Searching Pareto Solutions for Bi-objective Series k-out-of-N System Component Allocation Problems using the Breadth First Search Method

Yasuhiro Tsujimura1, Hidemi Yamachi1, Hisashi Yamamoto2 and Yasushi Kambayashi1

Abstract: A series-parallel system is one of the most popular redundant system models. It consists of subsystems that are connected in series and each subsystem consists of interchangeable components in parallel. There are two types of such systems: all components can work for all subsystems, or the components consisting of a certain subsystem can work only in that specific subsystem. For the first case, there are some heuristic methods to approximately solve the optimal component allocation. Yamachi et al.[9,10] proposed an algorithm that obtains the exact Pareto solutions for problems that are not so large, and a solution method that obtains approximate solutions for large problems. The method was based on an evolitional algorithm. For the second case, Yamachi et al.[13] proposed the depth first search (DFS) algorithm that obtains exact Pareto solutions using the branch and bound method. The method is made for a series k-out-of-n system, which is a system consisting of k-out-of-N subsystems that are connected in series. In this paper, we propose an alternate method for the second case; a new algorithm that employs the breadth first search (BFS) method to find Pareto solutions for a series k-out-of-n system more efficiently than the existing algorithm.

Keywords: Series-parallel redundant system, fault tolerance, multi-objective optimization, breadth first search

1. INTRODUCTION

For designing a highly reliable system, one has to take into account redundancy optimization as well as pursuing the reliability of each component in the system. In general, a system consists of several subsystems and each subsystem is further constructed from various components that have a variety of reliabilities. One of the standard redundant system models is the series-parallel system, in which parallel subsystems are serially arranged (Fig. 1). This kind of system has been intensively studied. The quest is how to allocate given components in order to achieve maximum system reliability under certain constraints. Each component has different reliabilities, cost, weight, size, etc. and the quest is to find a quasi-optimal approximation through large-space searching methods. Researchers have proposed a variety of heuristic solution methods such as genetic algorithms (GAs) [1,2]. On the other hand, some researchers have proposed methods that obtain exact solutions, such as employing the surrogate constraints algorithm [3] and dynamic programming [4].

There is another type of problem in which the number of components in each subsystem is predetermined. The quest is how to allocate the components in each subsystem in order to achieve maximum reliability for the entire system [5]. For this type of problem, it is known that, in order to obtain the optimal arrangement, not only the order relation of components with reliabilities have to be taken into account, but also the concrete values of reliabilities of components. It is not easy to find such an optimal allocation [6]. For solving this type of problem, several heuristic methods are proposed [7,8]. Yamachi et al. [9,10] formulated this problem as a multi-objective problem that minimizes system cost and maximizes system reliability. They proposed an algorithm that obtains exact Pareto solutions for problems that are not so large-scale [9], and a solution method that obtains approximate solutions.
2. FORMULATION OF COMPONENT ALLOCATION PROBLEM IN A SERIES K-OUT-OF-N SYSTEM

The model we discuss is a series k-out-of-N system that consists of n subsystems that are connected in series, and i-th subsystem is also a k-out-of-n:G system. The k-out-of-n:G system works if, and only if, more than or equal to k components out of n work.

We define the following notations for formulating the series k-out-of-N system component allocation problem.

\( n \) : the number of subsystems that comprise the system
\( |A| \) : the cardinality of set \( A \)
\( i \) : subsystem \( i \) \( (i = 1, 2, \ldots, n) \)
\( N_i \) : the set of components that can work for subsystem \( i \), i.e. \( \{1, 2, \ldots, |N_i|\} \)
\( M_i \) : the set of components that is allocated to subsystem \( i \), i.e. \( M_i \subseteq N_i \)
\( \{M_i, M_2, \ldots, M_n\} \) : the set of components that is allocated to subsystem 1 through subsystem \( i \)

For each \( i \) \( (i = 1, 2, \ldots, n) \) and \( j \) \( (j = 1, 2, \ldots, |N_i|) \)

\( c_{ij} \) : the cost of component \( j \) of subsystem \( i \)
\( r_{ij} \) : the reliability of component \( j \) of subsystem \( i \)

We also define \( c_{max} = \sum_{j=1}^{k} c_{ij} \) and \( c_{max}(l) = \sum_{j=1}^{k} c_{ij} \),

where \( l \) is an arbitrary integer in \([1, n]\).

For each \( M_i \subseteq N_i \) \( (i = 1, 2, \ldots, n) \),

\( k_i \) : the minimum number of working components for subsystem \( i \) work.

\( R(M_i) \) : the reliability of subsystem \( i \), i.e. the probability that more than or equal to \( k_i \) components work, which is constructed using all the elements in set \( M_i \) that is allocated to subsystem \( i \), that is;

\[
R(M_i) = \prod_{j \in M_i} (1 - r_{ij}) \prod_{j \in \{M_i\}^c} r_{ij} \quad (1)
\]

\( C(M_i) \) : the cost of subsystem \( i \) constructed using all the elements in set \( M_i \) that is allocated to subsystem \( i \), that is;

\[
C(M_i) = \sum_{j \in M_i} c_{ij} \quad (2)
\]

For each \( l, i = 1, 2, \ldots, n \)

\( R(M_1, M_2, \ldots, M_l) \) : the reliability of \( l \) subsystems that are subsystem 1 through subsystem \( l \) when subsystem \( i \) \( (i = 1, 2, \ldots, l) \) consists of all the elements in set \( M_i \), that is

\[
\prod_{i=1}^{l} R(M_i). \]

The structure of the balance of this paper is as follows: Section 2 introduces the component allocation problem for series k-out-of-N systems and formulates it, Section 3 introduces a theorem and propositions on the problem. In Section 4, we propose an efficient algorithm to solve the problem using the theorem described in Section 3. Section 5 demonstrates the effectiveness of our algorithm through numerical experiments. We then provide a conclusion of this research work in Section 6.
$C_{i}(M_{1}, M_{2}, \ldots, M_{l})$: the cost of $l$ subsystems that are subsystem $1$ through subsystem $l$, when subsystem $i$ ($i = 1, 2, \ldots, l$), consists of all the elements in set $M_{i}$, that is;

$$\sum_{i=1}^{l} C(M_{i})$$

where $R_{i}(M_{1}) = R(M_{1})$ and $C_{i}(M_{1}) = C(M_{1})$.

We assume that the series $k$-out-of-$N$ system component allocation problems that we are discussing in this paper satisfy the following conditions. For the cost of the component $c_{i}$, it is assumed that it is an integer without losing any generality. It is also assumed that $c_{i} \leq c_{2} \leq \ldots \leq c_{i \mid N}$ for each $i = 1, 2, \ldots, n$. Furthermore, it is assumed $r_{1} \leq r_{2} \leq \ldots \leq r_{N}$ without losing any generality. Because any subsystem uses component $j_{i}$ better than component $j_{2}$, where $j_{1} < j_{2}$. $c_{i} < c_{i \mid j}$ and $r_{i} > r_{i \mid j}$.

According to the notations, we define a Pareto solution as follows.

Assume a set of practical solutions $E$ is given. We call solution $\{M_{1}, M_{2}, \ldots, M_{n}\}$ the Pareto solution only if there are no solutions $\{M_{1}', M_{2}', \ldots, M_{l}\} \in E$ that hold the following conditions:

1. $C_{i}(M_{1}, M_{2}, \ldots, M_{n}) = C_{i}(M_{1}', M_{2}', \ldots, M_{n})$ and $R_{i}(M_{1}, M_{2}, \ldots, M_{n}) > R_{i}(M_{1}', M_{2}', \ldots, M_{n})$.
2. $C_{i}(M_{1}, M_{2}, \ldots, M_{n}) < C_{i}(M_{1}', M_{2}', \ldots, M_{n})$ and $R_{i}(M_{1}, M_{2}, \ldots, M_{n}) = R_{i}(M_{1}', M_{2}', \ldots, M_{n})$.
3. $C_{i}(M_{1}, M_{2}, \ldots, M_{n}) < C_{i}(M_{1}', M_{2}', \ldots, M_{n})$ and $R_{i}(M_{1}, M_{2}, \ldots, M_{n}) > R_{i}(M_{1}', M_{2}', \ldots, M_{n})$.

Furthermore, the following conditions are assumed, which are based on the formulae given by Ashrafi et al. [12].

**Conditions**

1. Each component is available only for a specific subsystem.
2. For each subsystem, at least one component must be selected.
3. Each component has its own cost and reliability that are evaluated beforehand.
4. The occurrence of the failure of each component is statistically independent.

Based on the notations and assumptions described above, we formulate the bi-objective series $k$-out-of-$N$ component allocation problem $Par$ as follows:

**Bi-objective Series $k$-out-of-$N$ component allocation problem $Par$:**

\[
R_{i}(M_{1}, M_{2}, \ldots, M_{n}) \rightarrow \max
\]

\[
C_{i}(M_{1}, M_{2}, \ldots, M_{n}) \rightarrow \min
\]

\[
\begin{align*}
\text{s.t.} \quad & \left| M_{i} \right| \geq 1 \quad (i=1,2,\ldots,n) \\
& M_{i} \subseteq N_{i} \quad (i=1,2,\ldots,n)
\end{align*}
\]

We define that the set of Pareto solutions of the problem $Par$ as $SPM$, and propose an efficient algorithm to search $SPM$.

### 3. STRUCTURE AND PROPERTIES OF THE PROBLEM

In this section, the following three problems are considered in order to solve the problem $Par$. Through discussing these problems, useful theorems to solve $Par$ were obtained.

#### 3.1 Bi-objective and single-objective problems

As preparation for solving the problem $Par$, in this section, the following three problems are considered; two of which solve a component allocation problem for only subsystem $i$. One problem is a bi-objective component allocation problem $PS(i)$ (Pareto solutions in a single subsystem), and the other is a single-objective component allocation problem $OS(i,c)$ (Optimal solutions in single subsystem) with a specific cost $c$. The third problem is a bi-objective component allocation problem $PM(i)$ (Pareto solutions in multiple subsystems).

First, $PS(i)$, the bi-objective component allocation problem in subsystem $i$, is formulated as follows:

For each $i (i = 1, 2, \ldots, n)$

**Problem $PS(i)$: $R(M_{i}) \rightarrow \max$ $C(M_{i}) \rightarrow \min$**

\[
\begin{align*}
\text{s.t.} \quad & \left| M_{i} \right| \geq 1 \\
& M_{i} \subseteq N_{i}
\end{align*}
\]

We define $SPS(i)$ (set of Pareto solutions in a single subsystem) as the set of Pareto solutions for the problem $PS(i)$.

Second, $OS(i,c)$, the single-objective component allocation problem in the subsystem $i$ with a specific cost $c$, is formulated as follows:

For each $i (i = 1, 2, \ldots, n)$ and $c (c = 0, 1, \ldots, c_{\text{max}})$

**Problem $OS(i,c)$: $R(M_{i}) \rightarrow \max$ $C(M_{i}) = c$**

\[
\begin{align*}
\text{s.t.} \quad & \left| M_{i} \right| \geq 1 \\
& M_{i} \subseteq N_{i}
\end{align*}
\]

We define $SOS(i,c)$ (set of optimal solutions in single subsystem) as the set of optimal solutions for the problem $OS(i,c)$.

Third, $PM(i)$, the bi-objective component allocation problem (Pareto solutions in multiple subsystems) is defined as follows:
For each \( i = 1, 2, \ldots, n \)

\[
\text{Problem } PM(i): \quad R(M_1, M_2, \ldots, M_i) \rightarrow \max \\
C_i(M_1, M_2, \ldots, M_i) \rightarrow \min \\
\text{s.t. } |M_i| \geq 1 \\
M_i \subseteq N_i, \quad i = 1, 2, \ldots, l
\]

We define \( SPM(l) \) (set of Pareto solutions in multiple subsystems) as the set of Pareto solutions for \( PM(l) \), where \( SPM(1) = SPS(1) \) and \( SPM = SPM(n) \).

### 3.2 Relations among the three problems

In this subsection, the relationship between the three problems stated in the previous subsection are described, and the theorem and propositions that contribute to constructing the algorithm that solves the bi-objective component allocation design problem are deduced. We start by describing the relationship between the problems \( PM(l) \) and \( PS(l) \).

**Theorem**

(1) \( SPM(1) = SPS(1) \)

(2) For each \( i = 2, 3, \ldots, n \),

\[
SPM(i) \subseteq A(i), \quad \text{where} \\
A(i) = \{ \{M_1, M_2, \ldots, M_i\} | \{M_1, M_2, \ldots, M_{i-1}\} \in SPM(i-1), \ M_i \in SPS(i) \}
\]

**Proof**

It is obvious that (1) holds. Suppose, by way of contradiction, that \( \forall \{M_1, M_2, \ldots, M_i\} \in SPM(i) \) and \( \{M_1, M_2, \ldots, M_{i-1}\} \in SPM(i-1) \). Then, there must exist a set of Pareto Solutions \( \{M_1', M_2', \ldots, M_{i-1}'\} \) with

\[
R(M_1, M_2, \ldots, M_{i-1}) \leq R(M_1', M_2', \ldots, M_{i-1}') \quad \text{and} \\
C_i(M_1, M_2, \ldots, M_{i-1}) \geq C_i(M_1', M_2', \ldots, M_{i-1}') .
\]

This means that there must also exist a Pareto solution \( \{M_1', M_2', \ldots, M_{i-1}', M_i\} \) with

\[
R(M_1, M_2, \ldots, M_i) \leq R(M_1', M_2', \ldots, M_i') \quad \text{and} \\
C_i(M_1, M_2, \ldots, M_i) \geq C_i(M_1', M_2', \ldots, M_i')
\]

This contradicts the assumption \( \{M_1, M_2, \ldots, M_i\} \in SPM(i) \). Therefore

\( \{M_1, M_2, \ldots, M_i\} \in SPM(i-1) \). In a similar way, the assumption \( M_i \in SPS(i) \) leads to a contradiction. Therefore, (2) holds.

Using the Theorem, we can obtain \( SPM \) if we can a) obtain \( SPS(i) \) for each \( i = 1, 2, \ldots, n \), b) obtain \( A(i) \) from \( SPS(i) \) and \( SPS(i) \) for each \( i = 2, 3, \ldots, n \), c) select \( SPM(i) \) from \( A(i) \) for each \( i = 2, 3, \ldots, n \).

**Figure 3** illustrates the relations among \( SPS(i) \), \( SPM(i-1) \), \( A(i) \) and \( SPS(i) \). If we consider only the problem of obtaining \( SPM(i) \) from \( SPM(i-1) \) and \( SPS(i) \) instead of solving problems b) and c), in order to propose an algorithm, we need to find:

1) How to obtain \( SPM(i) \) from \( SPM(i-1) \) and \( SPS(i) \) for each \( i = 2, 3, \ldots, n \), and

2) How to obtain \( SPS(i) \) for each \( i = 1, 2, \ldots, n \).

First, we consider 1) how to obtain \( SPM(i) \) from \( SPM(i-1) \) and \( SPS(i) \). For this, we enumerate all the combinations of \( SPM(i-1) \) and \( SPS(i) \) to obtain all solutions of \( A(i) \). To select \( SPM(i) \) from \( A(i) \), we use an array indexed by the cost. As enumerating solutions of \( A(i) \) one by one, if the array already has a solution with the same cost, and the solution in hand has higher reliability, we replace the existing solution by the newly obtained solution. By continuing such updates, when enumeration of all the solutions is over, we have an array that contains the set of maximal reliability solutions for each cost.

In order to obtain \( SPM(i) \) from the array, we take advantage of Proposition 1, written below, which is obvious from Proposition 3' [13]; that is, the Pareto characteristics.

**Proposition 1**

Suppose \( C_i(M_1, M_2, \ldots, M_i) > C_i(M_1', M_2', \ldots, M_i') \).

If \( R(M_1, M_2, \ldots, M_i) \leq R(M_1', M_2', \ldots, M_i') \), then \( \{M_1, M_2, \ldots, M_i\} \notin SPM(i) \).
Table 2. Solutions of $SPS$

<table>
<thead>
<tr>
<th>Solution</th>
<th>$C$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SPS(1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{2}</td>
<td>8</td>
<td>0.800</td>
</tr>
<tr>
<td>{1}</td>
<td>9</td>
<td>0.900</td>
</tr>
<tr>
<td>{1,2}</td>
<td>17</td>
<td>0.980</td>
</tr>
<tr>
<td>$SPS(2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{3}</td>
<td>8</td>
<td>0.700</td>
</tr>
<tr>
<td>{2}</td>
<td>9</td>
<td>0.800</td>
</tr>
<tr>
<td>{1}</td>
<td>12</td>
<td>0.950</td>
</tr>
<tr>
<td>{1,3}</td>
<td>20</td>
<td>0.980</td>
</tr>
<tr>
<td>{1,2}</td>
<td>21</td>
<td>0.990</td>
</tr>
<tr>
<td>{1,2,3}</td>
<td>29</td>
<td>0.997</td>
</tr>
<tr>
<td>$SPS(3)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{1}</td>
<td>8</td>
<td>0.750</td>
</tr>
<tr>
<td>{3}</td>
<td>10</td>
<td>0.850</td>
</tr>
<tr>
<td>{2}</td>
<td>12</td>
<td>0.950</td>
</tr>
<tr>
<td>{1,3}</td>
<td>18</td>
<td>0.960</td>
</tr>
<tr>
<td>{1,2}</td>
<td>20</td>
<td>0.990</td>
</tr>
<tr>
<td>{2,3}</td>
<td>22</td>
<td>0.990</td>
</tr>
<tr>
<td>{1,2,3}</td>
<td>30</td>
<td>0.998</td>
</tr>
</tbody>
</table>

We use Proposition 1 to obtain $SPM(I)$ from the array. We scan the array in the ascending order of costs to find the solutions with the highest reliability. When the scanning is over, we then find the solution set $SPM(I)$ in the array. Repeating these processes until $I=n$, we finally obtain $SPM$.

Here, we prove that this procedure is able to obtain $SPM$. For this, we define $SS(I)$ as the set of solutions obtained from $A(I)$ using a procedure based on Proposition 1.

Proposition 2

$C$: Cost of node  
$R$: Reliability of node

For each $I$ ($I=1, 2, \ldots, n$), $SS(I) = SPM(I)$

Proof

Since $^{\ast}\{M_1, M_2, \ldots, M_I\} \in SPM(I)$, it is obvious that $\{M_1, M_2, \ldots, M_I\} \in SS(I)$. Therefore $\{M_1, M_2, \ldots, M_I\} \in SPM(I)$. On the contrary, $\forall \{M_1, M_2, \ldots, M_I\} \in SS(I)$. Now suppose $\{M_1, M_2, \ldots, M_I\} \in SS(I)$. Then, there must exist $\{M'_1, M'_2, \ldots, M'_I\}$ such that $R_i(M'_1, M'_2, \ldots, M'_I) \leq R_i(M_1, M_2, \ldots, M_I)$ and $C_i(M'_1, M'_2, \ldots, M'_I) \geq C_i(M_1, M_2, \ldots, M_I)$, because $\{M_1, M_2, \ldots, M_I\}$ is not a Pareto solution. However, the procedure must have removed $\{M_1, M_2, \ldots, M_I\}$, and then $\{M_1, M_2, \ldots, M_I\} \in SS(I)$. This is a contradiction. Therefore, $\{M_1, M_2, \ldots, M_I\} \in SPM(I)$.

Q.E.D.

3.3 Example of search tree

In this section, we give an example of our search tree (Fig. 4). This tree illustrates an example of the problem shown in Table 1. Table 2 shows the solutions of $SPS(I)$ with costs and reliabilities. Each node just below the start node (first layer) corresponds to each solution of $SPM(I)$. They are equal to $SPS(I)$ as stated in the Theorem. In order to obtain $SPM(2)$, we need to calculate $A(2)$ by enumerating all combinations of $SPM(1)$ and $SPS(2)$. Each node of $A(2)$ is the combination of $SPM(1)$ and $SPS(2)$, and shown in the second layer. We choose nodes stored in the array that have highest Pareto values from $A(2)$. Those selected are shown in the third layer. As selecting nodes for the array, when we

![Fig. 4 Search tree.](image-url)
find nodes with the same cost in the solution of \( A(2) \),
we choose the one with the highest reliability to store in the array.

For example, in the tree shown in Fig. 4, there are three solutions which cost 29 in \( A(2) \) (i.e.,
1 \{1\}{1.3}, 2 \{2\}{1.2} and 3 \{1,2\}{1\}).
The best reliability solution (i.e., 2 \{1,2\}{1\}) is selected to be stored in the array, and other two are discarded.

Using Propositions 1 and 2, we obtain \( SPM(2) \) from the array.
In order to obtain \( SPM(2) \), we scan the array and eliminate solutions of high cost and low reliability.
For example, \( 2 \) in Fig. 4 dominates 1, so we eliminate 4. The fourth layer of the tree in Fig. 4 shows results obtained for \( SPM(2) \).
Repeating this procedure for \( SPM(2) \) and \( SPS(3) \), we obtain \( SPM(3) \).

Next, we consider 2) in 3.2. How to obtain \( SPS(i) \).
We consider the relationship between problems \( PS(i) \) and \( OS(i,c) \), and state the relationship as Proposition 3 from Proposition 2[13], which is useful for obtaining \( SPS(i) \).

**Proposition 3**
For each \( i \) \((i=1, 2, \ldots, n)\). \( SPS(i) \subseteq \bigcup\limits_{c=0}^{c=\text{max}} \text{SOS}(i,c) \).

**Proof:** We assume \( M_i \in SPS(i) \) and denote \( C(M_i) = c \).
If \( M_i \in SOS(i,c) \) does not hold, there must exist a set \( M_i' \) with \( R(M_i) < R(M_i') \) and \( C(M_i') = c \).
This contradicts the assumption \( M_i \in SPS(i) \).
Therefore, \( M_i \in SOS(i,c) \).

In order to obtain \( SPS(i) \) from \( \bigcup\limits_{c=0}^{c=\text{max}} \text{SOS}(i,c) \),
Proposition 1' is useful.

**Proposition 1'**
For each \( i \) \((i=1, 2, \ldots, n)\), we assume \( C(M_i) > C(M_i') \).
Then, if \( R(M_i') \leq R(M_i) \), \( M_i \in SPS(i) \).

**Proof:** It is obvious from the Pareto characteristics.

In order to use Proposition 3, we need to obtain all \( SOS(i,c) \) for each \( i \) \((i=1, 2, \ldots, n)\) and \( c \).
In this paper, we enumerate all the solutions \( M_i \) \((i \in N)\), and calculate the reliability \( R(M_i) \) and cost \( C(M_i) \).
We will describe how to calculate the reliability later this section.
We then store the highest reliability \( R(M_i) \) and \( M_i \) in an array indexed by the cost \( c \), in the same way we did to obtain \( SPM(i) \).
Figure. 5 shows such an array.

In order to obtain \( SPS(i) \) from the array described immediately above, we make use of

<table>
<thead>
<tr>
<th>Cost ( c )</th>
<th>Maximum reliability ( R(M_i) )</th>
<th>Best solution ( M_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.70</td>
<td>{1}</td>
</tr>
<tr>
<td>1</td>
<td>0.72</td>
<td>{1,2}</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>( c_{\text{max}} )</td>
<td>0.96</td>
<td>{1,2,3}</td>
</tr>
</tbody>
</table>

**Proposition 1'** Using Proposition 1', we scan the array and obtain the solution set \( SPS(i) \).

### 3.4 Property for calculation of reliability

Finally, we describe a property for the efficient calculation of reliability.
As we mentioned before, in order to obtain \( SPS(i,c) \), we have to generate all combinations of the components (i.e., \( M_i \)) that perform the subsystem \( i \), and then compute their costs and reliabilities.
The reliability of a \( k \)-out-of-\( n \)-G system is given in Proposition 4 as follows. Such a system works if more than or equal to \( k \) out of \( n \) components work.

Suppose \( n \) components are available for a subsystem and \( r_i \) instead of \( r_i' \) to make it short, is the reliability of each component \( i \) \((i=1, 2, \ldots, n)\). We describe Proposition 4 from Proposition 5[13], which is useful to compute the reliabilities.

**Proposition 4**
(1) Let \( R(k,A) \) be the reliability of a \( k \)-out-of-\( l \)-G system consisting of all components in the set \( M_\text{a}(\{1,2,\ldots, l\}) \). Then,

\[
R(k, A) = \left\{ \begin{array}{ll}
\prod_{j=1}^{k} r_j & (k = l) \\
\prod_{j=1}^{k-1} r_j \cdot 1 & (k = 0)
\end{array} \right.
\]

(2) \( R(M_i) = \sum_{k=1}^{\lceil |M_i| \rceil} R(k, M_i) \).

**Proof:** (1) holds by the law of total probability, and (2) holds by the definition of \( R(M_i) \).

The recursive structure of Proposition 4 (1) reveals that it is easily represented by what is called a signal flow diagram (SFD)[14]. Figure 6 shows the SFD that illustrates the algorithm to obtain reliabilities based on Proposition 4. In Fig. 6, each node corresponds to system reliability. The figure shows, for example, system reliability \( R(3,\{1,2,3,4,5\}) \) at the top node is calculated by adding the product of \( R(2,\{1,2,3,4\}) \) as the immediate
bottom-right node and $r_5$ to the product of $R(3,\{1,2,3,4\})$ as the immediate bottom-left entry and 1-$r_5$. The nodes in right bottom are those with bold frames in Fig.6, that is $R(0, \ A)$, have the values of 1.

As shown in Fig. 6, we start from these nodes and move to the top node in order to obtain $R(M)$.

4. THE ALGORITHM OF BFS

In this section, we propose an efficient algorithm using the BFS method to solve the bi-objective series $k$-out-of-$N$ system component allocation problem using the theorem and propositions described in the previous section.

The algorithm with BFS is as follows.

Step 1. Taking advantage of Proposition 2 and Proposition 1', we obtain $SPS(i)$ for each $i$ ($i=1, \ldots, n$), using the array indexed by the cost. In constructing the array, we make use of Proposition 4 to calculate the reliabilities.

Step 2. Taking advantage of the Theorem, we calculate $A(l)$ from $SPM(l-1)$ and $SPS(l)$, and obtain an array indexed by the cost $c$, which contains the highest reliability for the cost $c$.

Step 3. Taking advantage of Proposition 1, we scan the array and obtain $SPM(l)$.

Step 4. Repeat steps 2 and 3 incrementing $l$ by one until $l = n$.

We analyze the computational complexity of this algorithm here. First, we analyze the complexity of our searching procedure. In the Step 1, we have to compare reliabilities $\sum_{i=1}^{l} (2^{n_i} - 1)$ times for all subsystem $i$ and all costs $c$ to construct the array that contains $SOS(i,c)$. Therefore the computational complexity of the search $SOS(i,c)$ is $O(2^n n)$ where $|V_i| = \mu$. Since the maximum number of $SPS(i)$ in the array is equal to $c_{i_{\text{max}}}$ for each subsystem $i$, and we need to scan the array only once, the number of searches we have to perform is at most $\sum_{i=1}^{l} c_{i_{\text{max}}}$.

Since we can assume $c_{i_{\text{max}}}$ varies the order of the number of components, the computational complexity of the scan of the array for $n$ subsystems to obtain $SPS(i)$ from $SOS(i,c)$ is $O(n \mu)$. This is insignificant. Therefore, the computational complexity of the searching procedure in Step 1 is $O(2^n n)$.

In order to construct the array in Step 2, the number of comparisons we have to perform is the product of the number of elements of $SPS(l)$ and $SPM(l-1)$. The number of elements of $SPS(l)$ is approximately $\mu$, and that of $SPM(l-1)$ is less than or equal to $c_{i_{\text{max}}} (l - 1) = \sum_{i=1}^{l} c_{i_{\text{max}}}$. Since we can assume that $c_{i_{\text{max}}}$ varies according to the order of the number of components, the number of elements of $SPM(l-1)$ also varies according to the order of $\mu$. Therefore the computational complexity of the search to construct the array can be considered as $O(\mu^2)$, and the search up to $l=n$ is $O(n^2 \mu^2)$. Since obtaining $SPM(l)$ in Step 3 requires only one scan of the array, and the number of entries in the array is approximately the same as that of $SPM$, the computational complexity of the search up to $l = n$ is $O(n^2 \mu)$. Consequently, the computational complexity of our search procedure is $O(2^n n)$.

Second, we analyze the complexity of our reliability calculation algorithm. For the reliability calculation to obtain $SOS(i, c)$ in Step 1, we have to consider all the combinations, selecting those that are more than or equal to $k_i$ of $M_i$. Each case requires multiplications of the number of arrows in Fig. 6, and addition of the number of nodes that have two
branches. Both numbers are $\sum_{i=1}^{M_i} i^2$, and the total number of multiplications and additions is twice times that of $|M_i|^2$. Consequently, the computational complexity of our reliability calculation is $O(\mu^2)$.

Therefore, the computational complexity of our algorithm is $O(2^n n)$, i.e. $O(2^n)$.

In contrast, if we enumerate all the solutions $\{M_1, M_2, \ldots, M_n\}$, the number of solutions we have to search is $\prod_{i=1}^{n} (2^{M_i}-1)$. Therefore, the computational complexity of the search of all the enumerations is $O(2^n)$.

Equation (1) for computing the reliability described in Section 2 tells that the reliability is calculated by summing up all the products of the power of the reliability of components $k$ through $|M_i|$. In other words, if we consider $k = \mu/2$ here, the computation for $R(M)$ requires $\mu |M_i|$ times $|M_i|$ multiplications from through $|M_i|$ and

$$\sum_{k=3}^{n} \frac{\mu^k}{k!} C_k = \mu^2 2^{n-2}. $$

Considering the calculation up to subsystem $n$, the computational complexity of the reliability calculation by Eq. (1) straightforwardly is $O(2^{n/2} n \mu^2)$, (i.e.. $O(2^n)$).

As the result, the computational complexity of our algorithm is $O(2^n)$, whereas that of the enumeration is $O(2^n)$.

The space complexity of our algorithm is described as follows. The maximum number of elements of the array used in Step 2 is $\sum_{i=1}^{n} c_{i, \text{max}}$ and that in Step 3 is $c_{\text{max}}$ respectively. Therefore, the space complexity of our algorithm is $O(\mu)$.

5. NUMERICAL EXPERIMENTS FOR ALGORITHM EVALUATION

We conducted numerical experiments to confirm the efficiency of our algorithm. In our experiments, we set every $k_i$ equals to $|M_i|/2$. The computing environment for the experiments was Windows XP on a CPU Pentium IV 3.0 GHz.

In order to demonstrate efficiency, both in the search algorithm using BFS and in the reliability calculation using Proposition 4, we compared the experiments using four algorithms as follows.

<table>
<thead>
<tr>
<th>Table 3. Result of experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of subsystems</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4. Comparison of DFS and BFS for larger problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of subsystems</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

PS: Pareto Solutions

362 J Jpn Ind Manage Assoc

NII-Electronic Library Service
Table 3 shows the results of the experiments using the four algorithms. All of them found the same Pareto solutions, though the proposed algorithms using BFS processed the calculation at strikingly high-speed to find the entire answer for the same Pareto solutions. We can interpret the results that algorithms BFS1 and BFS2 effectively eliminate unnecessary searches compared to the algorithms EN and DFS. The elimination of unnecessary searches has more impact on the computation time than the mere computational complexity of the algorithms analyzed in the previous section.

To show the difference between BFS1 and BFS2 clearly, we conducted other experiments using larger problems. The results are shown in Table 4. We also conducted the same experiments using the DFS algorithm to show the eminent effectiveness of BFSs. The DFS algorithm took too much computing time, and we stopped experiments that required more than 10 hours (shown as '—').

For the experiment with ten subsystems and five components for each subsystem, the DFS algorithm took more than five hours. In contrast, BFS1 and BFS2 algorithms took around 0.01 sec. Even for 15 subsystems and 15 components for each, BFS1 and BFS2 took 20 sec and 11 sec respectively. As the number of components increases, BFS2 becomes more effective than BFS1. This means that the reliability calculation algorithm using Proposition 4 is effective for cases with a large number of components.

Thus, the numerical experiments confirm the effectiveness of the proposed algorithm (i.e., BFS2).

6. CONCLUSIONS

In this paper, we proposed an efficient algorithm to search Pareto solutions of bi-objective series k-out-of-N component allocation problems that make use of problem characteristics. The outline of our algorithm is as follows:

For each subsystem $l$ such that $l = 2, \ldots, n$ ($n$ is the number of subsystems).

(1) For each subsystem, obtain the solutions of single-objective component allocation problem (cost constant and reliability maximum), and then select the Pareto solutions of the bi-objective problem for each subsystem from the solutions just obtained.

(2) Using BFS (breadth first search), from all the combinations of the Pareto solutions for subsystem $l-1$ and the Pareto solutions for only subsystem $l$ that is obtained in (1), compute the Pareto solutions for subsystems $l$ through $l$.

(3) Increment $l$ one by one and repeat (2) until $l=n$ to obtain the set of Pareto solutions for the bi-objective component allocation problem for all subsystems.

(4) Calculate the reliability using the signal flow diagram (SFD) for the $k$-out-of-$N$:G system.

Numerical experiments demonstrated the effectiveness of the algorithm we proposed in this paper. We believe that we can obtain the exact Pareto solutions of almost all realistic problems in practical time using our algorithm.

REFERENCES


[10] Yamachi H., Tsujimura Y., Yamamoto H. and Kambayashi Y.: An application and


