

## A Heuristic Method for Solving Redundancy Optimization Problems in Complex Systems

Jae Hwan Kim\* and Bong Jin Yum\*\*

\* Department of Applied Mathematics  
Korea Maritime University, Pusan, Korea

\*\* Korea Advanced Institute of  
Science and Technology, Taejon, Korea

*Keywords*— Redundancy optimization, Complex systems, Heuristic algorithm, Excursion.

### *Reader Aids*

Purpose : Provide a new heuristic algorithm

Special math needed for explanations : Optimization

Special math needed for results : Same

Results useful to : Reliability engineers and theoreticians

*Summary & Conclusions*— This paper presents a new heuristic method for solving constrained redundancy optimization problems in complex systems. The proposed method allows "excursions" over a bounded infeasible region, which may alleviate the risks of being trapped at a local optimum. Computational results show that the proposed method performs consistently better than other heuristic methods (Shi, Kohda and Inoue, simulated annealing) in terms of solution quality. In terms of computing time, the Shi method is best while simulated annealing is the slowest. Comparing the proposed and the Kohda and Inoue methods, we observe moderate increase in

computing time for the former as compared to the latter. In summary, if computing time is of primary concern, then the Shi method is the one to be recommended among the heuristic methods we considered. If, however, solution quality is of more concern and if one is willing to accept moderate increase in computing time for better solutions, then the proposed method is believed to be an attractive alternative to other heuristic methods.

## 1. Introduction

The reliability of a system can be increased by properly allocating redundancies to subsystems under various resource and technological constraints. The problem of optimal redundancy allocation also appears as a subproblem in the simultaneous optimization of component reliability and redundancy [1]. The purpose of this paper is to present a heuristic algorithm for solving such constrained redundancy optimization problems as above in complex systems.

The redundancy optimization problem is usually formulated as a non-linear integer program which is in general difficult to solve due to the considerable amount of computational effort required to find an exact optimal solution. Therefore, various heuristic methods have been developed (see [2]–[8] among others, and also [1], [9]–[11] for a review). Of particular interest is the one proposed by Kohda and Inoue [7] in which a new criterion of local optimality is presented. They showed that their method generate solutions which are optimal in "2-neighborhood", while the solutions obtained by the previous methods are optimal only in "1-neighborhood". For a problem with monotonically nondecreasing constraints (which may usually be the case for the redundancy optimization problem) the Kohda and Inoue algorithm performs a series of selection and exchange operations within the feasible region (i.e., a solution selected by a certain criterion is examined by subtracting a redundancy from one subsystem and adding a redundancy to

another to see if this exchange yields an improved feasible solution). The heuristic algorithm presented in this paper was motivated by the so called DETMAX algorithm [12] which has been successfully used for solving a nonlinear integer optimization problem that arises in optimal experimental design. The DETMAX algorithm was originally designed to handle one simple constraint (i.e., select exactly  $n$  points in a  $p$ -dimensional space such that a certain nonlinear objective function which depends on the selected  $n$  points is maximized). In this paper we modified the DETMAX algorithm to handle multiple nonlinear constraints. In the proposed method, the Kohda and Inoue requirement that exchanges (i.e., adding and subtracting a redundancy concurrently) be conducted within the feasible region is relaxed, and instead, "excursions" (i.e., examination of a series of infeasible solutions generated by some rules) are allowed over a bounded infeasible region. In an excursion, adding a redundancy needs not be immediately followed by subtracting a redundancy, or vice versa. In an iteration of the algorithm, the excursion eventually returns to the feasible region with a possibly improved solution than the starting one, or ends with a solution which is beyond the bounded infeasible region, for which case the whole algorithm stops (see Fig. 1) Thus excursions can alleviate the risks of being trapped in a local optimum. Computational experiments were conducted to compare the performance of the proposed algorithm to those of Shi's [8] and Kohda and Inoue's [7]. Also included in the experiment was simulated annealing [13, 14, 15] which has emerged as a promising heuristic search method for complex combinatorial problems.

### Notation & Nomenclature

$n$  number of subsystems

$m$  number of constraints

$x_i$  number of components in subsystem  $i$ ; a positive integer,

$$i = 1, 2, \dots, n$$

$x$  ( $x_1, x_2, \dots, x_n$ )

$x^0$  initial feasible solution

$x^f$  feasible solution generated by an iteration

$x^*$  best solution found by the algorithm

$x^c$  current best solution

$g_{ji}(x)$  amount of resource  $j$  consumed at subsystem  $i$

$b_j$  amount of resource  $j$  available

$$b_j^c = b_j - \sum_{i=1}^n g_{ji}(x_i)$$

$r_i$  reliability of a single component of subsystem  $i$

$R_i(x_i)$  reliability of subsystem  $i$  with  $x_i$  components

$Q_i(x_i)$  unreliability of subsystem  $i$  with  $x_i$  components

$R_s(x)$  system reliability

$x(\pm i)$  ( $x_1, x_2, \dots, x_i \pm 1, \dots, x_n$ ); add/subtract 1 at subsystem  $i$

$\Delta g_{ji}(\pm i)$  increment/decrement in resource  $j$  at subsystem  $i$   
for increasing/decreasing  $x_i$  by 1

$\Delta R_s(\pm i)$  increment/decrement in system reliability

for increasing/decreasing  $x_i$  by 1

$\Delta_j$  positive constant to be added to  $b_j$  to define the boundary of *BIFR*

$F$  failure set

$E$  set of solutions generated in the course of an excursion

*FR* feasible region

*BIFR* bounded infeasible region

## 2. Problem formulation

### *Assumptions*

1. The system and all of its subsystems are coherent.

2. The system consists of  $n$  subsystems, each of which us (1-out-of- $x_i : G$ ).
3. All component states are  $s$ -independent.
4. Each constraint is an increasing function of  $x_i$  and is additive among subsystems.
5. System reliability  $R_s$  is known in tems of  $R_i$ 's.

The redundancy optimization problem is formulated as:

$$\begin{aligned}
 \text{(P1)} \quad & \text{Maximize } R_s(x) \\
 & \text{subject to } \sum_{i=1}^n g_{ji}(x_i) \leq b_j, \quad j = 1, 2, \dots, m \\
 & \quad \quad \quad x_i \geq 1, \text{ integers, } i = 1, 2, \dots, n.
 \end{aligned}$$

### 3. Algorithm

The proposed algorithm consists of a series of iterations (see Fig.1), each of which always starts with the current best solution( $x^c$ ), makes an "excursion" in a bounded infeasible region(*BIFR*),and then eventually

(Case 1) returns to the feasible region(*FR*) with a solution  $x^f$ , or

(Case 2) ends with a solution  $x^t$  which is beyond the *BIFR*.

In Case (2), the whole algorithm stops and  $x^* = x^c$ . In Case (1), an attempt is made to improve  $x^f$  according to the method to be described later, and then  $x^c$  is compared with (possibly improved)  $x^f$ . If  $R_s(x^f) > R_s(x^c)$ , then  $x^c$  is replaced by  $x^f$  and a new iteration starts. Otherwise, the solutions generated in the course of excursion (i.e.,  $x^1, x^2, \dots, x^k$  in Fig. 1) are placed in the so called "failure set"  $F$  and a new iteration starts with  $x^c$ . In Case (1), if  $x^f$  (after a possible improvement) is better than  $x^c$ , then the  $F$  set constructed in previous iterations is cleared. In the beginning of the algorithm  $F$  is empty.

The excursion in an iteration begins with  $x^c(+i)$  where  $i$  is selected based upon some criterion (e.g., (1)) and is guided by  $F$ . Let  $x^r(\in BIFR)$  be a (infeasible) solution constructed at any point of the excursion. Then, the rules for continuing the excursion are as follows.

(Rule 1) If  $x^r \notin F$ , then  $x^{r+1} = x^r(-i)$  where  $i$  is selected according to some criterion (e.g., (2)).

(Rule 2) If  $x^r \in F$ , then  $x^{r+1} = x^r(+i)$  where  $i$  is selected according to some criterion (e.g., (1)).

In Rule 1, the excursion is guided to move towards the direction that improves the feasibility, while in Rule 2 it is guided to search  $BIFR$  further, which may eventually ends with a better feasible solution. Note that as the algorithm proceeds it gets more and more difficult to obtain a better feasible solution than the current best solution, which implies that the  $F$  sets are enlarged and the excursions stray widely over  $BIFR$  (i.e., the excursions examine more points in  $BIFR$  before returning to  $FR$ , or finally generate a point beyond  $BIFR$  to stop the algorithm).

Some important steps of the algorithm are described further as follows.

#### *Initial Feasible Solution*

An initial feasible solution can be generated in several ways. A simple, twophase procedure may be described as follows. The first phase of the procedure starts with  $x^0 = (1, 1, \dots, 1)$ , randomly determines  $i$ , and checks if  $x^0(+i)$  is feasible. If so, the first phase continues. Otherwise, the second phase of the procedure tries to improve  $x^0$  found in the first phase by filling the slacks of resources as much as possible. The resulting  $x^0$  is taken to be the current best solution  $x^c$  for the first iteration of the algorithm.

Another possibility is to take the final solution generated by a simple heuristic method as an initial solution. for instance, the Nakagawa and

Nakashima approach [5] extended for complex systems by Kuo et,al. [9] can be used for this purpose.

*Selection Criteria*

Let  $x^r$  be a (infeasible) solution at any point of excursion. If  $x^r \in F$ , then  $i$  for  $x^r(+i)$  is selected according to the following "forward selection criterion".

$$(1) \quad \max_{1 \leq i \leq n} [\Delta R_s(+i) / \sum_{j=1}^m \{\Delta g_{ji}(+i) / b_j\}]$$

If  $x^r \notin F$ , then  $i$  for  $x^r(-i)$  is selected according to the following "backward selection criterion".

$$(2) \quad \min_{\substack{1 \leq i \leq n \\ x_i \neq 1}} [\Delta R_s(-i) / \sum_{j=1}^m \{\Delta g_{ji}(-i) / b_j\}]$$

We do not claim that the above criteria are best for the proposed algorithm. Other existing criteria ([4], [5], [6] among others) may be also used instead of (1) or (2) with necessary modifications.

*Bounded Infeasible Region(BIFR)*

The boundary of the infeasible region to be searched is determined such that

$$|b_j^c| = |b_j - \sum_{i=1}^n g_{ji}(x_i)| \leq \Delta_j, \quad j = 1, 2, \dots, m,$$

where  $\Delta_j$ 's are predetermined positive constants. If  $\Delta_j$ 's are large, then the possibility of finding a good solution is increased since more iterations



(and therefore more excursions) can be made over a wide *BIFR*, but the computational time may be increased. On the other hand, using small  $\Delta_j$ 's may result in the reduction of the computational time, but may also reduce the possibility of finding a good solution. Therefore, tradeoffs have to be made on choosing appropriate  $\Delta_j$ 's.

Several schemes are conceivable for determining  $\Delta_j$ 's. A simple method is to set  $\Delta_j = \alpha_j b_j$  where  $\alpha_j$  is a positive constant. Another possibility may be to determine  $\Delta_j$  to allow a few decision variables to be increased by 1 additionally. For instance, when the constraints are linear (i.e.,  $g_{ji}(x_i) = c_{ji}x_i$ ),  $\delta_j$  may be set to

$$c_{j(1)} + c_{j(2)} \quad \text{or} \quad 2 \times c_{j(1)}, \quad j = 1, 2, \dots, m$$

where  $c_{j(1)}$  and  $c_{j(2)}$  are respectively the largest and the next largest values of  $\{c_{ji}, i = 1, 2, \dots, n\}$ .

If  $\Delta_j$ 's are set to somewhat large values to increase the possibility of finding a good solution, then it should be kept in mind that the number of elements in the failure set  $F$  could be inflated. One way of avoiding this difficulty is to limit the maximum allowable number of elements in  $F$  to a certain manageable value and terminate the algorithm if this maximum is reached.

#### *Improvement of Feasible Solution $x^f$*

As mentioned earlier, a feasible solution  $x^f$  generated by the excursion of an iteration is improved within *FR* by selecting  $i$  for  $x^f(+i)$  according to some criterion such as (1) and filling the slacks of resources as possible.

We now summarize the proposed algorithm as follows.

0. Determine  $\Delta_j, j = 1, 2, \dots, m$ .



1. Find an initial feasible solution  $x^0$ . Let  $x^c = x^0$  and  $x = x^c$ .
2. Let  $E = F = 0$ . Choose  $i$  according to criterion (1).  $x = x(+i)$ .  
If  $|b_j^c| > \Delta_j$  for any  $j = 1, 2, \dots, m$ , then go to step 7.
3. Let  $E = E \cup x$ . If  $x \notin F$ ,  $x = x(-i)$  where  $i$  is selected according to criterion (2). Go to step 5. Otherwise, let  $F = F \cup E$  and  $E = \emptyset$ . Select  $i$  according to criterion (1).  $x = x(+i)$ .
4. If  $|b_j^c| > \Delta_j$  for any  $j = 1, 2, \dots, m$ , then go to step 7. Otherwise, go to step 3.
5. If  $x$  is in *BIFR*, then go to step 3. Otherwise, let  $x^f = x$  and improve  $x^f$  as much as possible.
6. If  $R_S(x^f) > R_S(x^c)$ , then  $x^c = x^f$ . Let  $x = x^c$  and go to step 2. Otherwise, let  $F = F \cup E$  and  $E = \emptyset$ .  $x = x^c$ . Select  $i$  according to criterion (1).  $x = x(+i)$ . If  $|b_j^c| > \Delta_j$  for any  $j = 1, 2, \dots, m$ , then go to step 7. Otherwise, go to step 3.
7. Stop.  $x^* = x^c$ .

#### 4. Example

We apply the proposed method to the two examples considered by previous authors.

##### *Example 1*

A complex network system considered in [4], [7], [8] is shown in Fig. 2.

There is only one linear constraint,  $\sum_{i=1}^5 c_{1i}x_i \leq 20$ . The subsystem data are :

$i$	1	2	3	4	5
$r_i$	0.70	0.85	0.75	0.80	0.90
$c_{1i}$	2	3	2	3	1

The system reliability expression [8] is:

$$\begin{aligned}
 R_s(x) = & R_1(x_1)R_2(x_2)Q_3(x_3)Q_5(x_5) \\
 & + Q_1(x_1)R_3(x_3)R_4(x_4)Q_5(x_5) \\
 & + [R_1(x_1)R_3(x_3) + R_3(x_3)R_5(x_5) \\
 & + R_5(x_5)R_1(x_1) - 2R_1(x_1)R_3(x_3)R_5(x_5)] \\
 & \cdot [R_2(x_2) + R_4(x_4) - R_2(x_2)R_4(x_4)].
 \end{aligned}$$

The boundary of *BIFR* is determined by taking  $\Delta_1 = c_{j(1)} + c_{j(2)} = 6$ , and an initial solution is obtained based upon the two-phase method described in Section 3. In the following, we describe how the algorithm proceeds for the above example. See also Fig. 3.

0.  $\Delta_1 = 6$ .
1.  $x^0 = (2, 2, 1, 2, 2)$ .  $x^c = x^0$ .  $x = x^c$ .
2.  $E = F = \emptyset$ .  $x = x(+3) = (2, 2, 2, 2, 2)$ .  $|b_1^c| = 2 < \Delta_1 = 6$ .
3.  $E = E \cup x = \{(2, 2, 2, 2, 2)\}$ .  $x \notin F$ .  $x = x(-5) = (2, 2, 2, 2, 1)$ .
5.  $x \in BIFR$ . Go to step 3.
3.  $E = E \cup x = \{(2, 2, 2, 2, 2), (2, 2, 2, 2, 1)\}$ .  $x \notin F$ .  $x = x(-4) = (2, 2, 2, 1, 1)$ .
5.  $x^f = x = (2, 2, 2, 1, 1)$ .  $x^f$  can be improved to  $x^f = x(+1) = (3, 2, 2, 1, 1)$ .
6.  $R_s(x^f) = 0.9932 > R_s(x^c) = 0.9765$ .  $x^c = x^f = (3, 2, 2, 1, 1)$ .  $x = x^c$ .  
Go to step 2.
2.  $E = F = \emptyset$ .  $x = x(+4) = (3, 2, 2, 2, 1)$ .  $|b_1^c| = 3 < \Delta_1 = 6$ .
3.  $E = E \cup x = \{(3, 2, 2, 2, 1)\}$ .  $x \notin F$ .  $x = x(-4) = (3, 2, 2, 1, 1)$ .
5.  $x^f = (3, 2, 2, 1, 1)$ .  $x^f$  cannot be improved.
6.  $R_s(x^f) = R_s(x^c)$ .  $F = F \cup E = \{(3, 2, 2, 2, 1)\}$ .  $E = \emptyset$ .  $x = x^c = (3, 2, 2, 1, 1)$ .  $x = x(+4) = (3, 2, 2, 2, 1)$ .  $|b_1^c| = 3 < \Delta_1 = 6$ . Go to step 3.

3.  $E = E \cup x = \{(3, 2, 2, 2, 1)\}$ .  $x \in F$ .  $F = F \cup E = \{(3, 2, 2, 2, 1)\}$ .  
 $E = \emptyset$ .  $x = x(+3) = (3, 2, 3, 2, 1)$ .
4.  $|b_1^c| = 5 < \Delta_1 = 6$ . Go to step 3.
3.  $E = E \cup x = \{(3, 2, 3, 2, 1)\}$ .  $x \notin F$ .  $x = x(-1) = (2, 2, 3, 2, 1)$ . Go to step 5.
5.  $x \in BIFR$ . Go to step 3.
3.  $E = E \cup x = \{(3, 2, 3, 2, 1), (2, 2, 3, 2, 1)\}$ .  $x \notin F$ .  $x = x(-4) = (2, 2, 3, 1, 1)$ . Go to step 5.
5.  $x^f = (2, 2, 3, 1, 1)$ .  $x^f$  cannot be improved.
6.  $R_s(x^f) = 0.9923 < R_s(x^c) = 0.9932$ .  $F = F \cup E = \{(3, 2, 2, 2, 1), (3, 2, 3, 2, 1), (2, 2, 3, 2, 1)\}$ .  $E = \emptyset$ .  $x = x^c = (3, 2, 2, 1, 1)$ .  $x = x(+4) = (3, 2, 2, 2, 1)$ .  $|b_1^c| = 3 < \Delta_1 = 6$ . Go to step 3.
3.  $E = E \cup x = \{(3, 2, 2, 2, 1)\}$ .  $x \in F$ .  $F = F \cup E = \{(3, 2, 2, 2, 1), (3, 2, 3, 2, 1), (2, 2, 3, 2, 1)\}$ .  $E = \emptyset$ .  $x = x(+3) = (3, 2, 3, 2, 1)$ .
4.  $|b_1^c| = 5 < \Delta_1 = 6$ . Go to step 3.
3.  $E = E \cup x = \{(3, 2, 3, 2, 1)\}$ .  $x \in F$ .  $F = F \cup E = \{(3, 2, 2, 2, 1), (3, 2, 3, 2, 1), (2, 2, 3, 2, 1)\}$ .  $E = \emptyset$ .  $x = x(+4) = (3, 2, 3, 3, 1)$ .
4.  $|b_1^c| = 8 > \Delta_1 = 6$ . Go to step 7.
7. Stop.  $x^* = x^c = (3, 2, 2, 1, 1)$ .

The best solution obtained is  $x^* = (3, 2, 2, 1, 1)$  for which  $R_s(x^*) = 0.9932$ . It is the global optimal solution as shown in [7]. Kohda and Inoue [7] Shi [8] also obtained the same solution, while Aggarwal [4] obtained  $(2, 1, 3, 2, 1)$  for which  $R_s = 0.9921$ . In fact, the problem was solved 10 times using the proposed algorithm, each with a different initial feasible solution generated by the two-phase procedure, and 6 out of 10 cases yielded the optimal solution.

*Example 2*

The second example was considered in [8], and the corresponding system structure is illustrated in Fig. 4. The problem is to

$$\begin{aligned} \text{Maximize } R_s(x) = & R_1(x_1) + Q_1(x_1)R_2(x_2)R_4(x_4) \\ & + Q_1(x_1)R_2(x_2)R_3(x_3)Q_4(x_4) \end{aligned}$$

$$\text{subject to } \sum_{i=1}^4 c_{1i}x_i \leq 30$$

$$\sum_{i=1}^4 c_{2i}x_i \leq 40$$

$$x_i \geq 1, \text{ integers, } i = 1, 2, 3, 4.$$

The subsystem data are as follows.

$i$	1	2	3	4
$r_i$	0.80	0.75	0.70	0.65
$c_{1i}$	6	4	3	2
$c_{2i}$	9	4	4	3

With  $\Delta_1 = 2 \times c_{1(1)} = 12$ ,  $\Delta_2 = 2 \times c_{2(1)} = 18$ , and an initial feasible solution  $x^0 = (2, 1, 1, 4)$ , the proposed algorithm yields a solution  $x^* = (3, 1, 1, 1)$  for which  $R_s(x^*) = 0.9974$ . Based upon a complete enumeration, we found that this is the global optimal solution. For the same problem, Shi [8] obtained  $x^* = (2, 2, 1, 3)$  for which  $R_s(x^*) = 0.9970$ . This problem was also solved 10 times by the proposed algorithm with different initial feasible solutions, and 5 out of 10 cases yielded the optimal solution.

## 5. Computational experiments and results

The performance of the proposed method was compared with those of Shi's, Kohda and Inoue's, and simulated annealing methods.

The proposed method is most closely related to Kohda and Inoue's in the sense that it also employs procedures for alleviating the risks of being trapped in a local optimum. The Shi method is included in the experiment due to its unique feature. That is, it is different from the other heuristic methods in that it is based on minimal paths, and therefore, is believed to yield solutions with relatively less computing time.

Recently, a class of heuristic search strategies has emerged for solving complex combinatorial problems, with implications for the field of artificial intelligence. They include simulated annealing, genetic algorithm, neural networks, tabu search, etc. [16]. Although the primary purpose of the computational experiment is to compare the performance of the proposed method to those of the previous heuristics designed to solve the redundancy optimization problems, we also included simulated annealing in the experiment due to its popularity for solving complex combinatorial problems.

The problems considered in the experiment have the same structure as (P1) except that the constraints are replaced by

$$\sum_{i=1}^n c_{ji}x_i \leq b_j, \quad j = 1, 2, \dots, m.$$

Test problems were generated for the following combinations of problem parameters.

$$n = 7 \text{ (Fig. 5a)}, 10 \text{ (Fig. 5b)}, 15 \text{ (Fig. 5c)}$$

$$m = 1, 5$$

$$\{b_j\} = \text{'large'}, \text{'small'}$$

This results in 12 sets of test problems. For each set, 10 test problems were generated according to the following schemes.

$\{c_{ji}\} =$  random uniform deviates from  $(0, 100)$

$\{r_i\} =$  random uniform deviates from  $(0.6, 0.8)$

$\{b_j\} = \{w_j \times \sum_{i=1}^n c_{ji}\}$ .  $w_j =$  random uniform deviates from

$(1.5, 2.5)$  for 'small'  $\{b_j\}$  and from  $(2.5, 3.5)$  for 'large'  $\{b_j\}$ .

The four heuristic methods were programmed in FORTRAN, and each of 120 test problems was solved by the four methods on a HP-9000 computer.

For each set of test problems, an expression for the system reliability was obtained by the method of recursive disjoint products [17].

We tried two schemes for generating initial feasible solutions. One is the two-phase method, and the other is the extended Nakagawa and Nakashima method (see Section 3). Various test problems were solved with initial solutions obtained by both methods (when the two-phase method is applied, each problem was solved 10 times, each with a different initial solution), and we found that the Nakagawa and Nakashima method generally yielded better results with less computing time. Therefore, in the computational experiment, initial feasible solutions for simulated annealing, the proposed, and Kohda and Inoue methods were generated by the extended Nakagawa and Nakashima method using 0.5 as a balancing coefficient (see Section 3). For the Shi method initial solutions were set to  $(1, 1, \dots, 1)$  since using an initial feasible solutions generated by the extended Nakagawa and Nakashima method results in an immediate termination of the algorithm.

For simulated annealing, the initial temperature was set to 0.001 and is lowered at each iteration by a factor of 10. The number of simulation runs at

each temperature was set to 1000 (refer to Cerny [15] for a detailed description of the procedure). These parameter values were selected considering the tradeoff between computing time and solution quality after several different combinations were tried.

The *BIFR* for the proposed method was determined by setting  $\Delta_j = 2 \times c_{j(1)}$  for all  $j$ . In implementing the Kohda and Inoue method, the sensitivity function with  $\alpha = 0.5$  was used.

Computational results are summarized in Table 1. When  $n = 7$ ,  $m = 1$  or 5, and  $w_j \in (1.5, 2.5)$ , exact optimal solutions were obtained by complete enumeration. Performances of each method are assessed in terms of average relative error (A), maximum relative error (M), optimality rate (O), and average execution time of ten problems (T) defined as follows.

$$A_i = \frac{1}{10} \sum_{j=1}^{10} \frac{R_j^* - R_{ij}}{R_j^*}$$

$$M_i = \max_j \left\{ \frac{R_j^* - R_{ij}}{R_j^*} \right\}$$

$O_i$  = number of times (out of 10 problems) method  $i$  yields the best system reliability,

where

$R_{ij}$  = system reliability obtained by method  $i$  for the  $j$ th test problem.  $j = 1, 2, \dots, 10$ .

$R_j^*$  = the best system reliability obtained by any of the four methods or by complete enumeration for the  $j$ th test problem.  $j = 1, 2, \dots, 10$ .



From Table 1, we observe the following.

1. In terms of solution quality (A, M, O), the performance of the proposed method is consistently better than those of others. The Kohda and Inoue method performs somewhat worse than the proposed one, while performances of the Shi method and simulated annealing are not in general comparable to those of the other two.
2. In terms of solution time (T), the Shi method requires the least amount of computing time as expected, while simulated annealing requires the most. Comparing the proposed and Kohda/Inoue methods, we observe moderate increase in computing time in the former.

## 6. Concluding remarks

Computational results indicate that the excursion of the proposed method is an effective strategy for finding 'good' solutions to complex redundancy optimization problems. The proposed method is most closely related to the Kohda and Inoue method in the sense that both employ strategies for alleviating the risk of being trapped at a local optimum. Computational results show that the former performs consistently better than the latter (this is believed to be due to the larger search region of the former than the latter), with moderate increase in computing time.

In summary, if computing time is of primary concern, then the Shi method is the one to be recommended. If, however, solution quality is of more concern and if one is willing to accept moderate increase in computing time for better solutions, then the proposed method is believed to be an attractive alternative.

Although not included in the present study, comparisons of the present method with such newly emerged search strategies as genetic algorithm, neural networks, and tabu search, etc. and finding out the possibility of combining their useful features should be fruitful areas of future research.

We believe that further elaboration of the proposed method is worthwhile. This may include investigating the possibility of reducing the computing time required by the proposed method while maintaining its ability to generate 'good' solutions by trying various schemes for generating initial feasible solutions, determining the *BIFR*, excursions, etc.

Finally, it is worth noting that the Kohda and Inoue method [7] does not require the monotonically nondecreasing property of a constraint and that the Shi method [8] deals directly with the minimal path sets rather than  $R_s$ . Incorporation of the above useful features into the present algorithm may be another fruitful area of future research.

### References

- [1] H.H. Lin, W. Kuo, "A comparison of heuristic reliability optimization methods", World Productivity Forum & 1987 International Industrial Engineering Conference Proceedings, Institute of Industrial Engineers, pp.. 583-589.
- [2] J. Sharma, K.V. Venkateswaran, "A direct method for maximizing the system reliability", IEEE Trans. Reliability, vol. R-20, 1971 Nov, pp. 256-259.
- [3] K.K. Aggarwal, J.S. Gupta, K.B. Misra, "A new heuristic criterion for solving a redundancy optimization problem", IEEE Trans. Reliability, vol. R-24, 1975 Apr, pp. 86-87.
- [4] K.K. Aggarwal, "Redundancy optimization in general systems", IEEE Trans. Reliability, vol. R-25, 1976 Dec, pp. 330-332. (Corrections, vol. R-26, 1977 Dec, pp. 345).
- [5] Y. Nakagawa, K. Nakashima, "A heuristic method for determining optimal reliability allocation", IEEE Trans. Reliability, vol. R-26, 1977 Aug, pp. 156-161.

- [6] K. Gopal, K.K. Aggarwal, J.S. Gupta, "An improved algorithm for reliability optimization", IEEE Trans. Reliability, vol. R-27, 1978 Dec, pp. 325-328.
- [7] T. Kohda, K. Inoue, "A reliability optimization method for complex systems with the criterion of local optimality", IEEE Trans. Reliability, vol. R-31, 1982 Apr, pp. 109-111.
- [8] D.H. Shi, "A new heuristic algorithm for constrained redundancy optimization in complex systems", IEEE Trans. Reliability, vol. R-36, 1987 Dec, pp. 621-623.
- [9] W. Kuo, C.L. Hwang, F.A. Tillman, "A note on heuristic methods in optimal system reliability", IEEE Trans. Reliability, vol. R-27, 1978 Dec, pp. 320-324.
- [10] Y. Nakagawa, S. Miyazaki, "An experimental comparison of the heuristic methods for solving reliability optimization problems", IEEE Trans. Reliability, vol. R-30, 1981 June, pp. 181-184.
- [11] F.A. Tillman, C.L. Hwang, W. Kuo, "Optimization of Systems Reliability", Marcel Dekker, New York, 1985.
- [12] T.J. Mitchell, "An algorithm for the construction of D-optimal experimental designs", Technometrics, vol. 16, 1974 May, pp. 203-210.
- [13] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, "Equation of state calculations by fast computing machine", J. Chem. Phys., vol. 21, 1953, pp. 1087-1092.
- [14] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "Optimization by simulated annealing", Science, vol. 220, 1983, pp. 671-680.
- [15] V. Cerny, "A thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm", J. Optimization Theory and Applications, vol 45, 1985, pp. 41-45.

Jae Hwan Kim and Bong Jin Yum

- [16] F. Glover, H.J. Greenberg, "New Approaches for Heuristic Search : A bilateral linkage with artificial intelligence", *European Journal of Operational Research*, vol. 39, 1989, pp. 119-130.
- [17] M.O. Locks, "Recursive disjoint products : A review of three algorithms", *IEEE Trans. Reliability*, vol. R-31, 1982 Apr, pp. 33-35.



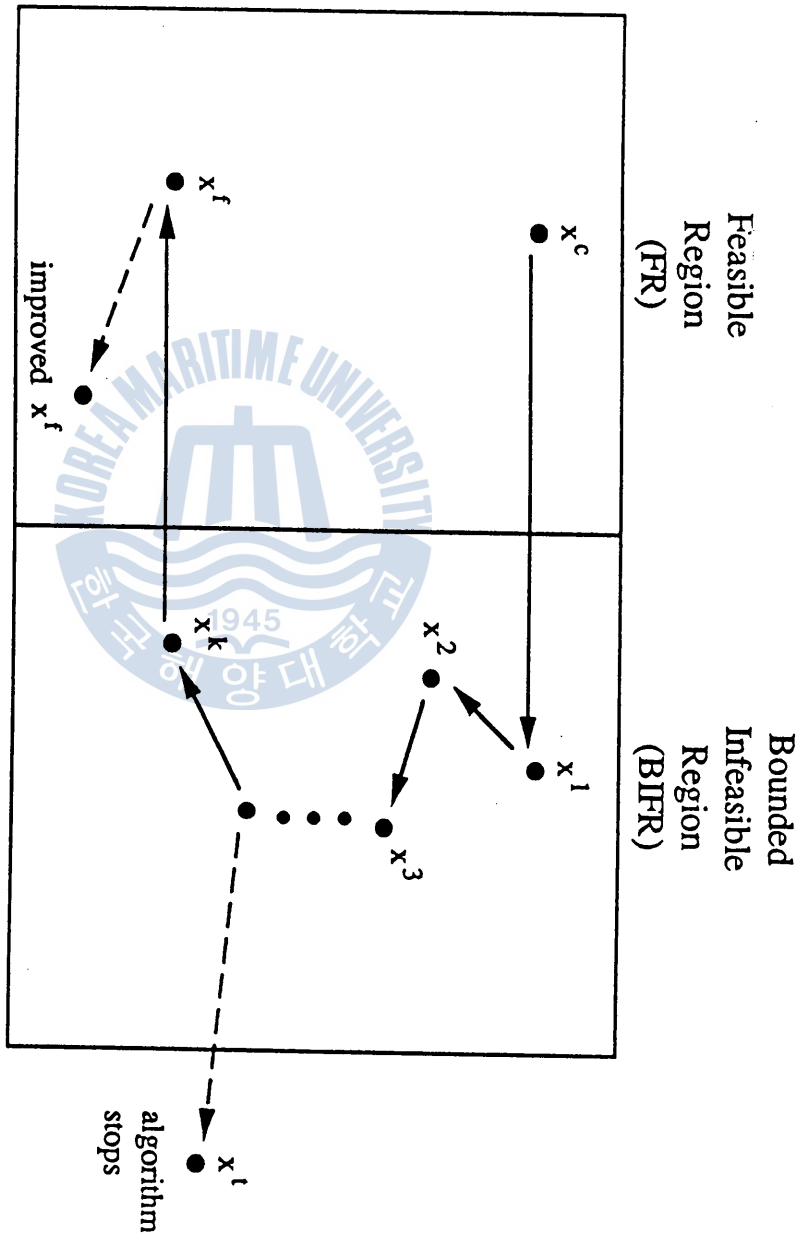


Fig. 1. An iteration of the algorithm.

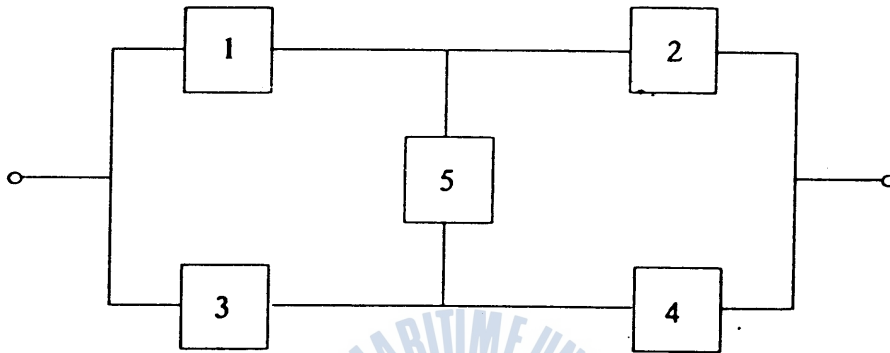


Fig. 2. A bridge network system (Example 1).

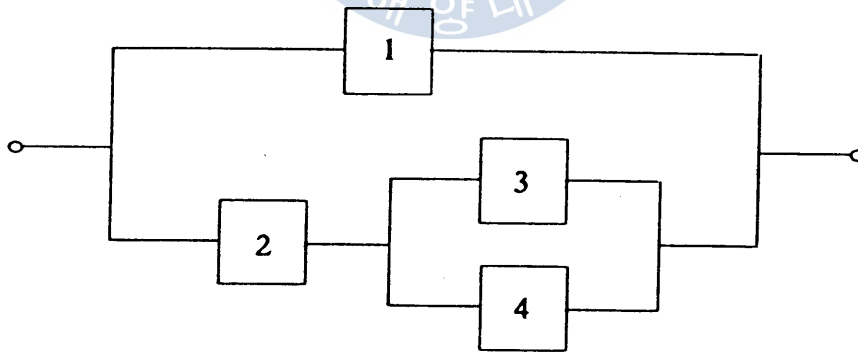


Fig. 4. A composite system (Example 2).

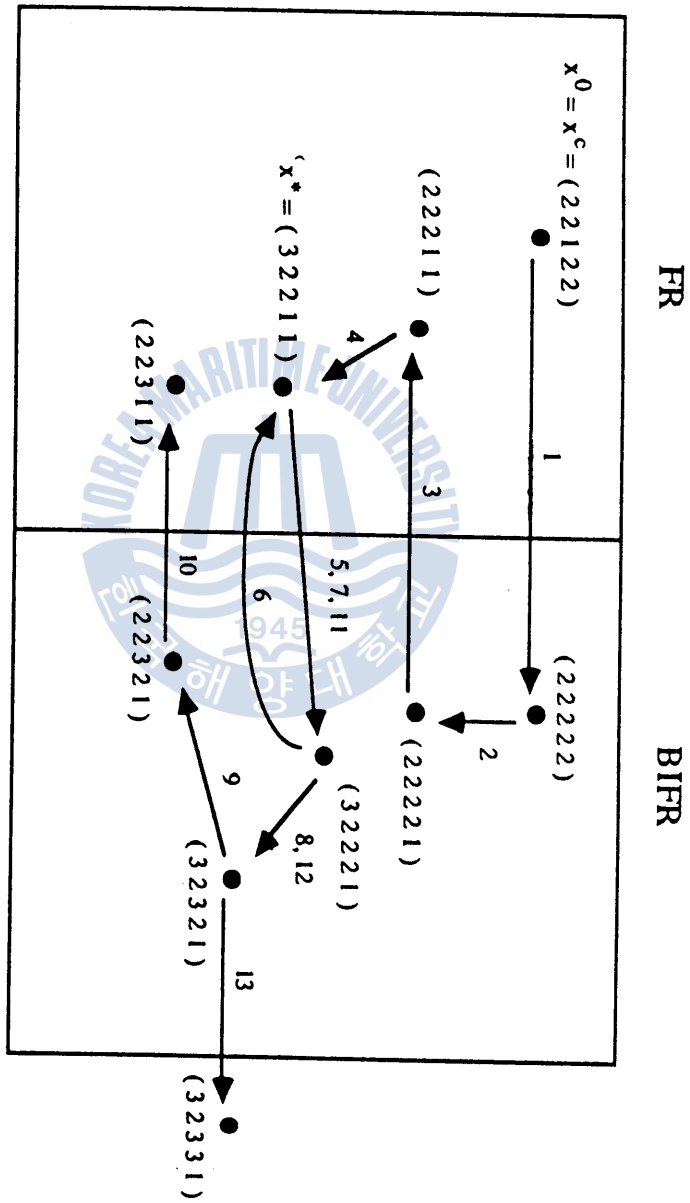
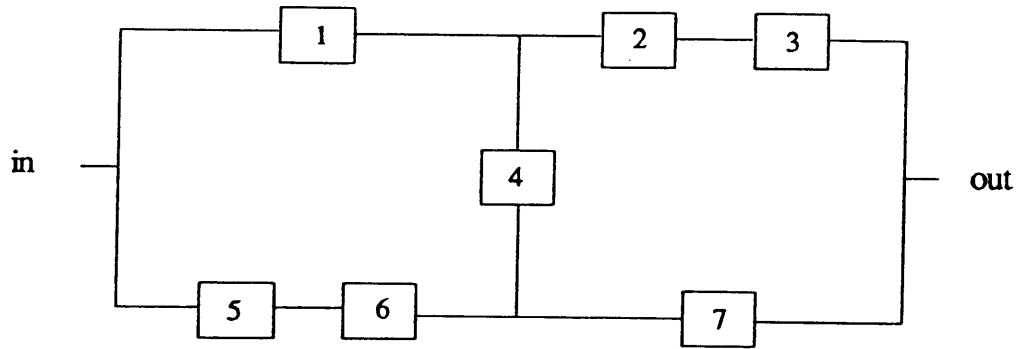
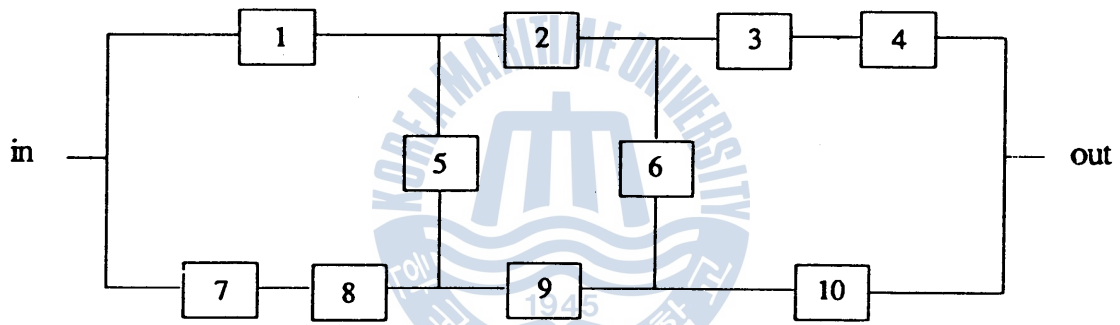


Fig. 3. Solution procedure for the example 1.

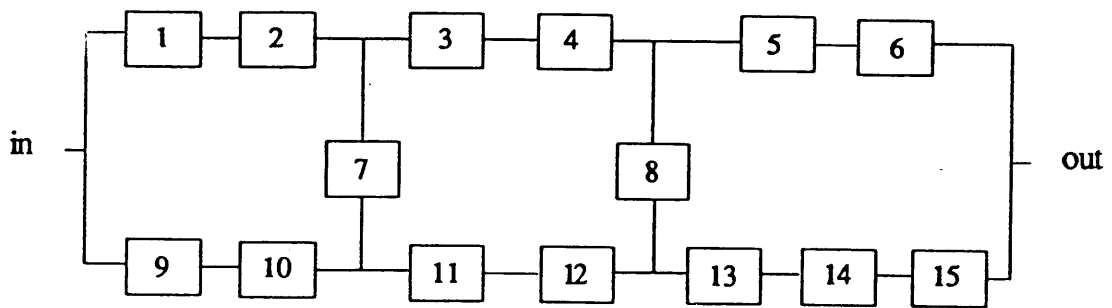




(a)  $n = 7$



(b)  $n = 10$



(c)  $n = 15$

Fig. 5. Three complex systems considered in the computational experiments.

