.5631.

54.4.4 Nonrepairable Systems

The same algorithms that are used for repairable systems can also be used for nonrepairable systems. However, the individual subsystems should be solved using different methods.

The general form for the cost of the single subsystem is

$$\begin{aligned} &T(n_i) = c_i n_i + c_{\mathrm{f}}\left[1 - R(n_i)\right] \,, \\ &R(n_i) \equiv R_i(n_i) \,. \end{aligned} \tag{54.87}$$

Optimal design policies for k-out-of-n systems, i. e., systems with single subsystems, are studied in [54.9,25,33]. Theorem 54.1 presents optimal design policies for k-out-of-n cold/hot-standby systems. Let

Case 1: For hot-standby systems:

$$\begin{aligned} f(n_i) &\equiv \binom{n_i}{k_i - 1} p_i^{k_i} q_i^{n_i - k_i + 1} \\ m_0 &\equiv \max\left[k_i, \mathrm{gilb}\left(\frac{k_i - 1}{p_i}\right)\right] \,; \end{aligned} \tag{54.88}$$

Case 2: For cold-standby systems:

$$\begin{aligned} f(n_i) &\equiv \exp(-k_i\lambda_i t) k_i\lambda_i \frac{(k_i\lambda_i t)^{n_i - k_i}}{(n_i - k_i)!} \,, \\ m_0 &\equiv \max\left[k_i, \mathrm{gilb}\left(k_i\lambda_i t + k_i - 1\right)\right] \,; \end{aligned} \tag{54.89}$$

Definition

$$\begin{aligned} m_1 &\equiv \frac{T(k_i)}{c_i} \,, \\ m_2 &\equiv \inf\{n_i \in [m_0, m_1] : f(n_i) < c_i/c_{\mathrm{f}}\} \,. \end{aligned} \tag{54.90}$$

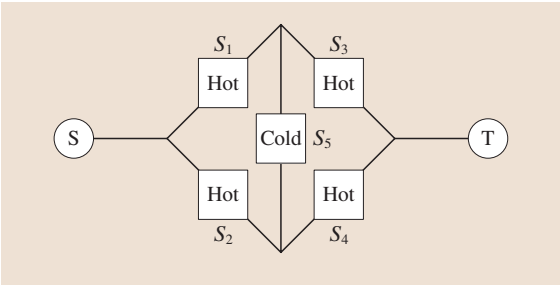

Fig. 54.4 A non-repairablebridge network

Theorem 54.4

For fixed k_i, p_i, c_i, and c_f, the optimal value of n_i that minimizes $T(n_i)$ is n_i^*:
if $f(m_0) < c_i/c_f$, then $n_i^* = k_i$
else if $f(k_i) \geq c_i/c_f$, then $n_i^* = m_2$
else if $T(k_i) > T(m_2)$, then $n_i^* = m_2$
else $n_i^* = k_i$.

Proof: Refer to [54.9, 33]. ■

Numerical Examples

Bridge Network. Consider a hypothetical example where subsystems are connected as a bridge network (Fig. 54.4). The parameters of the system are shown in Table 54.7.

1. The lower bound for n_i^* is $n_i^L = k_i$. Therefore, $\boldsymbol{n}^L = \{2, 3, 3, 2, 3\}$.
2. Without using the results of this chapter, i. e., from reference [54.32], $\boldsymbol{n}^U$ is $\{665, 334, 268, 223, 192\}$. This produces more than 2.4×10^{12} solution vectors. Therefore, exhaustive searching is almost impossible.
3. Using Theorem 54.1 and Corollary 54.3, $\boldsymbol{n}^U$ is $\{6, 6, 7, 5, 8\}$. At present, the search space contains 2, 400 solution vectors. The search space can be reduced further by using the proposed iterative procedure.
4. Within two iterations, we obtain the stable $\boldsymbol{n}^U = \{5, 6, 6, 4, 6\}$. Now the search space contains 768 solution vectors.
5. With exhaustive searching, the optimal vector of components that minimizes $T(\boldsymbol{n})$ is $\boldsymbol{n}^* = \{5, 3, 6, 2, 3\}$. The corresponding system reliability is 0.9998, and the corresponding minimum cost is 44.7640.

Table 54.7 Parameters for a bridge network

Subsystem	k_i	Standby	λ_i	p_i	c_i
S_1	2	hot	0.0001	0.904837	1
S_2	3	hot	0.00005	0.951229	2
S_3	3	hot	0.0001	0.904837	2.5
S_4	2	hot	0.00005	0.951229	3
S_5	3	cold	0.0002	0.818731	3.5

Mission time (t) = 1000 time units
Cost per unit downtime (c_f) = 10 000 cost units

The algorithm presented in this chapter reduced the search space and allowed us to use exhaustive searching to find the optimal component vector.

Series System. Consider a hypothetical space vehicle consisting of the following subsystems arranged in a series configuration:

- guided control subsystem consisting of computers,
- monitoring subsystem consisting of sensors,
- power-generating subsystem.

1. The lower bound for n_i^* is $n_i^L = k_i$. Therefore, $\boldsymbol{n}^L$ is $\{2, 3, 2\}$.
2. Because it is a series system, $h_i^0 = 0$ for all i.
3. Using Theorem 54.1 and Corollary 54.3, $\boldsymbol{n}^U$ is $\{6, 13, 10\}$.
4. In this example, with exhaustive searching, the optimal vector of components that minimizes $T(\boldsymbol{n})$ is $\boldsymbol{n}^* = \{6, 13, 10\}$, which is equivalent to $\boldsymbol{n}^U$. The corresponding system reliability is 0.9834, and the corresponding minimum cost is 139.63.

54.4.5 Conclusions

The proposed lower and upper bounds are easy to find. The computational complexity to find these bounds is linearly proportional to the number of subsystems (m). The lower and upper bounds reduce the search space drastically. Therefore, within a practical time frame, high-quality optimal solutions can be found.

The numerical results demonstrate that, if a subsystem is in series with the rest of the system, then the optimal value for that subsystem is generally equivalent to its upper bound. Therefore, the configuration of such a subsystem can be considered as fixed while performing the simultaneous optimization. This reduces the search space further. For example, in a series system, all subsystems are in series with the rest of the system. Therefore, the optimal solution vector matches with the vector of the upper bounds.

Computational Advantages

One of the main contributions of this chapter is providing the lower and upper bounds on the optimal number of spares in each subsystem. The upper bound for parallel subsystems can be found in constant time using Corollary 54.1. Therefore, the computational complexity of finding the upper bound for this case is O(1). For the general case, the upper bound can be found from Theorem 54.1 using binary searching methods. Therefore, the computational complexity for the general case in finding the upper bound is O[log(m_1)], which is equivalent to O[log(k_i)].

1. For series systems with parallel subsystems, it is known that the near-optimal solution, which is the optimal solution in most cases, is equivalent to its upper bound. Because there are m subsystems, the computation complexity is O(m).
2. For series systems with k-out-of-n subsystems, it is known that the near-optimal solution, which is the optimal solution in most cases, is equivalent to its upper bound. Because there are m subsystems, the computation complexity is O[m log(k)].
3. For the general case, where k-out-of-n type subsystems are arranged in a complex network configuration, the tighter bounds on the decision variable reduces the search space exponentially. Let $r < 1$ be the reduction factor in the width of the bounds obtained by using the proposed tighter bounds with respect to some existing wider bounds on the decision variables [54.33]. Then, the search space is reduces by r^m times. If stochastic algorithms are used to find the optimal solution, then the chances of finding the exact solution will be high with the reduced search space. Let S_o be the search space with some existing bounds, and S_r be the search space with the proposed bounds. Here $S_r = r^m S_o$. Now, randomly select M vectors from the search space. With the original search space, we can find the exact solution with probability $\frac{M}{S_o}$. With the reduced search space, we can find the exact solution with probability $\frac{M}{S_r} \equiv \frac{M}{r^m S_o}$. Therefore, the chances of finding the exact solution increases exponentially. However, evolutionary algorithms, such as GA and SA, find the exact solutions with high probability compared to pure random algorithms. Therefore, the advantage may not be exponential in this case. However, the reduced search space always increase the chances of finding better-quality solutions. If the iterations are not fixed, the same quality of solutions can be obtained quickly with the reduced search space. Our further research will be in the experimental comparison of various algorithms with and without using the proposed bounds.

References

54.1 H. A. Taha: *Operations Research: An Introduction*, 6th edn. (Prentice-Hall of India, New Delhi 2000)

54.2 W. Kuo, V. R. Prasad: An annotated overview of system-reliability optimization, IEEE Trans. Reliab. **49**, 176–187 (2000)

54.3 K. B. Misra: On optimal reliability design: A review, Int. J. Syst. Sci. **12**, 5–30 (1986)

54.4 F. A. Tillman, C. L. Hwang, W. Kuo: Optimization techniques for system reliability with redundancy: a review, IEEE Trans. Reliab. **R26**, 148–155 (1977)

54.5 S. M. Ross: *Introduction to Probability Models*, 8th edn. (Academic, New York 2003)

54.6 W. Kuo, V. R. Prasad, F. A. Tillman, C. Hwang: *Optimal Reliability Design* (Cambridge Univ. Press, Cambridge 2001)

54.7 K. B. Misra: *Reliability Analysis and Prediction: A Methodology Oriented Treatment* (Elsevier, Amsterdam 1992)

54.8 S. V. Amari, H. Pham: *Optimal cost-effective design of complex systems*, Tenth ISSAT International Conference on Reliability and Quality in Design (ISSAT (ISSAT, New Brunswick, NJ 2004) pp. 320–324

54.9 H. Pham: On the optimal design of k-out-of-n: G subsystems, IEEE Trans. Reliab. **41**, 572–574 (1992)

54.10 Y. Chang, S. V. Amari, S. Kuo: Computing system failure frequencies and reliability importance measures using OBDD, IEEE Trans. Comp. **53**, 54–68 (2003)

54.11 S. V. Amari, J. B. Dugan, R. B. Misra: Optimal reliability of systems subject to imperfect fault-coverage, IEEE Trans. Reliab. **48**, 275–284 (1999)

54.12 D. S. Bai, W. Y. Yun, S. W. Cheng: Redundancy optimization of k-out-of-n:G systems with common-cause failures, IEEE Trans. Reliab. **40**, 56–59 (1991)

54.13 L. Nordmann, H. Pham: Reliability of decision making in human-organizations, IEEE Trans. Syst. Man Cyber. A **27**, 543–549 (1997)

54.14 H. Pham: Optimal cost-effective design of triple-modular-redundancy-with-spares systems, IEEE Trans. Reliab. **42**, 369–374 (1993)

54.15 H. Pham, W. J. Galyean: Reliability analysis of nuclear fail-safe redundancy, Reliab. Eng. Syst. Safety **37**, 109–112 (1992)

54.16 H. Pham: Optimal System-Profit Design of Series-Parallel Systems With Multiple Failure Modes, Reliab. Eng. Syst. Safety J. **37**, 151–155 (1992)

54.17 H. Pham: Optimal Design of Parallel-Series Systems With Competing Failure Modes, IEEE Trans. Reliab. **41**, 583–587 (1992)

54.18 M. S. Chern: On the computational complexity of reliability redundancy allocation in a series system, Oper. Res. Let. **11**, 309–315 (1992)

54.19 D. Coit, A. E. Smith: Reliability optimization of series-parallel systems using a genetic algorithm, IEEE Trans. Reliab. **45**, 254–260 (1996)

54.20 K. B. Misra: A simple approach for constrained redundancy optimization problems., IEEE Trans. Reliab. **R21**, 30–34 (1972)

54.21 V. K. Srivastava, A. Fahim: *k*-out-of-*m* system availability with minimum-cost allocation of spares, IEEE Trans. Reliab. **37**, 287–292 (1988)

54.22 Z. W. Birnbaum, J. D. Esary, S. C. Saunders: Multi-component systems and structures and their reliability, Technometrics **3**, 55–77 (1961)

54.23 S. V. Amari, J. B. Dugan, R. B. Misra: A separable method for incorporating imperfect fault-coverage into combinatorial models, IEEE Trans. Reliab. **48**, 267–274 (1999)

54.24 J. Rupe, W. Kuo: Performability of systems based on renewal process models, IEEE Trans. Syst. Man Cyber. A **28**, 691–698 (1998)

54.25 R. C. Suich, R. L. Patterson: *k*-out-of-*n*:G systems: some cost considerations., IEEE Trans. Reliab. **40**, 259–264 (1991)

54.26 D. H. Chi, W. Kuo: Reliability optimization of series-parallel systems using a genetic algorithm, Comp. Ind. Eng. **45**, 254–260 (1996)

54.27 B. F. Mitchell, R. J. Murry: Predicting operational availability for systems with redundant, repairable components and multiple sparing levels, Proc. Ann. Reliab. Maintainab. Symp., 301-305 (IEEE, 1996),

54.28 K. B. Misra, U. Sharma: An efficient algorithm to solve integer programming problems arising in system reliability design, IEEE Trans. Reliab. **R40**, 81–91 (1991)

54.29 K. B. Misra, V. Misra: A procedure for solving general integer programming problems, Microelectron. Reliab. **34**, 157–163 (1994)

54.30 V. R. Prasad, W. Kuo: Reliability optimization of coherent systems, IEEE Trans. Reliab. **49**, 323–330 (2000)

54.31 R. E. Barlow, F. Proschan: *Mathematical Theory of Reliability* (SIAM, Philadelphia 1996)

54.32 S. V. Amari, H. Pham: Optimal design of *k*-out-of-*n*:G subsystems subjected to imperfect fault-coverage, IEEE Trans. Reliab. **53**, 567–575 (2004)

54.33 S. V. Amari: Reliability, risk and fault-tolerance of complex systems. Ph.D. Thesis (Indian Institute of Technology, Kharagpur 1997)

54.34 M. Sasaki, S. Kaburaki, S. Yanagi: System availability and optimum spare units, IEEE Trans. Reliab. **R26**, 182–188 (1977)

54.35 G. L. Bilbro: Fast stochastic global optimization, IEEE Trans. Syst. Man Cyber. **24**, 684–689 (1994)

54.36 J. M. Renders, S. P. Flasse: Hybrid methods using genetic algorithms for global optimization, IEEE Trans. Syst. Man Cyber. B **26**, 243–258 (1996)

54.37 M. Zou, X. Zou: Global optimization: An auxiliary cost function approach, IEEE Trans. Syst. Man Cyber. A **30**, 347–354 (2000)

54.38 D. E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison Wesley, Reading 1989)

54.39 M. Gen, J. Kim: GA-based reliability design: state-of-the-art survey, Comp. Ind. Eng. **37**, 151–155 (1999)

54.40 I. O. Bohachevsky, M. E. Johnson, M. L. Stein: Generalized simulated annealing for function optimization, Technometrics **28**, 209–218 (1986)

54.41 B. Li, W. Jiang: A novel stochastic optimization algorithm, IEEE Trans. Syst. Man Cyber. B **30**, 193–198 (2000)