International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 3, 185–212, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.3-016

A Genetic Algorithm Based Hybrid Approach for Reliability-Redundancy Optimization Problem of a Series System with Multiple-Choice

Asoke Kumar Bhunia
Department of Mathematics
The University of Burdwan, Burdwan-713104, WB, India
Email: bhuniaak@rediffmail.com

Avijit Duary
Department of Mathematics
Sir J. C. Bose School of Engineering, SKFGI, Hoogly -712139, WB, India
Email: avijitduary.math@yahoo.com

Laxminarayan Sahoo
Department of Mathematics
Raniganj Girls' College, Raniganj-713358, West Bengal, India
Corresponding Author: lxsahoo@gmail.com

(Received December 10, 2016; Accepted February 26, 2017)

Abstract
The goal of this paper is to introduce an application of hybrid algorithm in reliability optimization problems for a series system with parallel redundancy and multiple choice constraints to maximize the system reliability subject to system budget and also to minimize the system cost subject to minimum level of system reliability. Both the problems are solved by using penalty function technique for dealing with the constraints and hybrid algorithm. In this algorithm, the well-known real coded Genetic Algorithm is combined with Self-Organizing Migrating Algorithm. As special cases, both the problems are formulated and solved considering single component without redundancy. Finally, the proposed approach is illustrated by some numerical examples and the computational results are discussed.

Keywords: Reliability-redundancy optimization, Multiple-choice constraints, Constrained integer nonlinear optimization, Genetic algorithm, Self-organizing migrating algorithm.

1. Introduction
In the present highly competitive business scenario, the reliability of a system (including industrial system) is widely regarded as an extremely important and crucial design measure. Consequently, the techniques / theories for the enhancement of system reliability play a pivotal role in the growth, development and improvement of power systems, telecommunication systems, manufacturing systems (Nourelfath and Nahas, 2003), advanced semiconductors, memory integrated circuits and nano systems (Ha and Kuo, 2006). The introduction of redundancy allocation is a commonly accepted technique, which is well known for its effectiveness in improving the reliability of a system. The problem associated with this method is known as redundancy allocation problem. The choice of the optimal combination of components of a system during design phase is guided by several factors like cost, performance, weight, size, technology, etc. Over the last few decades, a large number of researchers has explored this field of research. In this connection, one may refer to the works of Ghare and Taylor (1969), Tillman et al. (1977, 1980), Nakagawa et al. (1978), Chern (1992), Kuo (2001), Sun and Li (2002), Ha and Kuo (2006), Gupta et al. (2009) and others.

Due to the development of advanced technology and competitive market situations, for each component of a reliability system, various technologies are available. These technologies differ among themselves in terms of cost and reliability. This type of problem is known as reliability optimization problem with multiple choice constraints. In this area, the works of Nauss (1978), Sinha and Zoltners (1979), Sung and Lee (1994), Sung and Cho (2000), Nourelfath and Nahas (2003), Nahas and Nourelfath (2005) and others are worth mentioning. In their works, they did not consider the redundancy for each component. However, redundancies play an important role in reliability system. Mainly, it is used to increase the system reliability. On the other hand, they did not consider the cost optimization problem.

Genetic Algorithm (GA) is a very efficient and powerful heuristic search optimization method based on the mechanics of natural genetics and natural selection which mimics the Charles Darwin's evolutionary principle "Survival of the fittest". Prof. J. H. Holland (1975) first developed the concept of this algorithm. Thereafter, many works have been done for the development of this subject. The detailed works on the development of this subject are presented in the Goldberg (1989), Michalewicz (1996), Sakawa and Kato (2002) and others.

The focuses of the research on the hybridization have received significant interest in the recent years to solve the real-world problems (see Refs. Renders and Flasse, 1996; Salhi and Queen, 2004; Fan, et al., 2006; Pedamallu and Ozdamar, 2008). GA works very efficiently when it is combined with other algorithms or local search methods, rather than simple Genetic Algorithm (Chelouah and Siarry, 2003). To enhance the efficiency of GA, most of the researchers have proposed hybrid algorithms combining GA and various other algorithms.

Recently, a new stochastic optimization algorithm, viz. Self-Organizing Migrating Algorithm (SOMA) has been developed by Zelinka and Lampinen (2000). This is a population based stochastic search algorithm depending on the self-organizing behaviour of group of individuals in a social environment. Like Evolutionary Algorithm, it works with a population of individuals (in optimization, we refer to each solution as an individual). In SOMA, the individuals change their positions during migration loop (iteration). This change is enticed to the direction of the best individual from other individuals in a random fashion. This algorithm was seldom used in solving optimization problems. In this connection, one may refer to the recent works of Zelinka (2004), Nolle et al. (2005), Coelho (2009), Coelho and Alotto (2009), Coelho and Mariani (2010), Senkerik et al. (2010) and others.

This paper deals with a series system having several subsystems (with parallel redundancies) for each of which various technologies with different costs and reliabilities are available. For this system, two problems have been formulated and solved. In the first problem, system reliability is maximized subject to the budget constraint. On the other hand, in the second problem, system cost is minimized subject to a minimum level of system reliability. In both cases, problems are formulated as nonlinear integer programming problems and solved by using penalty function technique and a hybrid algorithm developed by combining real coded Genetic Algorithm and Self-Organizing Migrating Algorithm. As special cases, both the problems have been formulated and solved considering single component without redundancy. Finally, to illustrate the proposed approach, some numerical examples have been solved and the computational results have been discussed.

2. Nomenclature

n	number of subsystems of the main series system
M_j	number of technologies available for the subsystem j
m_{ij}	number of redundant components arranged in parallel in the subsystem j, when technology i is adopted (i.e., the number of redundancies provided by technology i for subsystem j)
r_{ij}	reliability of each component arranged in parallel in the technology i for subsystem j (r_{ij} ’s is assumed to be known)
c_{ij}	cost of each component arranged in parallel in the technology i for subsystem j (c_{ij} ’s is assumed to be known)
$f(.)$	objective function
R_s	system reliability
C_s	total cost of the system
$R_{\min}$	minimum desirable reliability of the overall system
$C_{\max}$	maximum permissible budget for the overall system
y_{ij}	decision variable (with $j = 1, 2, \cdots, n$ and $i = 1, 2, \cdots, M_j$) such that $y_{ij} = \begin{cases} 1 & \text{when technology } i \text{ is used in subsystem } j \\ 0 & \text{otherwise} \end{cases}$
F	feasible space
t	current iteration (or generation or migration loop)
$v_i^{(t)}$	i^{th} solution vector (chromosome) at t^{th} iteration [i.e., $v_i^{(t)} = (v_{i1}^{(t)}, v_{i2}^{(t)}, ..., v_{ik}^{(t)}, ..., v_{in}^{(t)})$]
$v_{ij}^{(t+1)}$	new value of j^{th} component for i^{th} active solution in $(t+1)^{th}$ iteration with step size
$v_{ij,\text{ start}}^{(t)}$	starting position of j^{th} component for i^{th} active solution in t^{th} iteration
$v_{Lj}^{(t)}$	position of j^{th} component of Leader in t^{th} iteration
$P(t)$	population of solution vectors (chromosomes) at t^{th} iteration
Pop_{size}	population size (i.e., total number of solution vector in a population)
T	maximum number of allowable iterations (generations)
p_{cross}	probability of crossover
p_{mute}	probability of mutation
StDev	standard deviation

3. Mathematical Formulation of Reliability–Redundancy Optimization Problem

The goal of the reliability–redundancy optimization problem is to determine an optimal redundancy allocation so as to maximize the overall system reliability under limited resource constraints. The reliability–redundancy optimizations are useful for system designs that are largely assembled and manufactured using off-the-shelf components and also have high reliability requirements (Coelho, 2009).

A well-known series system with n - independent subsystems has been considered. As depicted in Fig. 1, there are different technologies available for each of these n-subsystems. Each technology, when used for a subsystem, employs its own components arranged parallel with one another to form the subsystems. When the same technology used for a given subsystem, the components are identical in terms of cost and reliability. However, the component's reliability and cost may vary for different subsystems when the technology is given. Also, the component's reliability may vary for different technologies, when the subsystem is given. For each subsystem, only one technology can be adopted.

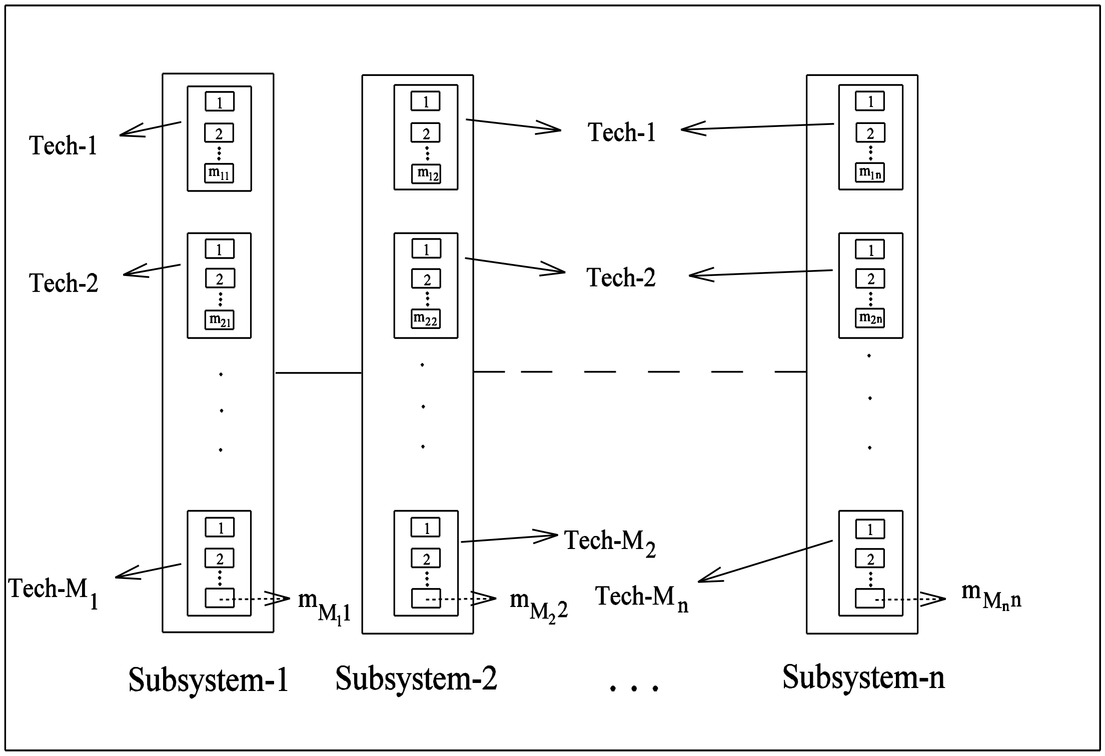

Fig. 1. Series system with n-subsystems

From the given above situation, the following two decision making problems may arise:

(i) To study and select the best combination of technologies along with the optimum number of redundant components in each subsystem, so that the system reliability is maximized subject to a budget constraint.

(ii) To study and select the best combination of technologies along with the optimum number of redundant components in each subsystem, so that the total cost is minimized subject to a given fixed level of system reliability.

With the help of earlier mentioned notations, the mathematical formulations of two decision-making problems (i) and (ii) discussed above are as follows:

Problem 1

$$\text{Maximize } R_s = \prod_{j=1}^{n} \sum_{i=1}^{M_j} y_{ij} \left\{1-\left(1-r_{ij}\right)^{m_{ij}}\right\} \tag{1}$$

subject to $\displaystyle\sum_{j=1}^{n} \sum_{i=1}^{M_j} y_{ij}\, c_{ij}\, m_{ij} \le C_{\max}$

where $\displaystyle\sum_{i=1}^{M_j} y_{ij} = 1 \quad \forall\ j = 1, 2, \cdots, n\ ;\ y_{ij} = \{0, 1\};\ \forall i = 1, 2, \cdots, M_j$ and $j = 1, 2, \cdots, n$.

Problem 2

$$\text{Minimize } C_s = \sum_{j=1}^{n} \sum_{i=1}^{M_j} y_{ij}\, c_{ij}\, m_{ij} \tag{2}$$

subject to $\displaystyle\prod_{j=1}^{n} \sum_{i=1}^{M_j} y_{ij} \left\{1-\left(1-r_{ij}\right)^{m_{ij}}\right\} \ge R_{\min}$

where $\displaystyle\sum_{i=1}^{M_j} y_{ij} = 1 \quad \forall\ j = 1, 2, \cdots, n\ ;\ y_{ij} = \{0, 1\}\ \ \forall i = 1, 2, \cdots, M_j$ and $j = 1, 2, \cdots, n$.

These two problems (1) and (2) belong to the category of constrained integer nonlinear optimization problems.

4. Constraint Handling Technique of Constrained Integer Nonlinear Optimization Problems

In the application of evolutionary algorithm for the given constrained integer nonlinear optimization problem, there arises an important question: how the algorithm handles the constraints relating to the optimization problem? During the last few decades, several methods have been proposed to handle the constraints for solving constrained optimization problems with the help of evolutionary algorithms (Michalewicz and Schoenauer, 1996; Koziel and Michalewicz, 1999; Deb, 2000; Coello, 2002). Among these methods, penalty function method is very popular. In this method, the constrained optimization problem is converted to unconstrained one in which the reduced objective function involves the original objective function and a penalty for violating the constraints. Recently Gupta et al. (2009) proposed a penalty function approach to handle the constraints. In this approach, to convert the constrained optimization problem to an unconstrained one, a large negative value (say, –M) is blindly assigned to the objective function for the infeasible solution (for maximization problem). In this case, if the constrained optimization problem is

Maximize $f(\mathbf{x})$
subject to $g_i(\mathbf{x}) \le 0,\ i = 1, 2, ..., m$

then the reduced unconstrained optimization problem is as follows:

Maximize $\hat{f}(\mathbf{x}) = f(\mathbf{x}) + \delta(\mathbf{x})$

where $\delta(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in F \\ \text{-M}, & \text{if } \mathbf{x} \notin F \end{cases}$

and $F = \{\mathbf{x}; g_i(\mathbf{x}) \le 0,\ i = 1, 2, ..., m\}$, the feasible space for the optimization problem.

For minimization problem, it is to be noted that instead of –M, +M is considered. For solving the earlier mentioned constrained integer nonlinear optimization problems (1) and (2), we have proposed a new hybrid algorithm, viz. C-RCSOMGA which is discussed in the next section.

5. Hybrid Algorithm C-RCSOMGA, Based on RCGA and SOMA

For the purpose of solving the reliability–redundancy optimization problems (1) and (2), we have developed a hybrid Real Coded Self-Organizing Migrating Genetic Algorithm (C-RCSOMGA) combining two different algorithms RCGA and SOMA. In this algorithm, we have used tournament selection, modified uniform crossover and modified mutation operators for RCGA as well as our proposed modified strategy for SOMA.

5.1 Real Coded Genetic Algorithm (RCGA)

Genetic Algorithm is a stochastic search method based on natural evolution and natural genetics. In this algorithm, initially a population of solutions is generated by Random Number Generator (RNG). Then this population is updated from iteration (generation) to iteration with respect to their fitness value through different well known genetic operators (viz. selection, crossover and mutation) until a termination criterion is satisfied. For implementation of this GA for the proposed algorithm, the following basic components are considered:

(a) Initialization of GA parameters and bounds of variables.
(b) Representation and initialization of solution.
(c) Evaluation of the fitness function.
(d) Selection process.
(e) Genetic operators (crossover, mutation to create the new offspring for improvement of the population).
(f) Termination criterion.

5.1.1 Initialization of GA Parameters and Bounds of Variables

Genetic Algorithm is dependent on some parameters, viz. population size (Pop_{size}), maximum number of allowable iterations (T), probability of crossover (p_{cross}) and probability of mutation (p_{mute}). But there is no hard and fast rule to choose the values of all parameters. From the literature (Goldberg et al., 1989; Michalewicz and Schoenauer, 1996; Jabeen and Bhunia, 2006), it is seen that, if the values of the parameters are not chosen in the reasonable range, there arise some difficulties. If the value of Pop_{size} is taken very large then the computational cost is large and also storing of data in computer in intermediate steps of GA may arise some difficulties at the time of computation. Again, if the value of Pop_{size} is taken very small then the good properties of genetic operators do not stem for evolution of the population. T varies from problem to problem and it depends upon the number of variables of the problem. From the natural genetics, it is obvious that the value of p_{cross} is always greater than that of p_{mute}.

5.1.2 Representation and Initialization of Solution

After initialization of GA parameters and bounds of variables, a successful implementation of GA is dependent on the representation of solution and also the initialization of an appropriate population. In this work, we have used the real numbers to represent the component of solution.

5.1.3 Evaluation of the Fitness Function

After getting a population of solutions, GA carries out the *evaluation* stage. In this stage, we have to evaluate the objective function value of each solution of initial population or improved population. In this work, we have considered the objective function value as fitness value of the solution.

5.1.4 Selection Process

In the *selection process*, we have handled the constraints of an optimization problem by the tournament selection method (Brindle, 1981; Goldberg et al., 1989). In this method, two or more solutions are chosen randomly from the population and the best solution from this group is selected as parent for the next iteration. This process is repeated Pop_{size} number of times. The size of the tournament selection may take the value from 2 to Pop_{size}. In our work, we have taken the size as 2. In t^{th} iteration, if two solutions $v_i^{(t)}$ and $v_j^{(t)}$ are considered for tournament selection, then any of these two will be selected for the next (i.e., $(t+1)^{th}$) iteration based on the following rules:

(a) If $f(v_i^{(t)})$ is better than $f(v_j^{(t)})$ [$f(v_i^{(t)}) > f(v_j^{(t)})$ in case of maximization problem, whereas $f(v_i^{(t)}) < f(v_j^{(t)})$ in case of minimization problem], where $v_i^{(t)}, v_j^{(t)} \in F$ then select $v_i^{(t)}$ otherwise select $v_j^{(t)}$.

(b) If $v_i^{(t)} \in F$ and $v_j^{(t)} \notin F$, then select $v_i^{(t)}$, otherwise if $v_i^{(t)} \notin F$ and $v_j^{(t)} \in F$, then select $v_j^{(t)}$.

(c) If $v_i^{(t)}, v_j^{(t)} \notin F$, then select the solution with lesser number of constraints violation.

(d) If $v_i^{(t)}, v_j^{(t)} \notin F$ and both the solution have equal constraints violation, then select any one of them.

5.1.5 Crossover

After selection process, the survived solutions take part in the *crossover operation*. The main objective of this operation is to create new offspring (probably better) by recombining the features of randomly selected two or more parent solutions. The crossover of two parents is inspired by the natural genetic process. In this work, we have used modified uniform crossover operation (Sahoo et al., 2012). Expected $[Pop_{size} * p_{cross}]$ (say, N_{cross}) (* denotes the product and denotes the integral value) number of solutions will take part in this operation. Here the uniform crossover operation has been used. The different steps of this operation at t^{th} generation are as follows:

Step-1: Find the integral value of $Pop_{size} * p_{cross}$ and store it in N_{cross}.

Step-2: Select two chromosomes $v_i^{(t)}$ and $v_j^{(t)}$ randomly from the population.

Step-3: Compute the components $\hat{v}_{ik}^{(t)}$ and $\hat{v}_{jk}^{(t)}$ ($k = 1, 2, ..., n$) of two offspring $\hat{v}_i^{(t)}$ and $\hat{v}_j^{(t)}$ by

either $\hat{v}_{ik}^{(t)} = v_{ik}^{(t)} - g$ and $\hat{v}_{jk}^{(t)} = v_{jk}^{(t)} + g$, if $v_{ik}^{(t)} > v_{jk}^{(t)}$

or, $\hat{v}_{ik}^{(t)} = v_{ik}^{(t)} + g$ and $\hat{v}_{jk}^{(t)} = v_{jk}^{(t)} - g$, otherwise

where g is a random integer number 0 and $| v_{ik}^{(t)} - v_{jk}^{(t)} |, k = 1, 2, ..., n$.

Step-4: Compute $v_i^{(t+1)}$ = argument of best of $\left\{ f(v_i^{(t)}),\ f(v_j^{(t)}),\ f(\hat{v}_i^{(t)}),\ f(\hat{v}_j^{(t)}) \right\}$;

and $v_j^{(t+1)}$ = argument of next best of $\left\{ f(v_i^{(t)}),\ f(v_j^{(t)}),\ f(\hat{v}_i^{(t)}),\ f(\hat{v}_j^{(t)}) \right\}$.

Step-5: Repeat Step-2 to Step-4 for $\frac{1}{2} N_{cross}$ times.

5.1.6 Mutation

From inspiration of genetic diversity in nature, in GA, *mutation operation* is performed to introduce the random variations into the population. Sometimes, it helps to get back the information lost in earlier iterations. Expected $[Pop_{size} * n * p_{mute}]$ (say, N_{mute}) (* denotes the product and denotes the integral value) number of genes / components will take part in mutation operation. According to Michalewicz (1996), mutation is also applied to whole solution vector rather than a single component of it. Basically, it is responsible for fine tuning capabilities of the system. In this work, we have used one–neighborhood mutation (Bhunia et al., 2010).
The computational steps of this operation at t^{th} generation are as follows:

Step-1: Find the integral value of the product of Pop_{size}, n and p_{mute} and store it in N_{mute}.

Step-2: Select a non-mutated gene $v_{ik}^{(t)}$ of solution $v_i^{(t)}$ for mutation.

Step-3: Create new gene $\hat{v}_{ik}^{(t)}$ of the new solution $\hat{v}_i^{(t)}$ by the following process as follows:

$$\hat{v}_{ik}^{(t)} = \begin{cases} v_{ik}^{(t)} + 1, \ if\ v_{ik}^{(t)} = l_{ik} \\ v_{ik}^{(t)} - 1, \ if\ v_{ik}^{(t)} = u_{ik} \\ v_{ik}^{(t)} + 1, \ if\ r < 0.5 \\ v_{ik}^{(t)} - 1, \ if\ r \geq 0.5 \end{cases}$$

where r is a uniformly distributed random number in [0, 1].

Step-4: Compute $v_i^{(t+1)}$ = argument of better of $\left\{ f(v_i^{(t)}),\ f(\hat{v}_i^{(t)}) \right\}$.

Step-5: Repeat Step-2 to Step-4 for N_{mute} times.

Step-6: Stop.

5.1.7 Termination Criterion

In GA, selection process, crossover, mutation and evaluation are performed repeatedly until a predefined termination criterion (In this work, we have considered the termination criterion as the number of iterations (generations) reaches T) is met.

The RCGA flow can be summarized as follows:

Algorithm-I

```
Step-1: Initialize the parameters of GA.
Step-2: Set $t = 0$
Step-3: Initialize $P(t)$.
Step-4: Evaluate the fitness function value of each solution of $P(t)$.
Step-5: Find the best solution from $P(t)$.
Step-6: Do the following until the termination condition is satisfied:
(a) Increase $t$ by 1.
(b) Select $P(t)$ from $P(t-1)$ by the selection process.
(c) Perform the crossover operation with probability $p_{cross}$.
(d) Perform the mutation operation with probability $p_{mute}$.
(e) Evaluate the fitness function value of each solution of newly created $P(t)$.
(f) Find the best solution from $P(t)$.

Step-7: Print the best solution.
Step-8: Stop.
```

5.2 Self-Organizing Migrating Algorithm (SOMA)

SOMA is a relatively new stochastic evolutionary algorithm, which is based on the social behavior of cooperative solutions and self-organization (e.g. a herd of animals looking for their food). From the existing literature, it is evident that this algorithm has ability to converge towards the global optima (Zelinka et al., 2001). It starts with a population of solutions initialized randomly over the search space at the beginning and then improves the population in loops (called migration loop). In each loop, considering the solution with highest fitness as leader (L), all other solutions (called active solutions (a_i)) will traverse in the direction of the leader. Whether the solution will travel a certain distance (called path length) towards the leader in N_s steps (number of steps) of defined length or not, it depends upon a perturbation parameter. This perturbation works as mutation operator of Genetic Algorithm. If the path length is greater than one, then active solution will over shoot the leader (Zelinka, 2004).

The main control parameters used in SOMA are Pop_{size} (number of solutions in the population), n (number of decision variables of objective function of optimization problem), $Path\_length$ (distance of movement of active solutions), $Step_{size}$ (size of a migration step), PRT (perturbation determines whether a solution will travel directly towards the Leader or not), $Migrations$ (number of iterations).The suggested values (Zelinka, 2004) for the above parameters are given in Table 1.

Parameter name	**Suggested range**
Pop_{size}	10 to any integer number
n	Problem dependent
PRT	(0, 1)
$Path\_length$	[1.1, 3]
$Step_{size}$	[0.11, $Path\_length$]
$Migrations$	10 to any integer number

Table 1. Suggested values for the SOMA parameters

5.2.1 Variations of SOMA

Based on different strategies, different versions of SOMA have been developed. Till now, Zelinka (2004) proposed five strategies as follows:
(i) *All-To-One*, (ii) *All-To-All*, (iii) *All-To-All Adaptive*, (iv) *All-To-Rand* and (v) *Clusters*.
Here we shall discuss SOMA with *All-To-One* strategy in details.

5.2.2 SOMA with All-To-One Strategy

The evolution of SOMA performs with perturbation (equivalent to mutation for GA) and crossover operation. Here the movement of active solutions in the search space is perturbed, not mutated. Perturbation depends on the PRT controlling parameter and perturbation vector ($PRTVector$) (Zelinka, 2004). This SOMA generates a random number to assign PRT parameter from the interval (0, 1). If PRT equals to 0 then the perturbation is not performed and if PRT equals to 1 then the stochastic nature of SOMA will be vanished. $PRTVector$ is created by the following condition:

If $r_1 < PRT$, then $PRTVector_j$ = 1, else $PRTVector_j$ = 0; where $j = 1, 2, \cdots, n$ and r_1 is a random number between 0 and 1. Before a solution starts its journey over the search space towards the Leader, this $PRTVector_j$ is created for each solution's component (see Fig. 2).

SOMA creates a new solution at $(t+1)^{th}$ iteration by the special operation (Zelinka, 2004) (known as crossover in case of SOMA) as follows:

$$v_{i\,j}^{(t+1)} = v_{i\,j,\,\text{start}}^{(t)} + \beta(v_{L\,j}^{(t)} - v_{i\,j,\text{start}}^{(t)})PRTVector_j \tag{3}$$

where $\beta \in \left[0, \text{ by } Step_{size} \text{ to, } Path\_length\right]$.

This operation helps the active solution to move its new position and travels in the search space towards the leader in N_S steps of defined length. The principle of SOMA with *All-To-One* strategy is shown in Fig. 2.

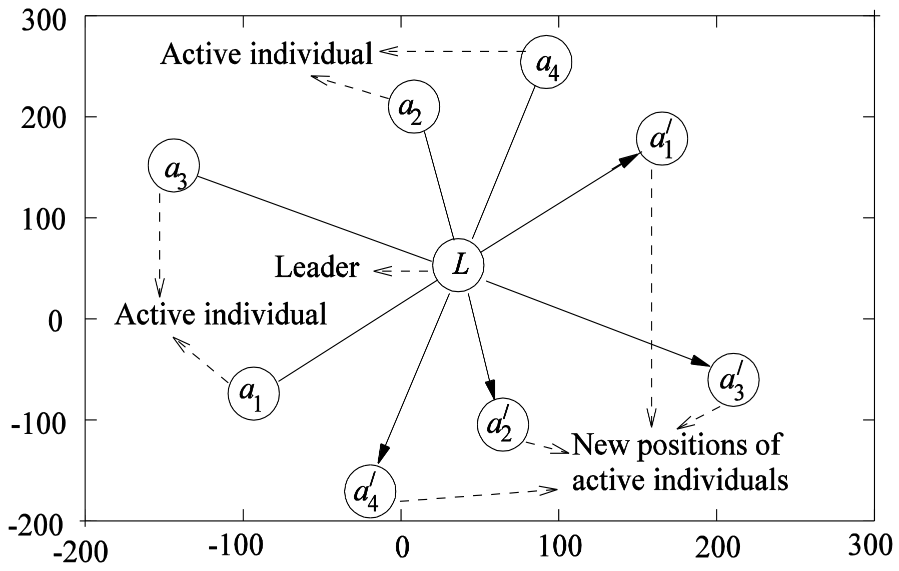

Fig. 2. Movement of active solutions (a_i) towards the Leader (L) for two dimensional search spaces

The different steps of **SOMA** are as follows:

Algorithm-II

```
Step-1: Generate initial population of solutions.
Step-2: Repeat the following for the number of migrations times:
  (i) Generate PRTVector parameter.
  (ii) Evaluate the fitness function value of each solution of $P(t)$.
  (iii) Find the best solution of population and consider it as leader ( $L$ ).
  (iv) Do the following for each active solution of the population:
     (a) Set each active solution as best solution.
     (b) Do the following for specified number of steps:
        ➢ Find the new solution towards the position of the leader ( $L$ ) starting from active by the Eq. (3).
        ➢ Evaluate the fitness function for new solution.
        ➢ If the fitness of new solution is better than the fitness of previous best solution then replace that best solution by the new one.
     (c) Replace the active solution by the best solution obtained from earlier Step-2 (iv) (b).
Step-3: Print the best solution.
Step-4: Stop.
```

5.3 Proposed Real Coded Self-Organizing Migrating Genetic Algorithm (C-RCSOMGA)

In this algorithm, initially a population is created by randomly generated solutions and is evaluated through the fitness function. After that, in each iteration, GA as well as SOMA operators are applied consecutively to the population to improve the same. At the end of the successful applications of GA operators, the best solution is selected according to the fitness value. Then, considering the best solution as leader (L) and others as active solutions (a_i), SOMA operators are applied. In this application, a perturbation vector ($PRTVector$) is created first, then for each active solution, a set of new solutions is created by equation (3) along the path from active to leader at an equal length. After this, the best solution (called Perturbed Leader, PL) is selected from the created new solutions for each active solution and the previous leader. In the next migration loop this PL will work as the leader (L). This process will be continued until the termination criterion is satisfied. In this connection, we call this new strategy as *All-To-One Adaptive* strategy which is incorporated in SOMA of our proposed C-RCSOMGA (see Fig. 3).

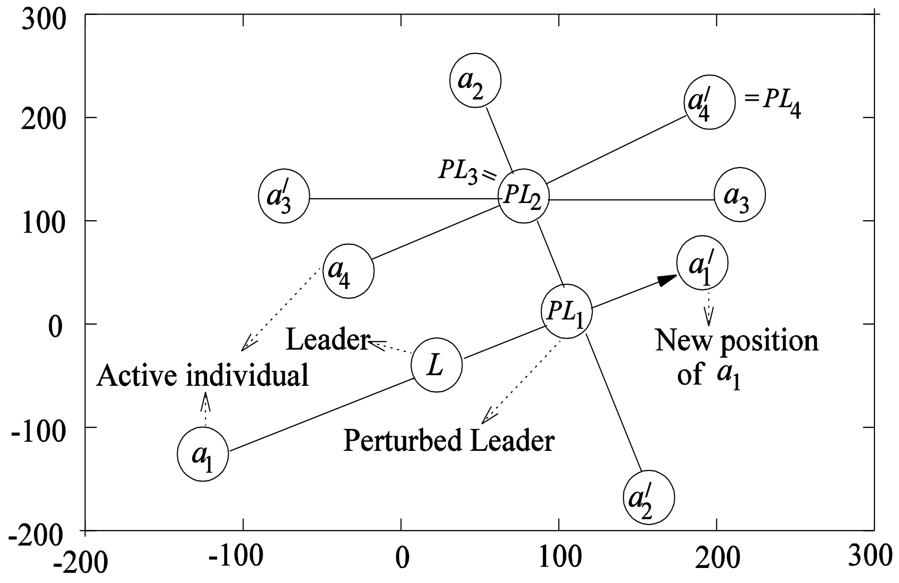

Fig. 3. The principle of C-RCSOMGA with All-To-One Adaptive strategy for two dimensional search spaces

The computational steps of C-RCSOMGA are as follows:

Algorithm-III

Step-1: Initialize the bounds of decision variables, parameters of GA and SOMA, and different parameters of the optimization problems.

Step-2: Set $t = 0$ [t , the number of current iteration].

Step-3: Initialize $P(t)$ [$P(t)$, the population of solutions / solutions at t -th iteration].

Step-4: Evaluate the fitness function value of each solution of $P(t)$.

Step-5: Find the best solution from $P(t)$.

Step-6: Increase the value of t by 1.

Step-7: Select $P(t)$ from $P(t-1)$ by tournament selection process.

Step-8: Apply modified uniform crossover with probability p_{cross}.

Step-9: Apply modified one-neighbourhood mutation with probability p_{mute}.

Step-10: Evaluate the fitness function value of each solution of $P(t)$.

Step-11: Find the best fitted solution (Leader, L) of $P(t)$ and consider all others solutions as active solutions of this population.

Step-12: Apply SOMA with *All-To-One Adaptive* strategy as follows:

(i) Do the following for each solution of the population:

(a) If the solution is infeasible then go to Step-12 (i) (c).

(b) If the absolute value of the difference between the objective function value of Leader (L) and objective function value of the solution is less than ε (in this work, ε = 0.001) then go to Step-12 (i).

(c) Generate $PRTVector$.

(d) Do the following for specified number of $Step_{size}$:

- Create a new population with the help of Eq. (3).
- Evaluate fitness for each solution of that new population.
- Find the best solution (Perturbed Leader, PL) from this created new population.

(e) If PL is better than L, replace the L by PL.

Step-13: If (termination criterion is satisfied) go to Step-14, otherwise go to Step-6.

Step-14: Print the best result.

Step-15: Stop.

6. Numerical Results and Discussion

To illustrate and also to compare the results obtained from the proposed algorithm with the existing algorithms, we have solved four examples. The data for these four examples have been taken from Nahas and Nourelfath (2005) and shown in Appendix (see Table A.1, A.2, A.3 and A.4). However, Nahas and Nourelfath (2005) solved these examples considering without redundancy. In these examples, the available budgets are \$1000, \$900, \$1000 and \$1400. First of all we have solved the examples for reliability optimization without redundancy and compared the results with the same obtained from the existing algorithms (Nahas and Nourelfath, 2005) Algorithm – 1 (AS+Alg1) and Algorithm – 2 (AS+Alg1+Local). These results have been show in Table 4, 5. Using the same numerical data of Example 1 to 4, reliability optimization problem with redundancy has been solved for different budget. On the other hand, cost optimization problems with / without redundancy have been solved considering different lower bounds of reliability. The computational results have been shown in Table 4 – 15.

Due to the stochastic nature of the proposed algorithm, 100 independent runs have been made for each problem considering different sets of random numbers. The proposed algorithm has been coded in C / C++ environment and the simulation has been done on a PC with Intel Core-i3 (2.5 GHz) processor in LINUX environment. The stopping criterion used in the proposed C-RCSOMGA performs up to maximum number of allowable iteration (T). The values of parameters of C-RCSOMGA are given in Table 2 and Table 3.

Parameters	Pop_{size}	p_{cross}	p_{mute}	PRT	$Path\_length$	$Step_{size}$
C-RCSOMGA	100	0.85	0.05	0.75	2	0.11 - 0.25

Table 2. Experimental setup for C-RCSOMGA

Example # (# of tech. used)	Reliability optimization problem		Cost optimization problem	
	without redundancy	with redundancy	without redundancy	with redundancy
1 (61)	300	2000	500	2000
2 (80)	300	2000	500	2000
3 (100)	500	2000	700	2000
4 (166)	1000	2000	1200	2000

Table 3. Experimental setup for C-RCSOMGA for the values of allowable maximum number of iteration (T) for different problems

Example # (# of tech. used)	% of feasible solution	Algorithm used	System Reliability			
			Average	StDev	Best	Min
1 (61)	N/A	AS+Alg1	0.85632000	0.00029000	0.85705000	0.85602000
	N/A	AS+Alg1+Local	0.85705000	0.00000000	0.85705000	0.85705000
	100	C-RCSOMGA	**0.85705458**	0.00000000	**0.85705458**	**0.85705458**
2 (80)	N/A	AS+Alg1	0.90036000	0.01011000	0.91504000	0.87868000
	N/A	AS+Alg1+Local	0.91504000	0.00000000	0.91504000	0.91504000
	100	C-RCSOMGA	**0.91504172**	0.00000000	**0.91504172**	**0.91504172**
3 (100)	N/A	AS+Alg1	0.95850000	0.00250000	0.96406000	0.95624000
	N/A	AS+Alg1+Local	0.96439000	0.00050000	0.96513000	0.96406000
	100	C-RCSOMGA	**0.96513425**	0.00000000	**0.96513425**	**0.96513425**
4 (166)	N/A	AS+Alg1	0.77546000	0.01646000	0.80632000	0.76074000
	N/A	AS+Alg1+Local	0.86491000	0.00038000	0.86543000	0.86465000
	100	C-RCSOMGA	**0.86543863**	0.00000000	**0.86543863**	**0.87000000**

Table 4. Comparison between the results of reliability optimization without redundancy for different Algorithm

N/A - indicates the non-availability of results in literature.

Example #(# of tech. used)	Algorithm used	Best solution among all runs	System Reliability corres. to best solution
		Selected technologies	
1 (61)	AS+Alg1+Local	3-4-5-2-3-3-2-3-2-2-2-3-4-3-2	0.85705000
	C-RCSOMGA	3-4-5-2-3-3-2-3-2-2-2-3-4-3-2	0.85705458
2 (80)	AS+Alg1+Local	3-3-3-4-2-3-3-2-4-1-2-3-4-3-1	0.91504000
	C-RCSOMGA	3-3-3-4-3-3-2-2-4-1-2-4-3-3-1	0.91504172
3 (100)	AS+Alg1+Local	3-3-4-4-3-3-2-2-3-2-2-4-4-4-2	0.96406000
	C-RCSOMGA	3-3-4-4-3-3-2-2-3-2-2-4-4-4-2	0.96513425
4 (166)	AS+Alg1+Local	3-3-3-5-2-3-2-2-3-1-2-3-4-4-1-2-3-3-4-2-3-2-2-3-1	0.86465000
	C-RCSOMGA	2-3-3-5-2-3-2-2-3-1-2-3-4-4-1-3-3-3-4-2-3-2-2-3-1	0.87000000

Table 5. Comparison between the best solutions of reliability optimization without redundancy for different Algorithm

Example # (# of tech. used)	% of feasible solution	Cost (in $)	System Reliability			
			Average	StDev	Best	Worst
1 (61)	4	1000.00	0.66330978	0.14592301	0.76649293	0.56012663
	100	1500.00	0.95737293	0.06539612	0.99786000	0.70272100
	100	2000.00	0.99942527	0.00103140	0.99984776	0.98971794
	100	2500.00	0.99997221	0.00002808	0.99999372	0.99987642
	100	3000.00	0.99999653	0.00000464	0.99999952	0.99997381
2 (80)	0	1000.00	*	*	*	*
	40	1500.00	0.94207788	0.086130537	0.99751069	0.67791754
	100	2000.00	0.99759396	0.007374591	0.99986934	0.94948385
	100	2500.00	0.99992546	0.000173946	0.99999524	0.99898519
	100	3000.00	0.99999140	0.000049970	0.99999989	0.99949879
3 (100)	0	1000.00	*	*	*	*
	3	1500.00	0.92626357	0.11169830	0.99724094	0.79751090
	69	2000.00	0.98288200	0.05065330	0.99995316	0.74795818
	100	2500.00	0.99987742	0.00034136	0.99999600	0.99695193
	100	3000.00	0.99999329	0.00003173	0.99999993	0.99968348
4 (166)	0	3000.00	*	*	*	*
	9	3500.00	0.99690361	0.004690514	0.99986526	0.98801831
	69	4000.00	0.99902158	0.006380859	0.99998640	0.94686575
	100	4500.00	0.99985665	0.000999104	0.99999716	0.98999433
	100	5000.00	0.99989291	0.000999533	0.99999986	0.98999914

Table 6. Computational results of reliability of reliability optimization with redundancy for different Budget for hybrid C-RCSOMGA

* - indicates that no feasible solution of the corresponding problem has been obtained.

From Table 4, it is observed that for all the examples, our proposed algorithm gives better solution than the existing algorithms (Nahas and Nourelfath, 2005) for reliability optimization without redundancy. It is also seen that among 100 runs maximum, minimum and average (mean) values of reliability be the same and obviously standard deviation is zero. In Table 5, best found solutions (selected technologies) with reliabilities have been shown for the existing algorithm (Nahas and Nourelfath, 2005) and our proposed algorithm for different examples. From this table, it is observed that for all the examples our algorithm gives the better result.

From Table 6, it is evident that for all the examples average, maximum and minimum values of reliabilities along with standard deviation have been shown considering different values of budget for reliability optimization problem with redundancy. In each case, number of feasible solutions obtained has also been displayed. It is also observed that for higher budget (i.e., for higher cost), the maximum and average reliabilities are greater and in all the runs the obtained results are feasible.

In Table 7, selected technologies, number of redundancies with reliability corresponding to the best found solution among all runs has been displayed for different budget costs for different examples. From this table, it is observed that system reliability is higher for higher cost. However, for examples 2, 3 and 4, feasible solutions are not obtained for budget costs \$1000, \$1000 and \$3000 respectively.

In Table 8 – 11, the computational results of cost optimization problem without redundancy have been displayed for different examples for different lower bound of system reliabilities. On the other hand, the computational results of the cost optimization problem with redundancy have been shown in Table 12 – 15. In Table 8, 9 and 11, it is seen that the feasible solutions of the cost optimization problem without redundancy in different technology of each subsystem are not available among all runs when the lower bounds of system reliability are 0.9000, 0.9500, 0.9900 and 0.9990 for example 1, 0.9990 for example 2 and 0.9990 for example 4 respectively.

Again from Table 8 – 11, it is observed that the standard deviations for total system cost are zero (0) in all cases except in one case when the lower bound of system reliability is 0.9990 for example 3. From Table 8 – 11, it is clear that the best-found cost is higher for higher imposed lower bound of system reliability. The same types of results have been shown in Table 12 – 15.

Example #(# of tech. used)	Cost($)	Best solution among all runs	System Reliability corres. to best solution
		Selected technologies (Selected no. of redundancies)	
1 (61)	1000.00	1-1-1-1-1-1-1-1-1-1-1-1-1-1 (3-3-3-4-1-3-1-5-2-2-2-3-4-6-1)	0.76649293
	1500.00	2-1-1-2-2-4-1-2-1-1-4-3-4-1-1 (2-5-6-3-2-1-3-2-4-3-1-2-1-8-2)	0.99786000
	2000.00	2-2-3-2-2-3-1-1-1-1-2-3-4-1-2 (3-3-3-4-3-2-4-8-5-4-2-2-2-8-2)	0.99984776
	2500.00	7-2-2-2-2-2-1-2-1-1-1-3-4-3-1 (1-4-5-5-4-5-5-3-7-4-6-3-2-3-3)	0.99999372
	3000.00	4-3-2-2-2-3-1-2-1-1-2-3-4-3-1 (2-3-5-6-5-3-6-3-7-8-3-4-3-4-4)	0.99999952
2 (80)	1000.00	* (*)	*
	1500.00	2-1-1-3-2-4-3-1-1-1-2-5-1-1-1 (2-4-6-2-2-1-2-6-4-2-2-1-6-6-2)	0.99751069
	2000.00	1-2-2-3-2-3-3-1-1-1-2-6-3-3-1 (6-3-3-3-3-2-2-8-5-4-2-1-3-3-3)	0.99986934
	2500.00	2-2-2-5-2-3-3-2-1-1-2-4-4-3-1 (4-4-5-3-4-2-2-3-7-4-3-2-2-4-4)	0.99999524
	3000.00	2-3-4-2-2-3-2-2-1-1-2-5-4-3-1 (4-3-3-7-4-3-4-4-8-5-3-2-3-5-4)	0.99999989
3 (100)	1000.00	* (*)	*
	1500.00	4-2-1-2-5-3-2-1-5-1-4-3-1-1-3 (1-2-5-3-1-2-2-5-1-2-1-2-5-6-1)	0.99724094
	2000.00	5-1-1-4-6-2-2-1-6-1-1-4-4-1-1 (1-6-8-2-1-5-3-7-1-4-4-2-2-8-3)	0.99995316
	2500.00	2-4-4-5-2-3-1-2-4-1-2-4-5-3-2 (4-2-3-2-4-2-5-3-2-6-3-2-2-4-2)	0.99999600
	3000.00	4-2-7-5-2-3-5-2-1-1-1-5-4-4-1 (2-5-2-2-4-3-2-4-8-8-7-2-3-3-4)	0.99999993
4 (166)	3000.00	* (*)	*
	3500.00	4-2-8-3-4-1-6-2-1-1-2-4-1-1-4-1-2-4-5-3-3-6-1-3-4 (2-4-1-3-2-6-1-2-5-5-2-2-8-8-1-6-3-2-2-2-2-1-8-2-1)	0.99986526
	4000.00	3-1-2-4-2-3-4-2-4-1-5-4-7-4-6-6-2-2-4-2-3-7-2-8-1 (2-8-5-4-3-3-2-3-2-7-1-2-1-2-1-1-4-6-3-3-3-1-4-1-5)	0.99998640
	4500.00	3-3-6-5-4-4-4-2-1-2-2-4-4-2-1-2-3-2-3-4-3-4-2-5-1 (3-3-2-2-2-2-2-5-7-3-3-2-2-6-4-4-3-5-4-2-3-2-3-2-8)	0.99999716
	5000.00	5-3-4-5-4-4-6-2-5-1-1-4-4-4-2-2-2-2-5-5-3-2-2-1-2 (2-3-3-2-2-2-2-4-2-7-6-3-3-3-3-5-5-6-2-2-3-4-4-8-3)	0.99999986

Table 7. Computational results of corresponding best solution of reliability optimization with redundancy for different budget for hybrid C-RCSOMGA

* - indicates that no feasible solution of the corresponding problem has been obtained.

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)				Selected technologies corres. to best found solution
		Average	StDev	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	720.00	0	720.00 (0.76621405)	720.00 (0.75140173)	2-2-2-2-2-3-2-2-2-1-2-3-3-3-1
100	0.8000	770.00	0	770.00 (0.80559763)	770.00 (0.80559763)	2-3-3-2-2-3-2-2-2-1-2-3-3-3-1
100	0.8500	930.00	0	930.00 (0.85114618)	930.00 (0.85037926)	3-3-5-2-3-3-2-2-2-2-2-3-4-3-2
0	0.9000	*	*	*	*	*
0	0.9500	*	*	*	*	*
0	0.9900	*	*	*	*	*
0	0.9990	*	*	*	*	*

Table 8. Computational results of cost optimization problem without redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 1
*- indicates that no feasible solution of the corresponding problem has been obtained.

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)				Selected technologies corres. to best found solution
		Average	StDev	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	720.00	0	720.00 (0.76033925)	720.00 (0.75065072)	2-2-2-3-2-3-1-2-2-1-2-3-3-3-1
100	0.8000	750.00	0	750.00 (0.80869072)	750.00 (0.80026686)	2-2-2-4-2-3-2-2-2-1-2-3-3-3-1
100	0.8500	790.00	0	790.00 (0.86029801)	790.00 (0.86029801)	2-2-3-4-2-3-2-2-3-1-2-3-3-3-1
100	0.9000	865.00	0	865.00 (0.90316727)	865.00 (0.90316727)	2-3-3-5-2-3-2-2-4-1-2-4-3-3-1
100	0.9500	995.00	0	995.00 (0.95260974)	995.00 (0.95070067)	3-3-4-5-3-3-3-2-4-2-2-4-4-3-1
72	0.9900	1325.00	0	1325.00 (0.99023365)	1325.00 (0.99023365)	4-4-5-5-4-4-4-4-5-2-3-5-4-6-2
0	0.9990	*	*	*	*	*

Table 9. Computational results of cost optimization problem without redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 2
*- indicates that no feasible solution of the corresponding problem has been obtained.

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)				Selected technologies corres. to best found solution
		Average	StDev	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	710.00	0	710.00 (0.76817779)	710.00 (0.75031118)	2-2-2-3-2-3-1-2-2-1-2-3-3-3-1
100	0.8000	730.00	0	730.00 (0.80456517)	730.00 (0.80052213)	2-2-2-3-2-3-1-2-3-1-2-3-3-3-1
100	0.8500	760.00	0	760.00 (0.85469711)	760.00 (0.85469711)	2-2-2-4-2-3-2-2-3-1-2-3-3-3-1
100	0.9000	830.00	0	830.00 (0.90679810)	830.00 (0.90269287)	2-3-3-4-2-3-2-2-3-1-2-3-3-4-1
100	0.9500	955.00	0	955.00 (0.95020322)	955.00 (0.95020322)	3-3-4-4-2-3-3-2-4-1-2-4-4-4-1
100	0.9900	1180.00	0	1180.00 (0.99014455)	1180.00 (0.99004557)	3-4-5-5-4-4-4-3-4-2-2-4-4-4-2
52	0.9990	1550.77	3.92232	1550.00 (0.99903024)	1570.00 (0.99900428)	5-5-7-5-5-4-5-4-5-3-4-5-6-5-3

Table 10. Computational results of cost optimization problem without redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 3

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)				Selected technologies corres. to best found solution
		Average	StDev	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	1225.00	0	1225.00 (0.76123401)	1225.00 (0.75330450)	2-2-2-4-2-3-2-2-3-1-2-3-3-3-1-2-2-2-4-2-3-2-2-3-1
100	0.8000	1265.00	0	1265.00 (0.80955458)	1265.00 (0.80955458)	2-2-3-4-2-3-2-2-3-1-2-3-3-3-1-2-2-3-4-2-3-2-2-3-1
100	0.8500	1365.00	0	1365.00 (0.85686995)	1365.00 (0.85128304)	2-3-3-4-2-3-2-2-3-1-2-3-4-3-1-3-3-3-4-2-3-2-2-3-1
100	0.9000	1490.00	0	1490.00 (0.90106570)	1490.00 (0.90016279)	3-3-4-4-3-3-3-2-3-1-2-3-4-4-1-3-3-4-4-2-3-2-2-3-1
100	0.9500	1650.00	0	1650.00 (0.95029611)	1650.00 (0.95029611)	3-3-4-5-3-3-3-2-3-2-2-4-4-4-2-3-3-4-4-3-3-3-2-3-2
100	0.9900	2090.00	0	2090.00 (0.99054161)	2090.00 (0.99001633)	4-4-5-5-4-3-4-4-4-2-3-5-4-5-2-4-4-5-5-4-4-4-4-4-2
0	0.9990	*	*	*	*	*

Table 11. Computational results of cost optimization problem without redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 4
*- indicates that no feasible solution of the corresponding problem has been obtained.

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)			Selected technologies corres. to best found solution (Selected number of redundancies)
		Average	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	837.00	720.00 (0.75043378)	1150.00 (0.75029566)	1-2-2-1-2-1-2-2-1-1-2-3-1-1-1 (2-1-1-2-1-2-1-1-2-1-1-1-2-3 1)
100	0.8000	858.10	755.00 (0.80050133)	1020.00 (0.80188406)	3-1-1-2-2-1-2-2-1-1-1-1-3-1-1 (1-2-3-1-1-2-1-1-2-1-2-2-1-3-1)
100	0.8500	911.20	810.00 (0.86104210)	1110.00 (0.85330449)	2-3-1-2-2-3-2-1-1-1-2-3-1-3-1 (1-1-2-2-1-1-1-3-2-1-1-1-3-1-1)
100	0.9000	983.40	870.00 (0.90063263)	1190.00 (0.90066850)	1-1-1-2-2-3-2-2-1-1-2-3-3-1-3 (3-2-3-2-1-1-1-1-2-1-1-1-1-4-1)
100	0.9500	1088.90	985.00 (0.95030210)	1200.00 (0.95024704)	1-3-2-2-1-3-1-2-1-1-1-3-1-1-2 (3-1-2-2-3-1-2-1-2-2-2-1-4-4-1)
100	0.9900	1412.20	1305.00 (0.99008213)	1625.00 (0.99049496)	1-1-5-2-2-1-1-1-1-1-2-3-1-1-2 (3-4-1-3-2-3-2-5-3-2-1-2-5-6-1)
100	0.9990	1801.30	1625.00 (0.99901825)	1995.00 (0.99900971)	3-5-2-2-2-3-2-1-1-1-1-3-3-1-1 (2-1-3-4-2-2-2-6-4-3-3-2-2-6-2)
100	0.9999	2220.60	2045.00 (0.99991102)	2510.00 (0.99991365)	6-2-1-2-2-3-1-1-1-1-2-1-4-1-2 (1-3-8-4-3-2-4-6-5-3-2-7-2-7-2)
100	0.99999	2600.50	2365.00 (0.99999029)	2945.00 (0.99999014)	4-4-5-2-2-3-2-2-1-1-2-3-4-2-1 (2-2-2-5-3-2-3-3-6-4-2-3-2-5-3)

Table 12. Computational results of cost optimization problem with redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 1

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)			Selected technologies corres. to best found solution (Selected number of redundancies)
		Average	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	861.00	715.00 (0.75332569)	1130.00 (0.75017713)	1-1-1-3-1-1-1-2-1-1-2-1-3-1-1 (2-2-2-1-2-2-1-1-2-1-1-2-1-3-1)
100	0.8000	902.40	775.00 (0.80007523)	1240.00 (0.80003961)	1-2-2-2-1-1-2-1-1-1-1-3-1-3-1 (2-1-1-1-3-2-1-3-2-2-2-1-3-1-1)
100	0.8500	939.20	825.00 (0.85050853)	1220.00 (0.85236908)	3-2-2-4-2-2-2-1-1-2-1-3-3-1-1 (1-1-2-1-1-2-1-3-2-1-1-1-1-3-1)
100	0.9000	1004.50	875.00 (0.90106520)	1280.00 (0.90137277)	2-3-1-2-2-1-1-1-1-1-2-2-3-1-1 (1-1-3-2-1-2-2-3-2-1-1-2-1-4-1)
100	0.9500	1106.20	975.00 (0.95126599)	1275.00 (0.95207199)	1-1-1-2-1-1-1-1-1-1-1-4-3-1-2 (3-3-3-2-3-3-2-4-2-2-2-1-1-4-1)
100	0.9900	1412.90	1275.00 (0.99002985)	1880.00 (0.99011295)	1-2-1-2-3-3-4-1-1-3-3-1-4-1-1 (3-2-4-3-1-1-1-4-3-1-1-4-1-6-2)
100	0.9990	1782.80	1595.00 (0.99900702)	1980.00 (0.99900596)	1-1-2-2-2-2-2-2-1-1-4-5-3-1-1 (4-5-3-4-2-3-2-3-4-3-1-1-2-8-2)
100	0.99999	2512.00	2280.00 (0.99999027)	2880.00 (0.99999144)	1-1-2-2-2-3-3-2-1-1-1-5-4-3-1 (6-8-5-5-3-2-2-3-6-4-5-2-2-3-3)

Table 13. Computational results of cost optimization problem with redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 2

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)			Selected technologies corres. to best found solution (Selected number of redundancies)
		Average	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	940.30	725.00 (0.75110484)	1155.00 (0.75005996)	3-1-3-4-1-3-2-1-1-1-1-1-1-1 (1-2-1-1-2-1-1-2-2-1-1-2-2-3-1)
100	0.8000	988.10	760.00 (0.80308843)	1360.00 (0.82297242)	3-1-3-5-2-3-2-1-3-1-2-3-1-1-1 (1-2-1-1-1-1-1-2-1-1-1-1-2-2-1)
100	0.8500	1011.10	800.00 (0.85202849)	1290.00 (0.86985933)	4-1-1-3-1-1-2-1-1-1-1-3-3-3-1 (1-2-3-1-2-2-1-3-2-1-2-1-1-1-1)
100	0.9000	1041.90	875.00 (0.90759672)	1360.00 (0.90115269)	3-1-1-4-1-2-1-1-3-1-1-4-1-1-1 (1-2-3-1-3-2-2-3-1-1-2-1-3-3-1)
100	0.9500	1178.50	950.00 (0.95029904)	1730.00 (0.95216062)	3-3-3-5-1-3-4-1-1-1-2-4-4-1-1 (1-1-1-1-3-1-1-4-2-2-1-1-1-3-1)
100	0.9900	1409.00	1225.00 (0.99030730)	1730.00 (0.99066244)	3-1-1-3-3-4-1-1-4-1-2-5-4-1-3 (2-4-4-2-1-1-2-4-1-2-1-1-1-5-1)
100	0.9990	1811.60	1590.00 (0.99912943)	2015.00 (0.99900853)	2-5-2-4-1-5-6-1-5-2-1-5-4-1-1 (2-1-3-2-5-1-1-5-1-2-3-1-2-6-2)
100	0.9999	2186.60	1935.00 (0.99990842)	2570.00 (0.99990140)	2-1-1-4-1-3-3-2-1-1-2-4-7-4-5 (3-6-7-2-6-2-2-2-5-3-2-2-1-2-1)
100	0.99999	2589.10	2270.00 (0.99999075)	3025.00 (0.99999441)	2-2-5-5-2-4-8-1-1-1-4-4-4-1-1 (3-4-2-2-3-2-1-8-6-4-2-2-2-8-3)

Table 14. Computational results of cost optimization problem with redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 3

% of feasible solution	Imposed lower bound of system reliability	Cost (in $)			Selected technologies corres. to best solution (Selected number of redundancies)
		Average	Best (corres. System Reliability)	Worst (corres. System Reliability)	
100	0.7500	1680.30	1425.00 (0.75062799)	1980.00 (0.75547077)	4-1-3-3-2-2-1-1-1-1-2-3-7-1-1-4-1-2-2-2-2-2-2-2-1 (1-2-1-1-1-2-2-3-2-1-1-1-1-3-1-1-2-2-2-1-1-1-1-1-1)
100	0.8000	1759.00	1400.00 (0.80021206)	2185.00 (0.80100284)	3-1-3-2-2-3-2-1-3-1-1-1-3-1-1-4-1-1-3-1-3-4-1-3-1 (1-2-1-2-1-1-1-3-1-1-2-3-1-3-1-1-2-2-1-3-1-1-3-1-1)
100	0.8500	1827.00	1420.00 (0.85202907)	2290.00 (0.85023684)	1-2-1-4-1-3-2-1-1-2-1-3-1-1-1-1-2-4-4-2-2-1-1-1-1 (3-1-3-1-3-1-1-3-2-1-2-1-3-3-1-2-2-1-1-1-2-2-3-2-1)
100	0.9000	1891.60	1580.00 (0.90138278)	2190.00 (0.90000340)	2-3-1-4-2-1-1-3-1-1-2-1-3-4-2-1-1-2-4-1-2-2-1-1-1 (2-1-3-1-1-2-2-1-2-2-1-3-1-1-1-3-3-2-1-3-2-1-3-2-2)
100	0.9500	2085.40	1810.00 (0.95140623)	2495.00 (0.95192691)	4-1-1-4-5-2-4-1-5-1-1-4-3-1-3-1-1-2-2-1-2-2-1-1-1 (1-3-4-1-1-2-1-4-1-2-2-1-1-4-1-3-3-2-2-3-3-1-4-3-2)
100	0.9900	2536.10	2200.00 (0.99005870)	3010.00 (0.99000153)	1-1-5-3-1-2-3-1-4-1-1-3-5-1-1-3-1-1-5-1-3-6-1-4-2 (4-4-1-2-5-3-1-5-1-2-3-2-1-6-2-1-4-5-1-4-2-1-5-1-1)
100	0.9990	3283.40	2940.00 (0.99903265)	3870.00 (0.99903163)	2-1-1-3-5-4-5-5-1-1-1-3-7-1-4-7-5-6-3-6-1-1-2-3-1 (3-5-8-3-1-1-1-1-4-3-3-2-1-7-1-1-1-1-3-1-4-4-2-2-3)
100	0.9999	3807.10	3290.00 (0.99990197)	4265.00 (0.99990278)	1-2-7-4-2-5-3-2-4-1-2-4-4-3-1-2-2-2-3-4-5-3-1-6-1 (6-3-1-2-3-1-2-3-2-3-2-2-2-3-3-3-3-4-3-2-1-2-8-1-3)
100	0.99999	4402.10	3985.00 (0.99999001)	5105.00 (0.99999024)	4-2-2-5-4-3-4-2-3-1-1-5-4-1-1-7-2-5-4-2-3-1-2-4-1 (2-5-4-2-2-3-2-3-3-5-5-2-2-8-3-1-4-2-3-4-2-5-3-2-4)

Table 15. Computational results of cost optimization problem with redundancy for different technology of each subsystem for hybrid C-RCSOMGA for example 4

7. Concluding Remarks

This paper deals with two reliability optimization problems for a series system with parallel redundancy incorporating multiple choice constraints. In the first problem, system reliability is maximized subject to a budget constraint whereas in the second problem, system cost is minimized subject to a minimum level of system reliability. These problems are NP-hard problems. To solve these, we have developed hybrid heuristic approach based on parameter free penalty technique, RCGA and SOMA. Here, penalty function is to obtain the feasible solutions. In this penalty technique, only a large negative value (in case of maximization problem) or a very large value (in case of a minimization problem) is considered for infeasible solution. From the experimental results, it is observed that the optimal or near to optimal solution (though the optimality cannot be tested analytically) can be obtained quickly.

Acknowledgment
For this research, the first author would like to acknowledge the financial support provided by the Council of Scientific and Industrial Research (CSIR), New Delhi, India.

References

Bhunia, A. K., Sahoo, L., & Roy, D. (2010). Reliability stochastic optimization for a series system with interval component reliability via genetic algorithm. *Applied Mathematics and Computation*, *216*(3), 929-939.

Brindle, A. (1981). Genetic algorithms for function optimization (Doctoral dissertation and Technical Report TR81-2). *Edmonton: University of Alberta, Department of Computer Science.*

Chelouah, R., & Siarry, P. (2003). Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, *148*(2), 335-348.

Chern, M. S. (1992). On the computational complexity of reliability redundancy allocation in a series system. *Operations Research Letters*, *11*(5), 309-315.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11), 1245-1287.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*, 311–38.

dos Santos Coelho, L. (2009). An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications. *Reliability Engineering and System Safety*, *94*(4), 830-837.

dos Santos Coelho, L. (2009). Self-organizing migration algorithm applied to machining allocation of clutch assembly. *Mathematics and Computers in Simulation*, *80*(2), 427-435.

dos Santos Coelho, L., & Alotto, P. (2009). Electromagnetic optimization using a cultural self-organizing migrating algorithm approach based on normative knowledge. *IEEE Transactions on Magnetics*, *45*(3), 1446-1449.

dos Santos Coelho, L., & Mariani, V. C. (2010). An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect. *Energy Conversion and Management*, *51*(12), 2580-2587.

Fan, S. K. S., Liang, Y. C., & Zahara, E. (2006). A genetic algorithm and a particle swarm optimizer hybridized with Nelder–Mead simplex search. *Computers and Industrial Engineering*, *50*(4), 401-425.

Ghare, P. M., & Taylor, R. E. (1969). Optimal redundancy for reliability in series systems. *Operations Research*, *17*(5), 838-847.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning ‘addison-wesley, 1989. *Reading, MA*.

Goldberg, D., Deb, K., & Korb, B. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, (3), 493-530.

Gupta, R. K., Bhunia, A. K., & Roy, D. (2009). A GA based penalty function technique for solving constrained redundancy allocation problem of series system with interval valued reliability of components. *Journal of Computational and Applied Mathematics*, *232*(2), 275-284.

Ha, C., & Kuo, W. (2006). Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems. *European Journal of Operational Research*, *171*(1), 24-38.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.

Jabeen, S. D., & Bhunia, A. K. (2006). Real-coded genetic algorithm with variable rates of cross-over and mutation: a basis of global optimization for multi-modal functions via interval technique. *International Journal of Computer Mathematics*, *83*(12), 853-866.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary computation*, *7*(1), 19-44.

Kuo, W. (2001). *Optimal reliability design: fundamentals and applications*. Cambridge University Press.

Michalewicz, Z. (1996). Genetic algorithms data structures evolution programs. *Springer, Berlin*.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, *4*(1), 1-32.

Nahas, N., & Nourelfath, M. (2005). Ant system for reliability optimization of a series system with multiple-choice and budget constraints. *Reliability Engineering & System Safety*, *87*(1), 1-12.

Nakagawa, Y., Nakashima, K., & Hattori, Y. (1978). Optimal reliability allocation by branch-and-bound technique. *IEEE Transactions on Reliability*, *1*, 31-38.

Nauss, R. M. (1978). The 0–1 knapsack problem with multiple choice constraints. *European Journal of Operational Research*, *2*(2), 125-131.

Nolle, L., Zelinka, I., Hopgood, A. A., & Goodyear, A. (2005). Comparison of a self-organizing migration algorithm with simulated annealing and differential evolution for automated waveform tuning. *Advances in Engineering Software*, *36*(10), 645-653.

Nourelfath, M., & Nahas, N. (2003). Quantized hopfield networks for reliability optimization. *Reliability Engineering & System Safety*, *81*(2), 191-196.

Pedamallu, C. S., & Ozdamar, L. (2008). Investigating a hybrid simulated annealing and local search algorithm for constrained optimization. *European Journal of Operational Research*, *185*(3), 1230-1245.

Renders, J. M., & Flasse, S. P. (1996). Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(2), 243-258.

Sahoo, L., Bhunia, A. K., & Kapur, P. K. (2012). Genetic algorithm based multi-objective reliability optimization in interval environment. *Computers and Industrial Engineering*, *62*(1), 152-160.

Sakawa, M., & Kato, K. (2002). An interactive fuzzy satisficing method for general multiobjective 0–1 programming problems through genetic algorithms with double strings based on a reference solution. *Fuzzy Sets and Systems*, *125*(3), 289-300.

Salhi, S., & Queen, N. M. (2004). A hybrid algorithm for identifying global and local minima when optimizing functions with many minima. *European Journal of Operational Research*, *155*(1), 51-67.

Senkerik, R., Zelinka, I., Davendra, D., & Oplatkova, Z. (2010). Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation. *Computers and Mathematics with Applications*, *60*(4), 1026-1037.

Sinha, P., & Zoltners, A. A. (1979). The multiple-choice knapsack problem. *Operations Research*, *27*(3), 503-515.

Sun, X. L., & Li, D. (2002). Optimality condition and branch and bound algorithm for constrained redundancy optimization in series systems. *Optimization and Engineering*, *3*(1), 53-65.

Sung, C. S., & Cho, Y. K. (2000). Reliability optimization of a series system with multiple-choice and budget constraints. *European Journal of Operational Research*, *127*(1), 159-171.

Sung, C. S., & Lee, H. K. (1994). A branch-and-bound approach for spare unit allocation in a series system. *European journal of operational research*, *75*(1), 217-232.

Tillman, F. A., Hwang, C. L., & Kuo, W. (1977). Optimization Techniques for System Reliability with Redundancy꜔A Review. *IEEE Transactions on Reliability*, *26*(3), 148-155.

Tillman, F. A., Hwang, C. L., & Kuo, W. (1980). *Optimization of systems reliability* (Vol. 4). Marcel Dekker Inc.

Zelinka, I, & Lampinen, J. (2000). SOMA – self-organizing migrating algorithm. *Proceedings of the 6th international Mendel conference on soft computing, Brno, Czech Republic*; 177–187.

Zelinka, I. (2004). SOMA—self-organizing migrating algorithm. In *New optimization techniques in engineering* (pp. 167-217). Springer Berlin Heidelberg.

Zelinka, I., Lampinen, J., & Nolle, L. (2001). On the theoretical proof of convergence for a class of SOMA search algorithms, In: Proceedings of the 7th International MENDEL Conference on Soft Computing., pp. 103-110. ISBN 0802141894X.

Appendix: The data for the four examples [15] that have been used in this paper and given below (see Table A.1, A.2, A.3 and A.4).

Sub-system		Technology 1	Technology 2	Technology 3	Technology 4	Technology 5	Technology 6	Technology 7	Technology 8
1	Reliability	0.9	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999999
	Cost($)	20	40	60	80	100	120	140	180
2	Reliability	0.85	0.9775	0.9966	0.9995	0.9999	–	–	–
	Cost($)	30	60	90	120	150	–	–	–
3	Reliability	0.8	0.96	0.99	0.998	0.9997	–	–	–
	Cost($)	20	40	60	80	100	–	–	–
4	Reliability	0.75	0.938	–	–	–	–	–	–
	Cost($)	30	40	–	–	–	–	–	–
5	Reliability	0.85	0.99	0.999	–	–	–	–	–
	Cost($)	20	40	65	–	–	–	–	–
6	Reliability	0.9	0.95	0.999	0.9999	–	–	–	–
	Cost($)	25	30	50	70	–	–	–	–
7	Reliability	0.95	0.99	–	–	–	–	–	–
	Cost($)	40	60	–	–	–	–	–	–
8	Reliability	0.85	0.995	0.999	0.9999	0.99999	–	–	–
	Cost($)	10	30	60	80	120	–	–	–
9	Reliability	0.9	0.95	–	–	–	–	–	–
	Cost($)	30	50	–	–	–	–	–	–
10	Reliability	0.99	0.999	0.9999	0.99999	0.999999	–	–	–
	Cost($)	15	40	70	100	130	–	–	–
11	Reliability	0.95	0.999	0.9998	0.99999	0.999998	0.9999999	–	–
	Cost($)	20	40	60	80	100	120	–	–
12	Reliability	0.8	0.9	0.99	–	–	–	–	–
	Cost($)	40	60	85	–	–	–	–	–
13	Reliability	0.75	0.85	0.99	0.999	–	–	–	–
	Cost($)	30	50	80	100	–	–	–	–
14	Reliability	0.8	0.95	0.99	–	–	–	–	–
	Cost($)	10	30	40	–	–	–	–	–
15	Reliability	0.99	0.999	0.9999	0.99999	–	–	–	–
	Cost($)	50	80	110	140	–	–	–	–

Table A.1. Data for example 1 (with 61 variables)

Sub-system		Technology 1	Technology 2	Technology 3	Technology 4	Technology 5	Technology 6	Technology 7	Technology 8
1	Reliability	0.9	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999999
	Cost($)	20	40	60	80	100	120	140	180
2	Reliability	0.85	0.9775	0.9966	0.9995	0.9999	–	–	–
	Cost($)	30	60	90	120	150	–	–	–
3	Reliability	0.8	0.96	0.99	0.998	0.9997	–	–	–
	Cost($)	20	40	60	80	100	–	–	–
4	Reliability	0.75	0.938	0.97	0.99	0.995	–	–	–
	Cost($)	30	40	60	70	80	–	–	–
5	Reliability	0.85	0.99	0.999	0.9999	–	–	–	–
	Cost($)	20	40	65	80	–	–	–	–
6	Reliability	0.9	0.95	0.999	0.9999	–	–	–	–
	Cost($)	25	30	50	70	–	–	–	–
7	Reliability	0.95	0.99	0.999	0.9999	–	–	–	–
	Cost($)	40	60	80	100	–	–	–	–
8	Reliability	0.85	0.995	0.999	0.9999	0.99999	–	–	–
	Cost($)	10	30	60	80	120	–	–	–
9	Reliability	0.9	0.95	0.98	0.995	0.9999	–	–	–
	Cost($)	30	50	70	90	120	–	–	–
10	Reliability	0.99	0.999	0.9999	0.99999	0.999999	–	–	–
	Cost($)	15	40	70	100	130	–	–	–
11	Reliability	0.95	0.999	0.9998	0.99999	0.999998	0.9999999	–	–
	Cost($)	20	40	60	80	100	120	–	–
12	Reliability	0.8	0.9	0.99	0.999	0.9999	0.99999	0.999997	0.9999995
	Cost($)	40	60	85	100	120	140	155	170
13	Reliability	0.75	0.85	0.99	0.999	–	–	–	–
	Cost($)	30	50	80	100	–	–	–	–
14	Reliability	0.8	0.95	0.99	0.996	0.9993	0.9999	0.99996	0.999998
	Cost($)	10	30	40	60	80	95	120	140
15	Reliability	0.99	0.999	0.9999	0.99999	–	–	–	–
	Cost($)	50	80	110	140	–	–	–	–

Table A.2. Data for example 2 (with 80 variables)

Sub-system		Technology 1	Technology 2	Technology 3	Technology 4	Technology 5	Technology 6	Technology 7	Technology 8
1	Reliability	0.9	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999999
	Cost($)	20	40	60	80	100	120	140	180
2	Reliability	0.85	0.9775	0.9966	0.9995	0.9999	–	–	–
	Cost($)	30	60	90	120	150	–	–	–
3	Reliability	0.8	0.96	0.99	0.998	0.9997	0.9999	0.99999	0.999999
	Cost($)	20	40	60	80	100	120	140	160
4	Reliability	0.75	0.938	0.98	0.999	0.9999	–	–	–
	Cost($)	30	40	50	60	70	–	–	–
5	Reliability	0.85	0.99	0.999	0.9999	0.99998	0.999998	0.9999998	0.99999998
	Cost($)	20	40	65	80	100	120	140	155
6	Reliability	0.9	0.95	0.999	0.9999	0.99999	–	–	–
	Cost($)	25	30	50	70	90	–	–	–
7	Reliability	0.95	0.99	0.997	0.9997	0.99997	0.999997	0.9999997	0.99999997
	Cost($)	40	60	80	100	120	140	160	180
8	Reliability	0.85	0.995	0.999	0.9999	0.99999	–	–	–
	Cost($)	10	30	60	80	120	–	–	–
9	Reliability	0.9	0.95	0.995	0.9995	0.99995	0.999995	0.9999995	0.99999995
	Cost($)	30	50	70	90	110	130	150	170
10	Reliability	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	–	–
	Cost($)	15	40	70	100	130	160	–	–
11	Reliability	0.95	0.999	0.9998	0.99999	0.999998	0.9999999	0.99999997	0.99999999
	Cost($)	20	40	60	80	100	120	140	160
12	Reliability	0.8	0.9	0.99	0.999	0.9999	–	–	–
	Cost($)	40	60	85	110	130	–	–	–
13	Reliability	0.75	0.85	0.99	0.999	0.9996	0.99996	0.999996	0.9999996
	Cost($)	30	50	80	100	120	140	160	180
14	Reliability	0.8	0.95	0.99	0.999	0.9999	–	–	–
	Cost($)	10	30	40	60	80	–	–	–
15	Reliability	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999998	0.99999999
	Cost($)	50	80	110	140	160	180	200	220

Table A.3. Data for example 3 (with 100 variables)

Sub-system		Technology 1	Technology 2	Technology 3	Technology 4	Technology 5	Technology 6	Technology 7	Technology 8
1	Reliability	0.9	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999999
	Cost($)	20	40	60	80	100	120	140	180
2	Reliability	0.85	0.9775	0.9966	0.9995	0.9999	–	–	–
	Cost($)	30	60	90	120	150	–	–	–
3	Reliability	0.8	0.96	0.99	0.998	0.9997	0.9999	0.99999	0.999999
	Cost($)	20	40	60	80	100	120	140	160
4	Reliability	0.75	0.938	0.98	0.999	0.9999	–	–	–
	Cost($)	30	40	50	60	70	–	–	–
5	Reliability	0.85	0.99	0.999	0.9999	0.99998	0.999998	0.9999998	0.99999998
	Cost($)	20	40	65	80	100	120	140	155
6	Reliability	0.9	0.95	0.999	0.9999	0.99999	–	–	–
	Cost($)	25	30	50	70	90	–	–	–
7	Reliability	0.95	0.99	0.997	0.9997	0.99997	0.999997	0.9999997	0.99999997
	Cost($)	40	60	80	100	120	140	160	180
8	Reliability	0.85	0.995	0.999	0.9999	0.99999	–	–	–
	Cost($)	10	30	60	80	120	–	–	–
9	Reliability	0.9	0.95	0.995	0.9995	0.99995	0.999995	0.9999995	0.99999995
	Cost($)	30	50	70	90	110	130	150	170
10	Reliability	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	–	–
	Cost($)	15	40	70	100	130	160	–	–
11	Reliability	0.95	0.999	0.9998	0.99999	0.999998	0.9999999	0.99999997	0.999999999
	Cost($)	20	40	60	80	100	120	140	160
12	Reliability	0.8	0.9	0.99	0.999	0.9999	–	–	–
	Cost($)	40	60	85	110	130	–	–	–
13	Reliability	0.75	0.85	0.99	0.999	0.9996	0.99996	0.999996	0.9999996
	Cost($)	30	50	80	100	120	140	160	180
14	Reliability	0.8	0.95	0.99	0.999	0.9999	–	–	–
	Cost($)	10	30	40	60	80	–	–	–
15	Reliability	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999998	0.999999995
	Cost($)	50	80	110	140	160	180	200	220
16	Reliability	0.9	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	0.99999999
	Cost($)	20	40	60	80	100	120	140	180
17	Reliability	0.85	0.9775	0.9966	0.9995	0.9999	–	–	–
	Cost($)	30	60	90	120	150	–	–	–
18	Reliability	0.8	0.96	0.99	0.998	0.9997	0.9999	0.99999	0.999999
	Cost($)	20	40	60	80	100	120	140	160
19	Reliability	0.75	0.938	0.98	0.999	0.9999	–	–	–
	Cost($)	30	40	50	60	70	–	–	–
20	Reliability	0.85	0.99	0.999	0.9999	0.99998	0.999998	0.9999998	0.99999998
	Cost($)	20	40	65	80	100	120	140	155
21	Reliability	0.9	0.95	0.999	0.9999	0.99999	–	–	–
	Cost($)	25	30	50	70	90	–	–	–
22	Reliability	0.95	0.99	0.997	0.9997	0.99997	0.999997	0.9999997	0.99999997
	Cost($)	40	60	80	100	120	140	160	180
23	Reliability	0.85	0.995	0.999	0.9999	0.99999	–	–	–
	Cost($)	10	30	60	80	120	–	–	–
24	Reliability	0.9	0.95	0.995	0.9995	0.99995	0.999995	0.9999995	0.99999995
	Cost($)	30	50	70	90	110	130	150	170
25	Reliability	0.99	0.999	0.9999	0.99999	0.999999	0.9999999	–	–
	Cost($)	15	40	70	100	130	160	–	–

Table A.4. Data for example 4 (with 166 variables)