A 3-Neighborhood Heuristic Algorithm for Constrained Redundancy Optimization in Complex Systems

MANJU AGARWAL and SUDHANSHU AGGARWAL

Department of Operational Research, University of Delhi, Delhi-110007.

(Received on August 20, 2005)

Abstract: Several heuristic algorithms for constrained redundancy optimization in complex systems have been proposed, giving solutions that are optimal in 1-neighborhood (mostly) or 2-neighborhood. Perhaps the most interesting and efficient heuristic algorithm is that given by Agarwal and Aggarwal [9] giving solutions that are optimal in 3-neighborhood. In this paper an improved 3-neighborhood heuristic algorithm is proposed. Suitable sensitivity factors are defined to search for optimal / near optimal solution. The algorithm is tested for 8 sets of problems (with linear constraints) each with 10 randomly generated data and, 5-unit bridge structure with nonlinear constraints. Computational results illustrate its effectiveness showing an overall improvement both in solution quality and computing time. As such the heuristic proposed is attractive and can be easily and efficiently applied to numerous real life systems.

Key Words: complex system, constrained redundancy optimization, heuristic algorithm, 3-neighborhood.

1. Introduction

In general, system reliability optimization problems are nonlinear programming problems and NP-hard. They are more difficult to solve than general nonlinear programming problems, because their solutions are integers. As such computational experience with exact algorithms has not yet been completely encouraging. In fact even with Dynamic Programming it is hard to solve problems with more than three constraints. Although Integer Programming methods yield integer solutions, but transforming nonlinear objective functions and constraints into linear forms so that integer programming methods can be applied is a difficult task. In addition, the various integer programming techniques do not guarantee that optimal solutions can be obtained in a reasonable time.

A heuristic algorithm is one that is designed to provide “good” (i.e., near optimal) solution. In many real life problems any solution that gives system reliability very close to the optimal value is satisfactory. In fact, exact optimal solution has less significance when the estimation of component reliabilities and resource consumption includes errors and approximations. As such, any simple and computationally efficient heuristic method may be useful for solving large-scale reliability optimization problems. Therefore, since 1970’s major focus has been on heuristic approaches to solve such problems. However recently, a class of heuristic search strategies, known as meta-heuristics, has also emerged. These are
genetic algorithms, simulating annealing, tabu search, and neural networks. Meta-heuristics are based on artificial intelligence and useful effective optimization tools for many applications. But, are more cumbersome involving more computational effort, usually larger computer memory and longer running times. As such, any simple and computationally efficient heuristic method may be useful for solving large-scale reliability optimization problems. For these reasons, researchers in reliability optimization have placed more emphasis on heuristic approaches to find approximate solutions in a reasonable time. Kuo et al. [1] provide a good survey of all the techniques. Recently, Coit [2], Shelokar et al. [3], Ramirez-Marquez et al. [4], and Zhao and Liu [5] have solved redundancy allocation problems using various approaches.

Almost all the existing heuristic algorithms search for the optimal solution iteratively, remaining within the feasible region, in which a redundancy is added to only one subsystem in each iteration; and the selection of the subsystem is based on a sensitivity factor. Thus the solutions obtained are optimal only in 1-neighborhood. It was in 1982 that Kohda and Inoue [6] presented a heuristic algorithm with the criterion of local optimality in which two feasible solutions obtained in successive iterations are in 2-neighborhoods of each other i.e. a redundancy is added in one subsystem and a redundancy is subtracted from another or redundancy is added in two subsystems. Shi [7] developed a heuristic algorithm based on minimal path sets in which the feasible solutions obtained in successive iterations are in 1-neighborhood or 2-neighborhoods of each other. All these algorithms confined their search to feasible region only. But Kim and Yum [8] developed a heuristic algorithm for separable, monotonic non-decreasing constraint functions in which excursions are allowed over a bounded infeasible region, i.e. the search for optimal is made not only in the feasible region but also into the bounded infeasible region and is much superior to the strategy of allowing only feasible solutions. In this algorithm two feasible solutions obtained in successive iterations need not even be in 2-neighborhoods of each other. They have shown that their algorithm performs better than those of KI [6] and Shi [7], giving improvement in various measures of performance: Average relative error A, Maximum relative error M, and Optimality rate O. Perhaps the most interesting and efficient heuristic algorithm is that given by Agarwal and Aggarwal (A&A) [9], applicable to separable monotonically non-decreasing constraint functions, in which the current best feasible solution is searched in 3-neighborhood by performing a series of selection and exchange operations, to get possibly improved solution. The search for the optimal solution is made from sides of, both, feasible and infeasible regions. Further it is also shown that their algorithm, in terms of A, M and O, performs even better than [8].

The heuristic 3-neighborhood algorithm proposed in this paper, to be referred as P-Alg, is an improvement over A&A [9] algorithm. In an iteration of the P-Alg a series of selection and exchange operations are performed i.e. the current best feasible solution is examined in 3-neighborhood by subtracting (adding) a redundancy from (to) one subsystem and adding (subtracting) a redundancy to (from) one or two subsystems based on suitable sensitivity factors to see whether this exchange yields still an improved feasible solution otherwise the algorithm stops. P-Alg is tested, on complex system structures from literature, for 8 sets of problems (with linear constraints) each with 10 randomly generated data and, 5-unit bridge structure with nonlinear constraints. Computational results illustrate effectiveness of P-Alg, both, in terms of solution quality (A, M, O) and computing time (seconds) in comparison to KY [8] and A&A [9] algorithms. Thus we believe that P-Alg is attractive and an improvement over A&A [9].
2. Notation and Assumptions

Most of the notation and assumptions used in this paper are the same as used in A&A [9]. However, additional important notation is given below:

- \( X \): Vector of decision variables
- \( p_i \): Reliability of a single component of subsystem \( i \)
- \( X^c \): Current best feasible solution
- \( X^f \): Feasible solution generated by iteration
- \( X^* \): Optimal / near optimal solution in 3-neighborhood
- \( F \): Set of all solutions generated in step 2
- \( X(\pm i, \pm j) \): the notation implies “add/subtract 1 to \( X_i \), and then add/subtract 1 to \( X_j \) in the vector \( X \)”. Even if \( i = j \), the procedure is followed as stated.
- \( R_i(X), Q_i(X) \): Reliability, Unreliability of subsystem \( i \)
- \( \Delta R_i(\pm i) \): Increment / decrement in system reliability \( R_i(X) \) for increasing / decreasing \( X_i \) by 1
- \( \Delta g_j^i(\pm i) \): Increment / decrement in resource \( j \) at subsystem \( i \) for increasing / decreasing \( X_i \) by 1
- \( S(i) \): the Improvement Potential
- \( M(i) \): the Improvement Potential

3. Problem Formulation

Reliability optimization problem is formulated as:

\[
P 1: \quad \text{Maximize:} \quad R_i(X)
\]

subject to
\[
\sum_{i=1}^{n} g_i^j(X_i) \leq C^j, \quad j = 1,\ldots,m
\]

\( X_i \) are positive integers, for all \( i \).

4. Algorithm

The algorithm makes search for the optimal / near optimal solution in 3-neighborhood. It does not go for an exact search in 3-neighborhood as done in KI [6] 2-neighborhood approach, but makes a search in, both, feasible and infeasible regions by taking the best possible 3-neighborhood combinations \( X(-i, +j, +k) \) and \( X(-i, -j, +k) \). Exact 3-neighborhood search may give slight improvement in the solution quality but would lead to increase in the computational time if we explore all the other possible 3-neighborhood combinations i.e. \( X(-i, -j, -k) \) and \( X(+i, +j, +k) \), as well, leading far beyond the feasible boundary in either direction. However, the steps of the algorithms take care of majority of the points in 3-neighborhood. Some important points of the algorithm are described.
4.1 Initial Feasible Solution

An initial feasible solution is generated by the method as explained below:
Start with $X^o = (1,1,...,1)$ and $\forall i$ compute $S(i)$, the Improvement Potential w.r.t. subsystem $i$, (Hoyland and Rausand [10]). Add one redundant component to each subsystem in the decreasing order of $S(i)$ till the solution becomes infeasible. Then calculate $M(i)$ of each subsystem and choose the subsystems, which have the highest, and the next highest values of $M(i)$ and subtract one redundant component from these subsystems and then calculate $S(i)$ for each subsystem and add one redundant component to the subsystem, which has the maximum value of $S(i)$. In case of a tie choose that subsystem for which
\[ \sum_{j=1}^{m} (\Delta g^i_j / C^j) \]
is minimum. Repeat this until the solution becomes feasible. The resulting solution is taken to be the initial feasible solution $X^o$.

4.2 Selection Criterion:

1. For $X \rightarrow X(+k)$, $k$ is selected according to the following forward selection criterion:
\[
S^+_k = \max_{1 \leq i \leq m} \left[ \frac{\Delta R_x (+i)}{\sum_{j=1}^{n} (\Delta g^i_j / C^j)} \right]
\]  
   \hspace{1cm} (1)

2. For $X \rightarrow X(-k)$, $k$ is selected according to the following backward selection criterion:
\[
S^-_k = \min_{1 \leq i \leq m} \left[ \frac{\Delta R_x (-i)}{\sum_{j=1}^{n} (\Delta g^i_j / C^j)} \right]
\]  
   \hspace{1cm} (2)

3. For $X \rightarrow X(+r,+s)$, $r$, $s$ are selected according to the following forward selection criterion:
\[
S^+(r,s) = \max_{1 \leq u,v \leq m} \left[ \frac{S(u)}{\sum_{j=1}^{n} (\Delta g^{+u}_j / C^j)}\right] + \left[ \frac{S(v)}{\sum_{j=1}^{n} (\Delta g^{+v}_j / C^j)}\right] _{X(+u)}
\]  
   \hspace{1cm} (3)
4.3 Steps of the Algorithm:

1. \( F = \phi \); find initial feasible solution \( X^\alpha \). \( X^c = X^\alpha \) and compute \( R_c(X^c) \).

2. Set \( X = X(-k) \) where \( k \) is selected according to the backward selection criterion \( S^{-}_k \). If \( X \in F \) then go to step 8. Otherwise \( F = F \cup X \).

3. Set \( X = X(+r,+s) \) where \( r \) and \( s \) are selected according to the forward selection criterion \( S^{+}(r,s) \). If \( X \) is feasible, \( X^f = X \) go to step 4; otherwise go to step 6.

4. Improve \( X^f \) as much as possible according to the forward selection criterion \( S^{+} \). The (possibly improved) solution \( X^f \) is obtained and \( R_c(X^f) \) is computed.

5. If \( R_c(X^f) > R_c(X^c) \) then \( X^c = X^f \) \( X = X^c \) go to step 2. Otherwise go to step 8.

6. Subtract redundant components added in step 3 and set \( X = X(+k) \) where \( k \) is selected according to the forward selection criterion \( S^{+}_k \). If \( X \) is again infeasible go to step 7; otherwise \( X^f = X \) go to step 4.

7. Subtract one redundant component added in step 6 and also subtract one redundant component according to the backward selection criterion \( S^{-}_k \). Add one redundant component to the subsystem, which has the maximum value of \( S(i) \). In case of a tie choose that subsystem for which

\[
\sum_{j=1}^{m} \left\{ \frac{\Delta g^{f}(i)/C^{f}}{C^{f}} \right\}
\]

is minimum. If \( X \) is feasible, \( X^f = X \) go to step 4. Otherwise go to step 8.

8. \( X^c = X^c \). Stop.

5. Computational Experiments & Results

KI [6], Shi [7], KY [8], A&A [9] and P-Alg are programmed in C++ and numerical computations have been carried out on a P4-2.40 GHz computer.

Example 1

First we apply P-Alg to the 5-unit Bridge system [6, 7, 8, 9] with nonlinear constraints:

Problem:

Maximize \( R_c(X) \)

subject to

\[
g_1(X) = \sum_{i=1}^{5} c_i X_i^2 \leq 100
\]

\[
g_2(X) = \sum_{i=1}^{5} d_i [X_i + \exp\left(\frac{X_i}{4}\right)] \leq 175
\]
$$g(X) = \sum_{i=1}^{5} w_i X_i \exp\left(\frac{X_i}{4}\right) \leq 200$$

$X_i$, $i = 1, 2, ..., 5$ are positive integers.

The subsystem data is:

<table>
<thead>
<tr>
<th>$p_i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_i$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$d_i$</td>
<td>0.7</td>
<td>7</td>
<td>15</td>
<td>9</td>
<td>0.4</td>
</tr>
<tr>
<td>$w_i$</td>
<td>0.7</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Solution:

1. $F = \emptyset$; $X^0 = (3,3,2,3,3)$; $X^c = X^0$; $R_c(X^c) = 0.998285$; $X = X^c$.

2. we go through $(3,3,2,3,3) \rightarrow (3,3,2,3,3)$; $X = (3,3,2,3,3)$; $X \not\in F$; $F = \{(3,3,2,3,3)\}$.

3. we go through $(3,3,2,3,3) \rightarrow (5,3,2,3,2)$; $X = (5,3,2,3,2)$ is feasible; $X^f = X$.

4. $X^f$ cannot be improved further; $X^f = (5,3,2,3,2)$; $R_c(X^f) = 0.999819$.

5. $R_c(X^f) > R_c(X^c)$; $X^c = X^f = (5,3,2,3,2)$; $X = X^c$.

2. we go through $(5,3,2,3,2) \rightarrow (5,3,2,3,2)$; $X = (5,3,2,3,2)$; $X \not\in F$; $F = \{(5,3,2,3,2)\}$.

3. we go through $(5,3,2,3,2) \rightarrow (7,3,2,3,1)$; $X = (7,3,2,3,1)$ is infeasible.

6. we go through $(7,3,2,3,1) \rightarrow (5,3,2,3,2)$; $X = (5,3,2,3,2)$ is infeasible.

7. we go through $(5,3,2,3,1) \rightarrow (5,3,3,2,1)$; $X = (5,3,3,2,1)$ is feasible; $X^f = X$.

4. $X^f$ can be improved to $X^f = (6,3,3,2,1)$; $R_c(X^f) = 0.999846$.

5. $R_c(X^f) > R_c(X^c)$; $X^c = X^f = (6,3,3,2,1)$; $X = X^c$.

2. we go through $(6,3,3,2,1) \rightarrow (6,3,3,2,1)$; $X = (6,3,3,2,1)$; $X \not\in F$; $F = \{(3,3,2,3,2),(5,3,2,3,2)(6,3,2,3,1)\}$.

3. we go through $(6,3,2,3,1) \rightarrow (7,4,2,2,1)$; $X = (7,4,2,2,1)$ is infeasible.

6. we go through $(7,4,2,2,1) \rightarrow (6,4,2,2,1)$; $X = (6,4,2,2,1)$ is feasible; $X^f = X$.

4. $X^f$ cannot be improved further; $X^f = (6,4,2,2,1)$; $R_c(X^f) = 0.999928$.

5. $R_c(X^f) > R_c(X^c)$; $X^c = X^f = (6,4,2,2,1)$; $X = X^c$.

2. we go through $(6,4,2,2,1) \rightarrow (6,3,3,2,1)$; $X = (6,3,3,2,1)$; $X \in F$.

8. $X^* = (6,4,2,2,1)$ is optimal in 3- neighborhood with $R_c(X^*) = 0.999928$. Stop.

The solution obtained by P-Alg is: $X^* = (6,4,2,2,1)$ with $R_c(X^*) = 0.999928$. By using KI [6], Shi [7], KY [8] and A&A [9] algorithms, the solutions $X^* = (1,2,4,3,1)$ with $R_c(X^*) = 0.998409$; $X^* = (7,3,2,2,1)$ with $R_c(X^*) = 0.999830$; $X^* = (4,4,3,2,2)$
with \( R_j(X^*) = 0.999850 \); and \( X^* = (6,4,2,2,1) \) with \( R_j(X^*) = 0.999928 \) respectively. It can be observed that P-Alg gives better solution than KI [6], Shi [7] and KY [8].

To understand the merit of the heuristic developed in this paper at depth, more numerical computations for complex systems comprising of 5 (bridge system), 7, 10 and 15 (KY [8]) have been carried out.

The problems in the computation have the same structure as \( P 1 \) except that the constraints are replaced by:

\[
\sum_{i=1}^{n} c_{ij} x_i \leq b_j , \quad j = 1,2,...,m .
\]

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

- \( n = 5, 7, 10, 15 \),
- \( m = 1, 5 \).

This results in 8 sets of problems. For each set, 10 problems were generated randomly:

- \( \{c_{ij}\} \) vary from (0, 100),
- \( \{p_i\} \) vary from (0.6, 0.95),
- \( \{b_j\} \) vary from (20, 100) with \( m = 1 \) and (50, 1000) with \( m = 5 \).

For each complex system, an expression for the system reliability is obtained in compact form by Abraham’s [11] algorithm.

The performance of the P-Alg with those of KI [6], Shi [7], KY [8] and A&A [9] is assessed in terms of A, M, O and average execution time \( T \) (in seconds) of the 10 problems for each system defined as follows:

\[
A_i = \text{Average Relative Error for algorithm } i = \frac{1}{10} \sum_{j=1}^{10} \frac{(R_j^* - R_{ij})}{R_j^*} .
\]

\[
M_i = \text{Maximum Relative Error for algorithm } i = \max_{1 \leq j \leq 10} \left\{ \frac{(R_j^* - R_{ij})}{R_j^*} \right\} .
\]

\[
O_i = \text{Optimality Rate for algorithm } i = \frac{\text{Number of times (out of 10 problems) algorithm } i \text{ yields the best system reliability}}{10} .
\]

\[
R_j^* = \text{System reliability obtained by algorithm } i \text{ for test problem } j ; \quad j = 1,2,...,10 .
\]

\[
R_{ij} = \text{The best system reliability obtained by any of the five algorithms; } \quad j = 1,2,...,10 .
\]

Computational results are summarized in Table 1.

Table 1 shows the result of the set of 80 problems solved with P-Alg, KI [6], Shi [7], KY [8] and A&A [9] algorithms. We analyze the results in terms of different measures of performance (A, M, O) and accordingly rank the algorithms. The ranking results are summarized in Table 2 showing the number of times an algorithm attains a particular rank.
Table 1: Comparison of Performance Measures

<table>
<thead>
<tr>
<th></th>
<th>KI Algorithm</th>
<th>Shi Algorithm</th>
<th>KY Algorithm</th>
<th>A&amp;A Algorithm</th>
<th>P- Alg</th>
</tr>
</thead>
<tbody>
<tr>
<td>5×1 A</td>
<td>.00215809</td>
<td>.00516149</td>
<td>.000020944</td>
<td>.00056913</td>
<td>.00009453</td>
</tr>
<tr>
<td>M</td>
<td>.01199642</td>
<td>.01376466</td>
<td>.0084927</td>
<td>.00504572</td>
<td>.0045523</td>
</tr>
<tr>
<td>O 5/10</td>
<td>.0021</td>
<td>.00516149</td>
<td>.000020944</td>
<td>.00056913</td>
<td>.00009453</td>
</tr>
</tbody>
</table>

| 5×5 A  | .00042488    | .01223261     | .00138358    | .0021534      | 0       |
| M      | .00215336    | .04757588     | .00979920    | .00215336     | 0       |
| O 8/10 | .00021      | .00516149     | .000020944   | .00056913     | 10/10   |

| 7×1 A  | .00372716    | .01773230     | .00106185    | .0051748      | .00016614|
| M 5/10 | .01441205    | .08235576     | .00393749    | .00437922     | .0016614|
| O      | .00042488    | .00516149     | .000020944   | .00056913     | 8/10    |

| 7×5 A  | .00065200    | .00714387     | .00467225    | .00163072     | .00112579|
| M 8/10 | .00638063    | .02610405     | .03306289    | .00780457     | .00780457|
| O 5/10 | .01094514    | .09999079     | .06991234    | .00268564     | .00259756|

| 10×1 A | .00090854    | .06065576     | .01530724    | .0090854      | .0042863 |
| M      | .00908536    | .12334378     | .12614152    | .00908536     | .00355276|
| O 9/10 | .01094514    | .14974849     | .02682564    | .10/10        | .01726283|

| 10×5 A | .00487676    | .01748386     | .00419218    | .00350737     | .00808505 |
| M      | .01959678    | .03431030     | .01995559    | .01959678     | .02465934|
| O 5/10 | .00487676    | .01748386     | .00419218    | .00350737     | .00808505 |

| 15×1 A | .00124742    | .08914075     | .07390590    | .00124742     | .00479539 |
| M      | .01247420    | .14108760     | .15860703    | .00124742     | .02105269|
| O 9/10 | .00124742    | .14108760     | .15860703    | .00124742     | .00479539 |

| 15×5 A | .00124742    | .08914075     | .07390590    | .00124742     | .00479539 |
| M      | .01247420    | .14108760     | .15860703    | .00124742     | .02105269|
| O 9/10 | .00124742    | .14108760     | .15860703    | .00124742     | .00479539 |

Table 2. Ranking of Algorithm s

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The effectiveness of the P-Alg in terms of computing time (sec.) is shown in Table 3.

Table 3. Comparison of Computing Times

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5×1</td>
<td>0.58</td>
<td>2.24</td>
<td>0.96</td>
<td>0.84</td>
</tr>
<tr>
<td>5×5</td>
<td>0.70</td>
<td>2.15</td>
<td>0.98</td>
<td>0.88</td>
</tr>
<tr>
<td>7×1</td>
<td>1.01</td>
<td>2.51</td>
<td>1.42</td>
<td>1.21</td>
</tr>
<tr>
<td>7×5</td>
<td>1.79</td>
<td>2.74</td>
<td>1.86</td>
<td>1.87</td>
</tr>
<tr>
<td>10×1</td>
<td>1.12</td>
<td>3.64</td>
<td>1.28</td>
<td>1.13</td>
</tr>
<tr>
<td>10×5</td>
<td>2.47</td>
<td>5.02</td>
<td>2.30</td>
<td>2.07</td>
</tr>
<tr>
<td>15×1</td>
<td>4.22</td>
<td>9.27</td>
<td>4.96</td>
<td>3.92</td>
</tr>
<tr>
<td>15×5</td>
<td>2.64</td>
<td>1.38</td>
<td>2.55</td>
<td>2.33</td>
</tr>
</tbody>
</table>
It can be observed that P-Alg is much faster than KY [8] and A&A [9]. KY [8] takes on an average 138% and A&A [9] takes on an average 15% more time than P-Alg. However, for 15×5 system computational time for KY [8] is less than for P-Alg which is due to its very low optimality rate i.e. 1/10. Though in comparison to KI [6] P-Alg takes slightly more time as expected, on an average 8% more, since it searches a wider range of solution area involving more computational steps, but, interestingly for larger systems 10×5, 15×1 and 15×5, it is faster. From all the above examples, set of 80 problems solved, it can be concluded that taking into account both, the solution quality (A, M, O) as well as computational time (secs), P-Alg is preferable for all system structures.

6. Conclusion

P-Alg appears to be very efficient in solving constrained redundancy optimization problems and is an improvement over A&A [9]. The heuristic proposed can be applied to numerous real life systems such as computer and communication systems, automobile, nuclear and defense systems etc. giving optimal or near optimal solution.

Acknowledgement

The authors are grateful to the referees for their useful comments and suggestions. Sudhanshu Aggarwal gratefully acknowledges the financial support received from the Council of Scientific and Industrial Research, New Delhi, India.

References


**Manju Agarwal** received her B.A. (Hons.) in Maths, M.A. (Operational Research), Ph.D. (Operational Research) in 1964, 1966 and 1981, respectively from University of Delhi, India. She visited Department of Industrial Engineering, University of Toronto and Department of Mathematics, Royal Military College of Canada, Kingston, Ontario, Canada during 1989-1991 and, Department of Mathematics and Statistics, Curtin University of Technology, Perth, Australia during May-June 2004. Besides, she has participated in International Conferences at USA, Canada, Japan, Greece, Austria, and Poland. Her present research interests include System Reliability Optimization, Multi-state System Reliability, Lattice Path Combinatorics in Queuing Theory and, Distribution of Runs and Patterns. She has published more than 65 papers in journals of International repute.

**Sudhanshu Aggarwal** received his B.Sc. (G), M.Sc. (Operational Research), Ph. D. (Operational Research) degrees in 1998, 2000 and 2006, respectively from University of Delhi, India. He visited Poland in 2005 for participating in the European Safety and Reliability Conference ESREL 2005. Presently he is working as Assistant Executive Secretary-III (P) in Indian National Science Academy, Bahadur Shah Zafar Marg, New Delhi-110002. His research area is System Reliability Optimization.