Genetic Algorithm and Simulated Annealing for Redundancy Allocation Problem with Cold-standby Strategy

Aida Karimia, Mani Sharifib,*, Amirhossain Chambaric

a MSc, Faculty of Mechanical and Industrial Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
b Assistant Professor, Faculty of Mechanical and Industrial Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
c Young Research Club, Qazvin Branch, Islamic Azad university, Qazvin, Iran

Received 10 December, 2010; Revised 6 January, 2011; Accepted 11 February, 2011

Abstract

This paper presents a new mathematical model for a redundancy allocation problem (RAP) with cold-standby redundancy strategy and multiple component choices. The applications of the proposed model are common in electrical power, transformation, telecommunication systems, etc. Many studies have concentrated on one type of time-to-failure, but in this paper, two components of time-to-failures which follow hypo-exponential and exponential distribution are investigated. The goal of the RAP is to select available components and redundancy level for each subsystem for maximizing system reliability under cost and weight constraints. Since the proposed model belongs to NP-hard class, we proposed two metaheuristic algorithms; namely, simulated annealing and genetic algorithm to solve it. In addition, a numerical example is presented to demonstrate the application of the proposed solution methodology.

Keywords: Redundancy allocation problem; Cold-standby; Series-parallel systems; Genetic algorithm; Simulated annealing.

1. Introduction

One of the most well-known reliability optimization problems is redundancy allocation problem (RAP) which involves the selection of components from among discrete choices with appropriate levels of redundancy to maximize system reliability under some predefined constraints. The RAP has been studied in great detail as an efficient means to select sound design configurations. Furthermore, the RAP is considered for various system structures such as series, parallel, network, parallel-series Yalaoui et al. [27], k-out-of-n by Coit et al. [7]. Accordingly, the series-parallel RAP is investigated in this paper. The configuration of the series-parallel system is presented in Fig. 1. The RAP can be classified into two groups: 1) Redundancy allocation problems without component mixing (RAPCM); those problems where a mix of components within a subsystem is not allowed; and 2) Redundancy allocation problems with a mix of components (RAPMC); those problems in which a mix of components is allowed within a subsystem (Kuo et al. [16]). This problem pertains to the first classification.

Whereas in active redundancy all components are operated from the time zero simultaneously, in the standby redundancy arrangement the redundant components are sequentially used in the system during component failure times. When the component in operation fails, one of the redundant units is switched on to continue the system operation.

There are three variants of the standby redundancy; namely, cold, warm, and hot. This paper pertains to cold-standby redundancy strategy. In the cold-standby redundancy, the component does not fail before it operates by Tavakkoli-Moghaddam et al. [25]. We classified literature review in this area based on active and cold-standby redundancy.

In active redundancy, in order to maximize the reliability of the system, different methods and algorithms were
developed including dynamic programming by Fyffe et al. [9] and Nakagawa et al. [19], integer programming by Bulfin et al. [3], and different types of meta-heuristic algorithms such as genetic by Ida et al. [13] and Coit et al. [8], tabu search by Ouzineb et al. [20], variable neighborhood search Liang et al. [17], and particle swarm optimization by Beji et al. [2]. Also, it is worth mentioning that an overview of research in this area can be found in this area can be found by Kuo et al. [15]. Besides, in cold-standby redundancy, a review of more than one hundred references describing reliability optimization researches with different types of redundancy is completed by Tillman et al. [26]. Among these studies, the minority pertained to standby redundancy. Robinson et al. [22] considered the cold-standby redundancy which concentrates on system design without repairable systems. Shankar et al. [23], Gurov et al. [10] investigated the problem of imperfect switching. Also, Coit et al. [7] presented a new problem formulation and solution method to determine the optimal system design configuration when a system design includes k-out-of-n subsystems which are designed with either active or cold-standby redundancy. Coit [6] proposed strictly cold-standby redundancy as an integer programming solution in the RAP area. Sharifi et al. [24] presented an efficient model in redundancy systems for cold-standby strategy with hypo-exponential Time-to-failure distribution.

In this article a new model for redundancy allocation problems without component mixing for series-parallel systems when redundancy strategy is cold-standby is proposed. Most mathematical models of general redundancy allocation problem assume one type of time-to-failure. Nowadays, exponential distribution is receiving more attention. The conditions of time-to-failure within a particular system design are much closer to the real world. To do so, this paper proposed a new model with two time-to-failures including exponential and hypo-exponential. The objective function is maximizing system reliability under cost and weight constraints. Since the RAP has been shown to be NP-hard by Chern [5], the simulated annealing and genetic algorithms have been proposed. The remainder of this paper is organized as follows: In section 2, the problem is defined and the mathematical model is illustrated. The proposed genetic algorithm and simulated annealing for solving the problem are investigated in Section 3. In section 4, the computational experiment and the analysis of the results are provided. At the end, some conclusions and suggestions for future research are presented in section 5.

2. Problem Definition

In this section, the mathematical model of the series-parallel system with s subsystem under cost and weight constraints is illustrated. In the proposed model, redundancy strategy is cold-standby with perfect switching. For this problem, component time-to-failure is distributed according to hypo-exponential and exponential. In addition, the following assumptions are provided:

Assumptions:

- Failed components do not damage the system, and are not repaired.
- Failures of individual components are s-independent.
- The states of the elements and the system are either good or have failed.
- The RAP without component mixing is considered.
- Components are cold-standby redundant.
- The supply of components is unlimited.
- The components of reliabilities, weights and costs, are known and deterministic.

3.1. Mathematical Model

Nomenclature

- $i$ index of subsystem $i = 1, 2, \ldots, s$;
- $s$ number of subsystems;
- $n_i$ number of components used in Subsystem $i$ $n_i \in \{1, 2, \ldots, n_{\text{Max},i}\};$
- $n(n_1, n_2, \ldots, n_s)$ $n$;
- $m_i$ number of available component choices for a subsystem $i$;
- $z_i$ index of component choice used for a subsystem $i$, $z_i \in \{1, 2, \ldots, m_i\}$;
- $z(z_1, z_2, \ldots, z_s)$;
- $n_{\text{Max},i}$ upper bound for $n_i$ ($n_i \leq n_{\text{Max},i}$);
- $t$ mission time (fixed);
- $r_{i,j}(t)$ reliability at time $t$ for the $j^{th}$ available component for subsystem $i$;
- $\lambda_{i,j}$ scale parameter the exponential distribution for $j^{th}$ available component for subsystem $i$;
- $\alpha_{i,j}, \beta_{i,j}$ parameters the hypo-exponential distribution for $j^{th}$ available component for subsystem $i$;
- $w$ system-level constraint limit for weight;
- $C$ system-level constraint limit for cost;
- $c_{ij}, w_{ij}$ cost and weight for the $j^{th}$ available component for the subsystem $j^{th}$.

$R(t,z,n)$ system reliability at time $t$ for designing vectors $z$ and $n$;
The mathematical formulation can be formulated as follows:

\[
\begin{align*}
\text{Max } Z &= R(t, z, n) \\
\text{s.t.: } & \sum_{i} c_{i} n_{i} \leq C \\
& \sum_{i} w_{i} n_{i} \leq W \\
& n_{i} \in \{1,2,\ldots,n_{\text{Max}}_{i}\} \\
& z_{i} \in \{1,2,\ldots,m_{i}\}
\end{align*}
\]

Objective function (1) is defined to maximize the reliability system. Constraint (2) considers the available cost. Constraint (3) considers the available weight.

3. Metaheuristics

In this section, two metaheuristic algorithms including genetic algorithms and simulated annealing are proposed to solve the problem. In the next subsection, we describe the algorithms for our problem.

3.2. Genetic Algorithm (GA)

GA is a stochastic search algorithm that is based on the mechanism of natural selection and natural genetics. The basic concepts of GA were introduced by Holland [11]. With regards to the growing interest and simplicity of the GA and its ability for discovering good solutions fast, this metaheuristic is selected as one of the solving methodologies. In the next subsections, we present the required steps for solving the problem by a GA.

3.2.1. Chromosome Representation

Each possible solution to this problem is a collection of selected components, and \( n_{i} \) parts in parallel for each subsystem. \( n_{i} \) parts can be chosen only in one combination amongst the \( m_{i} \) available components. The solution encoding is a \( 2 \times s \) matrix. The first and second rows demonstrate type of selected components, and then the number of selected components, respectively. The columns represent subsystem. Fig. 2 presents an example of encoding solution with 14 subsystems. This matrix represents a prospective solution with four of the third component used in parallel for the first subsystem; two of the second component used in parallel for the second subsystem, etc.

3.2.2. Initial Population

A GA requires a population of potential solutions of the given problem to be initialized. The initial population of individuals is randomly generated by a number of chromosomes (population size or \( \text{pop size} \)).

3.2.3. Constraint-handling and Fitness Function

This evaluation is achieved through the computation of the cost associated with each chromosome, using the fitness function. The offspring produced by the GA operators is likely to be infeasible. The most common approach in the GA community to handle constraint is to use penalties. The idea of this method is to transform a constrained optimization problem into an unconstrained one by adding or multiplying a certain value to/by the objective function based on the amount of constraint violation presented in a certain solution by Ozgur [21]. In this study, we use the multiplicative form of the penalty function \( \text{Pen}(S) \) and the fitness function \( \text{fitn}(S) \) with the following form:

\[
\text{Pen}(s) = \begin{cases} 
0 & \text{if } f \text{ is feasible} \\
0 & \text{otherwise}
\end{cases}
\]

Where \( f(s) \) the objective is function in Eq. (1) and \( s \) represents a solution. In this approach, we search for the solution that maximize \( \text{fitn}(s) \).

3.2.4. Selection Operator

In the next phase of the genetic algorithm, the chromosomes for the next generation are selected. In this paper, a “roulette wheel selection” procedure has been applied for the selection operator.

3.2.5. Crossover Operator

The crossover operator is the basic operator of producing new chromosomes in a GA. It operates on two-parent solutions with probability \( p_{c} \) and generates offspring by recombining both parent solution features. This operator first generates a random crossover mask and then exchanges relative genes between parents according to the mask by Hou [12]. For instance, the crossover is performed as depicted in Fig. 3.

![Fig. 2. chromosome representation](image-url)
3.2.6. Mutation Operator

Mutation is the second operation in a GA method that explores the solution spaces which are not explored by crossover operator. It operates on one parent solution with probability $p_m$. In this paper, the swap mutation operator was used. In swap operator, two position row matrixes are selected randomly and their contents are swapped. For instance, the mutation is performed as depicted in Fig. 4.

3.2.7. Stopping Criteria

In this research, the stopping criteria are defined as the number of generations. The algorithm will be stopped, when that reaches a predefined number of generations.

3.3. Simulated Annealing (SA)

SA is a well-known local search metaheuristic, as presented by Aarts et al. [1]. SA is based on the Monte Carlo method introduced by Metropolis et al. [18]. This idea was originally used to simulate a physical annealing process and was applied to combinatorial optimization for the first time in the 1980s independently by Kirkpatrick et al. [14], and Cerný [4]. The pseudo code of SA algorithm is presented in Fig. 5, where the following notation is used:

- $s$ = The current solution,
- $s^*$ = The best solution,
- $s_n$ = Neighboring solution,
- $f(s)$ = The value of objective function at solution $s$,
- $n$ = Repetition counter,
- $T_0$ = Initial temperature,
- $L$ = Number of repetition allowed at each temperature level,
- $p$ = Probability of accepting $s_n$ when it is not better than $s$.

For Applying SA to the problem under consideration, some requirements should be defined including a solution representation, fitness function, and the neighborhood identification of the current solution. In what follows we present the requirements of the SA algorithm for this problem.

![Fig. 3. Example of crossover operator](image1)

![Fig. 4. Example of mutation operator](image2)

![Fig. 5. Pseudo-code SA](image3)
3.3.1. Chromosome Representation

The solution representation in SA algorithm is the same as one in genetic algorithm.

3.3.2. Neighbour Generation

Neighborhood solution from the current solution performs according to pseudo-code presented in Fig.6.

```
Generate a random number, r ∈ (0, 1)
if r ≥ 0.5
    first row is selected, Generate a random number, b ∈ (0, 1)
    if b ≥ 0.5
        two elements are selected and swapped
        one element is selected and its value replaced with a random number between 1 and mj
    end
    else
        second row is selected, Generate a random number, q ∈ (0, 1)
        if q ≤ 0.5
            one element is selected and its value replaced with a random number between 1 and nMax,i
        else
            two elements are selected and swapped.
        end
    end
End
```

Fig. 6. Pseudo-code of neighbor solution

3.3.3. Constraint-handling and Fitness Function

This section is like above mentioned the section 4.2.3 genetic algorithm.

Table 1 Component data for example

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0022</td>
<td>-</td>
<td>0.0013</td>
<td>0.0009</td>
<td>-</td>
<td>-</td>
<td>0.0002</td>
<td>0.0019</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0017</td>
<td>0.0020</td>
<td></td>
</tr>
<tr>
<td>Choice(j=1)</td>
<td>-</td>
<td>0.0007</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0008</td>
<td>0.0003</td>
<td>-</td>
<td>-</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0009</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0.0025</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0075</td>
<td>0.0042</td>
<td>-</td>
<td>-</td>
<td>0.0068</td>
<td>0.0008</td>
<td>0.0046</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>H</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>H</td>
<td>H</td>
<td>E</td>
<td>E</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>Choice(j=2)</td>
<td>-</td>
<td>0.0009</td>
<td>0.0014</td>
<td>-</td>
<td>-</td>
<td>0.0001</td>
<td>0.0016</td>
<td>0.0020</td>
<td>-</td>
<td>0.00010</td>
<td>0.0020</td>
<td>-</td>
<td>0.0017</td>
<td>0.0020</td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
<td>-</td>
<td>-</td>
<td>0.0005</td>
<td>0.0009</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0004</td>
<td>-</td>
<td>0.0008</td>
<td>0.0008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0068</td>
<td>-</td>
<td>-</td>
<td>0.0011</td>
<td>0.0023</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0006</td>
<td>-</td>
<td>0.0006</td>
<td>0.0044</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>E</td>
<td>E</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>E</td>
<td>E</td>
<td>H</td>
<td>E</td>
<td>E</td>
<td>H</td>
<td>E</td>
</tr>
<tr>
<td>Choice(j=3)</td>
<td>-</td>
<td>0.0012</td>
<td>-</td>
<td>-</td>
<td>0.0014</td>
<td>-</td>
<td>-</td>
<td>0.0011</td>
<td>0.0022</td>
<td>0.0006</td>
<td>-</td>
<td>0.0016</td>
<td>0.0011</td>
<td>0.0007</td>
</tr>
<tr>
<td></td>
<td>0.0007</td>
<td>-</td>
<td>-</td>
<td>0.0003</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0.0031</td>
<td>-</td>
<td>-</td>
<td>0.0065</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>5</td>
<td>-</td>
<td>1</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9</td>
<td>-</td>
<td>4</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>9</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>E</td>
<td>-</td>
<td>H</td>
<td>E</td>
<td>-</td>
<td>-</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>-</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>Choice(j=4)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0005</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0004</td>
<td>-</td>
<td>-</td>
<td>0.0014</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0001</td>
<td>0.0007</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0009</td>
<td>-</td>
<td>-</td>
<td>0.0005</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0022</td>
<td>0.0022</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0039</td>
<td>-</td>
<td>-</td>
<td>0.0065</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>5</td>
<td>3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>2</td>
<td>-</td>
<td>7</td>
<td>4</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>-</td>
<td>E</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>E</td>
<td>H</td>
<td>-</td>
<td>E</td>
<td>H</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

4. A Numerical Example

To evaluate the performance of the GA and SA, 33 test problems were provided by varying incrementally the available weight from 100 to 132 while fixing the available cost at $= 75$ in order to maximize system reliability at $= 250$ hours. This example is an adapted version of an example provided by Fyffe et al. [5]. In these problems, there is series-parallel system with 14 subsystems. Each subsystem has two, three or four components of choice and the number maximum of component within a subsystem has been defined to be three. The switch operates and fails as perfect switching is used. The data test problems are given in Table 1. All the test problems are coded MATLAB 7.8.0 (R2009a). Due to the stochastic nature of the proposed algorithms, for each of the test problems it is run 4 times and the best solution amongst them is considered as the final solution. The computational results for two algorithms are shown in Tables 2 and 3. The results show that there is no significant difference among the standard deviation of the two algorithms solutions. To compare the results of the best solution, we performed a paired T-test. Fig. 8 shows the results of T-test. Because of the p-value (0.187) greater than $= 0.05$, so, two algorithms have similar results and no one is better than the other one. Fig. 7 shows the standard deviation of the two proposed algorithms.
<table>
<thead>
<tr>
<th>Trial</th>
<th>SA  performance for example</th>
</tr>
</thead>
</table>
|       |                           |}

<table>
<thead>
<tr>
<th>Trial</th>
<th>GA performance for example</th>
</tr>
</thead>
</table>
|       |                           |}

Table 2

<table>
<thead>
<tr>
<th>Trial</th>
<th>SA  performance for example</th>
</tr>
</thead>
</table>
|       |                           |}

Table 3

<table>
<thead>
<tr>
<th>Trial</th>
<th>GA performance for example</th>
</tr>
</thead>
</table>
|       |                           |}

Table 3

<table>
<thead>
<tr>
<th>Trial</th>
<th>GA performance for example</th>
</tr>
</thead>
</table>
|       |                           |}

Table 3

<table>
<thead>
<tr>
<th>Trial</th>
<th>GA performance for example</th>
</tr>
</thead>
</table>
|       |                           |}
5. Conclusion

This paper proposes a new mathematical model for redundancy allocation problem for the series-parallel system with redundancy cold-standby strategy. In the proposed formulation, two types of time-to-failure including exponential and hypo-exponential are investigated. To solve the model, two metaheuristic algorithms including GA and SA are provided. The computational results indicated that the quality of solutions of two algorithms is similar. Considering the paired T-test outputs, both algorithms are efficient for this type of reliability optimization problem.

6. References


Aida Karimi et al./ Genetic Algorithm and Simulated Annealing...


