Solving a redundancy allocation problem with repairable identical components using simulation method and genetic algorithm

Amir Vahid Fakhredaie
Islamic Azad University, Dubai, UAE

Ayoub Veisi; Mani Sharifi
Islamic Azad University, Qazvin, Iran

Keywords
Reliability, Redundancy Allocation Problem, Standby Strategy, Monte Carlo Simulation, Genetic Algorithm.

Abstract
This article identifies and provides models for repairable RAP[1] in a series-parallel system and the capability of homogeneously allocating excess components. The objectives of the models are maximize reliability and maximize the MTTF[2] of the system. Since the objective function of these models has structural complexities, Monte Carlo simulation is used to calculate and extract these functions. The components failure and repair rates are considered to be constant and the policy of the system is cold standby. Two approaches, Perfect Computation and Genetic Meta-heuristics algorithm, are employed to calculate this model.

1. Introduction

Optimization is the art of selecting the best choice amongst alternatives. In every optimization problem there is at least one target function which has to be optimized. In this article, our objective is to independently optimize two target functions. Today, reliability optimization has numerous applications in almost all technical and engineering fields. Increasing competition and globalization of the markets has made reliability a pivotal focus of product design. One of the objectives of reliability is to design systems with high reliability. A comprehensive solution to optimal reliability problems has been provided by Kuo et al, (2001). In general, reliability of the whole system will be improved when through allocating redundant components the reliability of each of the components is improved. Redundancy allocation addresses simultaneous selection of components and system configuration, while accounting for all design limitations like system costs or reliability, when optimizing objective function (Coit and Smith, 1995). Using dynamic programming, Fyffe et al, (1968) were the pioneers in providing a mathematical model for redundancy allocation with the objective of maximizing system's reliability under weight and cost constraints. To find proper solutions to redundancy allocation, effective optimization algorithms, they utilized a variety of methods such as Dynamic Programming (Misra, 1972), Integer Programming (Sharma and Misra, 1990), and Genetic Algorithms (Goldberg, 1994). A comprehensive review of these approaches can be found in (Kuo et al, 2001) and (Shankar and Gururagan, 1993). Chern (1992) demonstrated that redundancy allocation problems can be classified as NP-hard (non-deterministic polynomial-time hard) and therefore various meta-heuristic can be employed to solve these problems. The system being studied in this text is a cold stand-by system. Using a sub-system with stand-by strategy needs components and equipment's that can recognize the error and activate redundant

[1] Redundancy Allocation Problem
[2] Mean Time To Failure

components, these equipment's are called switchovers. In practice, switchovers will provide a high degree of stability however; it is possible for switchovers to face errors. Shankar and Gururagan (1993) as well as Gurov and Utkin (1996) paid special attention to the problems of imperfect switchover.

Studies indicate that early improvements occurred right before the World War II. At that point in time, industries were not as modern and mechanized as they are today and errors and sudden break down of production machinery were causing major problems. In addition, most of the production equipment's had a relatively simpler design. Today, these equipment's are rather more sophisticated with extra attention being paid to the design of such products. The components of production machinery are designed to reparable. Many researchers are dedicating their studies to such systems. Yet, the majority of them keep holding the assumption that the intervals between errors as well as the intervals between repairing components are following exponential distribution.

Many products fall under the category of repairable systems. A repairable system is a system that can continue operating adequately after an error occurred in one or more of its components without the need of the whole system being replaced (Ascher and Feingold, 1987). Electronic and general machinery as well as communication equipment's are samples of repairable systems. This type of systems can properly continue working as long as they have one sound component. Serge et al, (1995) studied the reliability of a repairable system with periodic modifications. They provided a practical method to compute reliability parameters based on numerical functions. Later, an optimal geometric process model for cold standby repairable systems was presented (Zhang, 1999). In cold standby (policy) error rate of the standby components is zero. This means as long as a component is standby and is not functional within the system it will not face any errors. Seoa et al, (2003) estimated the lifetime and the reliability of systems with active redundancy and standby policy. They assumed that the faulty component will be repaired in minimum possible time. For this purpose they presented a simulation model to optimize reliability and define parameters on this basis. To estimate and optimize such problems methods like: a combination of genetic algorithms and Monte Carlo simulation (Marseguerra and Zio, 2000), discrete event simulation (Kurien, 1998), simulation modeling of repairable multi-component deteriorating systems has been employed.

The second part of this article is dedicated to describing the problem and mathematical modeling. In part 3 Monte Carlo simulation algorithm is explained. Part 4 defines and explains the solutions using examples. And Part 5 is dedicated to conclusion.

2. Problem definition

2.1. Nomenclatures

R(t):	System reliability in time (t)
λ:	Failure rate
μ:	Repair rate
n:	Number of components
m:	Number of repairers
C:	Maximum budget
c_1:	Cost of each piece of work
c_2:	Fixed cost of each repairer
c_3:	Time dependent cost of repairers
α:	Percentage of remaining budget

W;	Weight of total system
w:	Weight of each piece of work
p:	Switch functioning probability

2.2. Switching

When a standby redundant component is being used, not all the parallel components are being activated simultaneously in the circuit. At the start of operation, switchover connects the input to one of the parallel components. At this stage other parallel components remain standby in off mode. Switchover has the capability to distinguish between proper and improper behavior of components. At this point the switchover will function with probability p.

2.3. System Diagram

Following diagram depicts a system with n standby components and m repairers. According to this diagram, each component is repairable as long as the system is functioning and the system in these models has the capability of returning to its previous state. Components' failure rate is λ and the repair rate of components is μ. As the diagram shows, until a component fails each repairer works on one component. If we have more than one faulty component (as per exponential distribution characteristics, minimum adheres to exponential distribution) is the repair rate of each component. f is dysfunctional and irreparable system.

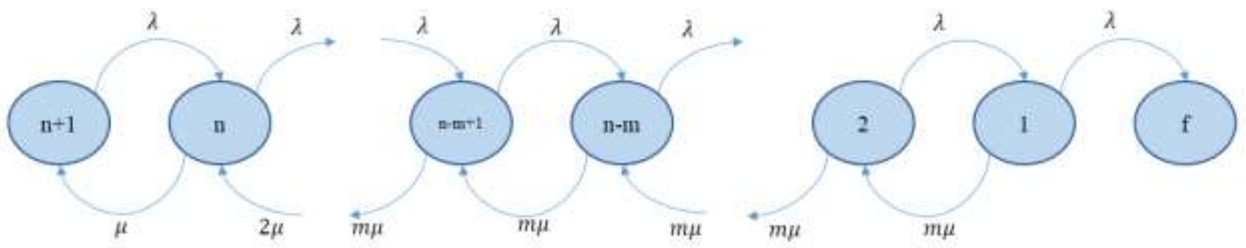


Figure 1. A system with n standby components and m repairers

2.4 System Equations

$$P'_{n+1}(t)+\lambda P_{n+1}(t)=\mu P_n(t)$$
$$P'_n(t)+(\lambda+\mu)P_n(t)=\lambda P_{n+1}(t)+2\mu P_{n-1}(t)$$
$$\vdots$$
$$P'_{n-m+1}(t)+(\lambda+m\mu)P_{n-m+1}(t)=\lambda P_{n-m+2}(t)+m\mu P_{n-m}(t)$$
$$P'_k(t)+(\lambda+m\mu)P_k(t)=\lambda P_{k+1}(t)+m\mu P_{k-1}(t) \quad ; \quad 2\le k\le m$$
$$P'_1(t)+(\lambda+m\mu)P_1(t)=\lambda P_2(t)$$
$$R(t)=\sum_{i=1}^{n+1}P_i(t)$$

2.5. Mathematical Model

The mathematical model of the problem is defined as follows:

$$Max \quad R(t) \qquad (1)$$
$$s.t:$$
$$nc_1+mc_2+tc_3\le C \qquad (2)$$
$$nw\le W \qquad (3)$$
$$n,m\in N$$

According to equation (1), the objective is defining the number of components, number of repairers, and the value of ɑ to maximize reliability. Limitations are given in equations (2) and (3) which are correspondingly related to the cost and weight of system.
The reliability given in equation (1) can be obtained by solving system of equations provided in 2.4 above. Since solving these equations is impossible, Monte Carlo simulation is utilized to obtain our objective function.

3. Simulation

As the objective function has structural complexities, Monte Carlo simulation is used to compute objective function. Monte Carlo is a method that uses random numbers to solve deterministic or even random problems. Before simulation random variables are being identified. In our system, components' lifetime and the length of repairing components by repairers are our random variables which in this article are assumed to be of exponential distribution.

3.1. Generating random numbers from exponential distributions

First the random number R$\in$ (0, 1) is generated and then we hold it equal to cumulative distribution function of exponential distribution to compute the corresponding variable. The obtained number will be a random number from the respective distribution.

$$T = -\frac{1}{\lambda} Ln(R) \quad ; \quad R \in [0,1]$$

To compute the objective simulation will be repeated several times for each of the answers (10 times in this article) and the average of calculated numbers will be considered as estimated objective function in each simulation. Each simulation will be carried out in the following manner:

Initial stage: First the lifetime of the whole system will be hold equal to zero. The number of working (sound) components will be hold equal to the total number of components purchased. The number of repairing components will be hold zero. The number of redundant repairers will be hold equal to the total number of employed repairers. Remaining budget will be hold equal to the total budget minus costs of purchasing components and cost of repairers (fixed cost of repairers).

After preparation stage, a random number from the distribution relating to the lifetime of components will be generated. After elapsing this period following operation will be performed:

1- The lifetime of the whole system will be added to the generated number
2- One unit will be deducted from the number of working components
3- In case repairer existed and when at least one repairer is idle, we dedicate one repairer and consequently deduct one unit from the number of idle repairers and add one unit to the number of components being repaired. If no idle repairer was available the faulty part has to wait until repairing of one of the faulty component is finished. Of course in this period we may face the failure of other components. If an idle repairer was available, one random number from the distribution associated with repair time will be generated. This random number will be considered as the time needed to repair this faulty component. We multiply this figure by variable cost of repair and will deduct the result from the total available budget. When the total remaining budget is less than zero the repair operation of the component will remain unfinished In this case, no budget will be left to dedicate to repairing the rest of the components and after all the components having failure the simulation will stop. But if the remaining budget was a figure greater than zero, after elapsing the generated random time (i.e. when the repair of the component is finished) one unit will be added to the number of working components. We can

face two distinct situations as the repair of component finishes. First: at that moment there will be no faulty component waiting for the repairer. In this case will increase the number of idle repairers by one. Second: as the repair finishes one or more faulty components are waiting for the repairer. In this case the number of idle repairers will not be increased and another random number from the distribution associated with repair time will be generated.

When the lifetime of a component elapses the switchover will, with probability P, function properly and the next component will be replaced in the system, consequently the above stages of 1 to 3 will be repeated. If the switchover, with probability 1-P, doesn't function then the whole system stops and the last number generated as the lifetime of this system will be considered as the lifetime of the system.

Notice: Each time, the simulation will continue to such point that either the switchover cease to function or the number of working components will turn into zero. When the remaining budget reaches zero, simulation does not stop. Only allocation of repairers to faulty components will stop. Thus the simulation will continue until every component is faulty or switchover stops working.

Notice: If during the time that one component is functioning, the repair of one of the faulty component finishes one unit to the number of working components as well as one unit to the number of repairers will be added.

Notice: It is possible that during the period that one component keeps functioning, the repair of several components finishes. Additionally, during the period that each component is being repaired several other components may face failure.

Notice: When the total budget is finishing, allocating repairers to faulty components may turn uneconomic and impractical. This happens because the high number of repairers allocated to repair jobs may accelerate finishing the budget and the repair of a considerable number of components will be left unfinished. As a result, after the budget reached a certain level (A percentage of the total budget minus purchasing cost which we call it alpha) we will stop allocating repairers to faulty components.

Notice: To compute MTTF we calculate the average of the ten thousand numbers obtained in each round of simulation. This average is MTTF.

Notice: to compute the probability of functioning up to the moment t, in each of the ten thousand simulations, the attained number will be compared to t. If the number is greater than t, the meaning is that the system has functioned up to moment t. We then divide the number of the times the system functioned up to moment t by the total number of simulations (10,000). The obtained number shows the probability of functioning up to moment t.

4. Solving methods

To find the solution two approaches, perfect computation and genetic algorithm are being used. As a reminder, a brief description of genetic algorithm will be provided. In addition, the objective is to maximize system's lifetime and reliability in 50 hours accounting for constraints $C_{Max} = 400{,}1000$

and $W_{Max} = 270$. The weight of each component $w = 9$ and reliability of switchover after 50 hours is equal to 0.995. Maximum number of allowed components in the system is set to 30.

4.1. Perfect computation method

Counts all the possible states and identifies the optimum outcome for different values of α. Considering the second limitation (weight limitation) the upper limit for the number of components equals to w/W which in this article is 30. And the upper limit for the number of repairers equals to $(n-1)$. We select the best outcomes with the corresponding α and presented along with the rest of the results in the following tables 1 and 2. (We considered $\alpha = 0,5,10,15,20,25,30$)

4.2. Genetic algorithm

4.2.1. A brief description about genetic algorithm

Genetic algorithm was introduced by John Holland (1976). Genetic algorithms are intelligent random optimization methods based on natural selection and genetic mechanisms. First equal to the number of population size initial answers will be generated either randomly or using heuristic methods and its fitness function will be computed. After that, using one of the selection strategies or methods chromosomes will be select to enter mating pool. Chromosomes inside mating pool are called parent chromosomes. During the next stage parent will be randomly selected in couples and genetic operators will be performing on them and a number of offspring are being produced. These offspring are being assessed against the fitness function and stop condition will be evaluated. If the condition exists the algorithm will be stopped otherwise the offspring will be selected to enter mating pool according to selection strategy. This cycle will continue until the condition is satisfied.

Crossover probability, mutation probability, population size, selection strategy, and genetic operators are needed to be defined at the start of the algorithm but they can later be altered.

4.2.2. Genetic algorithm

The results will be displayed in a one dimension array consisting of three bits. The first bit shows the number of components. The second bit represents the number of repairers. The third bid displays the level of remaining budget in percentage which in case the budget meets this level no more budgets will be dedicated to faulty components. This level of budget is called alpha. The problem chromosome presented in figure 1.

n	m	α

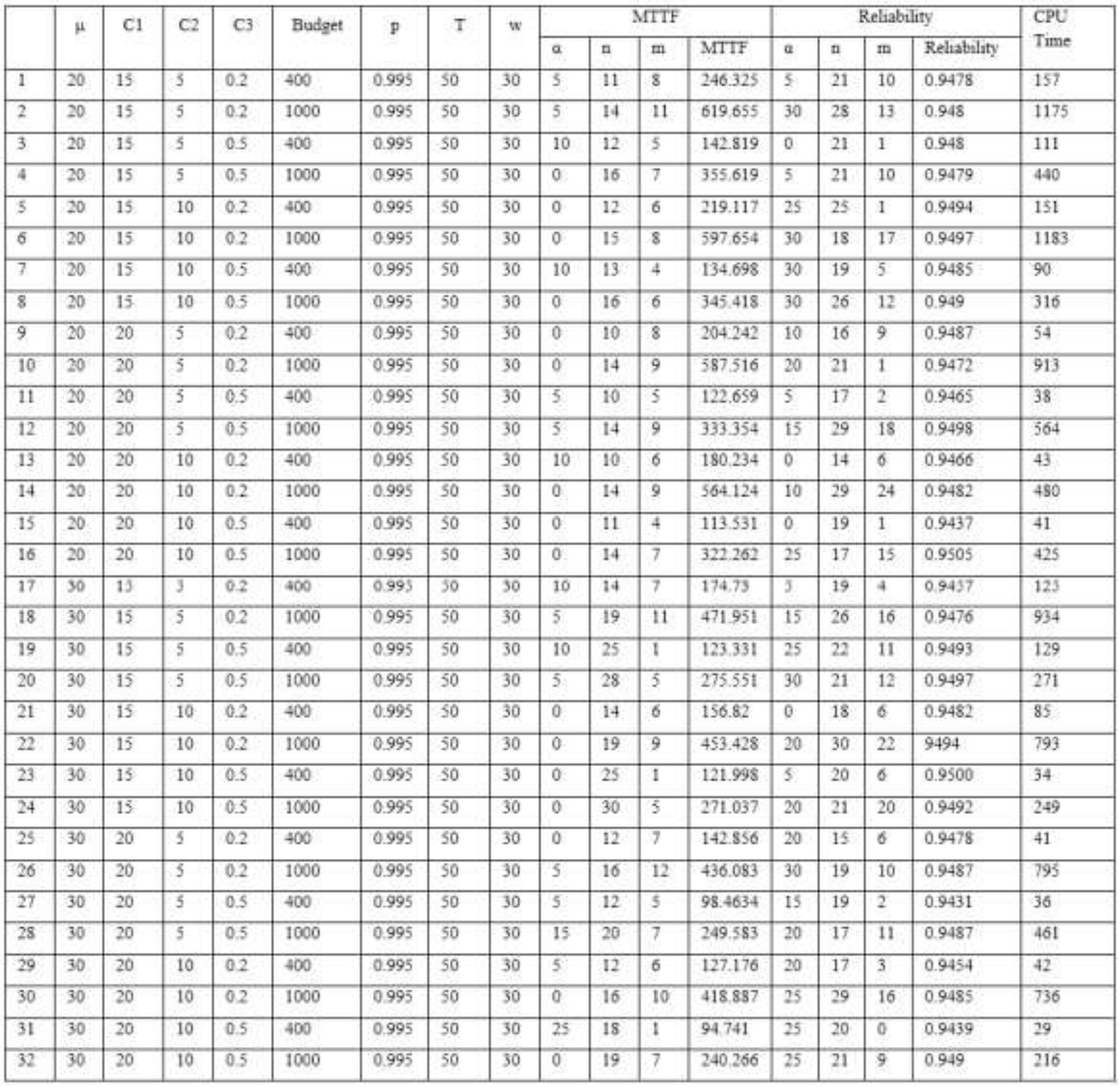
Figure 1: Chromosome of problem

4.2.3. Generating initial population

Generating initial population is completely random. To generate random answers, first the upper and lower limits have to be identified for each of the genomes and then random numbers will be generated from between these two limits. By dividing the total budget by the cost of each component, the upper limits of the number of system components will be defined and the lower limit is one. For this variable another upper limit exits as well and it is the weight of the system which can be converted to the limit of maximum number of system components. From these to limits whichever one is smaller will be selected as the final upper limit of the system.

Similarly, by dividing total budget by fixed cost of each repairer, the upper limit of the number of repairers can be extracted. In cease the number of repairers is equal to or greater than the number of components; the surplus will be practically of no use. For this reason and in order to limit the search criteria another upper limit will be considered for the number of repairers. This upper limit equals to number of components minus one. The minimum of these two limits will be considered as the final upper limit for the number of repairers. The lower limit of this variable is considered to be zero.

As per our experience, we consider the upper limit of alpha to be 30. Experience has showed that when values above 30 are assigned to alpha the resulted outcomes are not useful numbers. Zero is considered to be its lower limit. Alpha can be a fraction.



	μ	C1	C2	C3	Budget	p	T	w	MTTF				Reliability				CPU Time
									α	n	m	MTTF	α	n	m	Reliability	
1	20	15	5	0.2	400	0.995	50	30	5	11	8	246.325	5	21	10	0.9478	157
2	20	15	5	0.2	1000	0.995	50	30	5	14	11	619.655	30	28	13	0.948	1175
3	20	15	5	0.5	400	0.995	50	30	10	12	5	142.819	0	21	1	0.948	111
4	20	15	5	0.5	1000	0.995	50	30	0	16	7	355.619	5	21	10	0.9479	440
5	20	15	10	0.2	400	0.995	50	30	0	12	6	219.117	25	25	1	0.9494	151
6	20	15	10	0.2	1000	0.995	50	30	0	15	8	597.654	30	18	17	0.9497	1183
7	20	15	10	0.5	400	0.995	50	30	10	13	4	134.698	30	19	5	0.9485	90
8	20	15	10	0.5	1000	0.995	50	30	0	16	6	345.418	30	26	12	0.949	316
9	20	20	5	0.2	400	0.995	50	30	0	10	8	204.242	10	16	9	0.9487	54
10	20	20	5	0.2	1000	0.995	50	30	0	14	9	587.516	20	21	1	0.9472	913
11	20	20	5	0.5	400	0.995	50	30	5	10	5	122.659	5	17	2	0.9465	38
12	20	20	5	0.5	1000	0.995	50	30	5	14	9	333.354	15	29	18	0.9498	564
13	20	20	10	0.2	400	0.995	50	30	10	10	6	180.234	0	14	6	0.9466	43
14	20	20	10	0.2	1000	0.995	50	30	0	14	9	564.124	10	29	24	0.9482	480
15	20	20	10	0.5	400	0.995	50	30	0	11	4	113.531	0	19	1	0.9437	41
16	20	20	10	0.5	1000	0.995	50	30	0	14	7	322.262	25	17	15	0.9505	425
17	30	15	5	0.2	400	0.995	50	30	10	14	7	174.73	5	19	4	0.9437	123
18	30	15	5	0.2	1000	0.995	50	30	5	19	11	471.951	15	26	16	0.9476	934
19	30	15	5	0.5	400	0.995	50	30	10	25	1	123.331	25	22	11	0.9493	129
20	30	15	5	0.5	1000	0.995	50	30	5	28	5	275.551	30	21	12	0.9497	271
21	30	15	10	0.2	400	0.995	50	30	0	14	6	156.82	0	18	6	0.9482	85
22	30	15	10	0.2	1000	0.995	50	30	0	19	9	453.428	20	30	22	9494	793
23	30	15	10	0.5	400	0.995	50	30	0	25	1	121.998	5	20	6	0.9500	34
24	30	15	10	0.5	1000	0.995	50	30	0	30	5	271.037	20	21	20	0.9492	249
25	30	20	5	0.2	400	0.995	50	30	0	12	7	142.856	20	15	6	0.9478	41
26	30	20	5	0.2	1000	0.995	50	30	5	16	12	436.083	30	19	10	0.9487	795
27	30	20	5	0.5	400	0.995	50	30	5	12	5	98.4634	15	19	2	0.9431	36
28	30	20	5	0.5	1000	0.995	50	30	15	20	7	249.583	20	17	11	0.9487	461
29	30	20	10	0.2	400	0.995	50	30	5	12	6	127.176	20	17	3	0.9454	42
30	30	20	10	0.2	1000	0.995	50	30	0	16	10	418.887	25	29	16	0.9485	736
31	30	20	10	0.5	400	0.995	50	30	25	18	1	94.741	25	20	0	0.9439	29
32	30	20	10	0.5	1000	0.995	50	30	0	19	7	240.266	25	21	9	0.949	216

Table 1: The results of perfect computation method with $\alpha = 5$

4.2.4. Selection operator

Selection of parents for mutation or crossover operation is done using a method named tournament. According to this method, first a number of parents equal to the tour Size will be randomly selected. Then, among the parents whoever has the best fitness function will be selected. In this article the tour Size is 3.

	μ	C1	C2	C3	Budget	p	T	w	MTTF				Reliability				CPU Time
									α	n	m	MTTF	α	n	m	Reliability	
1	20	15	5	0.2	400	0.995	50	30	5	9	5	564.945	0	17	6	0.9719	90
2	20	15	5	0.2	1000	0.995	50	30	0	11	6	1297.04	10	21	11	0.9709	565
3	20	15	5	0.5	400	0.995	50	30	0	10	4	302.794	20	14	6	0.9727	98
4	20	15	5	0.5	1000	0.995	50	30	0	12	5	729.892	10	28	6	0.9728	624
5	20	15	10	0.2	400	0.995	50	30	0	9	4	525.729	15	20	0	0.9724	124
6	20	15	10	0.2	1000	0.995	50	30	5	12	5	1268.17	20	27	1	0.9713	971
7	20	15	10	0.5	400	0.995	50	30	5	10	3	290.469	0	12	2	0.9721	71
8	20	15	10	0.5	1000	0.995	50	30	0	13	4	715.276	0	24	20	0.9714	579
9	20	20	5	0.2	400	0.995	50	30	5	8	5	495.326	30	13	8	0.9734	67
10	20	20	5	0.2	1000	0.995	50	30	0	10	7	1241.9	15	28	11	0.9731	973
11	20	20	5	0.5	400	0.995	50	30	10	8	4	269.964	25	17	4	0.9714	45
12	20	20	5	0.5	1000	0.995	50	30	0	10	6	700.239	25	27	17	0.9733	498
13	20	20	10	0.2	400	0.995	50	30	0	8	5	457.092	20	13	2	0.9724	60
14	20	20	10	0.2	1000	0.995	50	30	0	10	6	1219.07	15	22	8	0.972	457
15	20	20	10	0.5	400	0.995	50	30	10	8	4	254.285	0	13	7	0.9733	38
16	20	20	10	0.5	1000	0.995	50	30	5	11	5	679.875	0	20	2	0.9718	501
17	30	15	5	0.2	400	0.995	50	30	0	10	6	403.222	20	21	3	0.9729	132
18	30	15	5	0.2	1000	0.995	50	30	0	12	7	1002.12	10	30	26	0.9723	894
19	30	15	5	0.5	400	0.995	50	30	5	24	1	246.468	5	24	1	0.9714	82
20	30	15	5	0.5	1000	0.995	50	30	10	25	3	558.518	25	17	12	0.9723	252
21	30	15	10	0.2	400	0.995	50	30	10	10	5	371.672	5	16	3	0.972	93
22	30	15	10	0.2	1000	0.995	50	30	0	14	6	967.014	30	16	12	0.9704	839
23	30	15	10	0.5	400	0.995	50	30	25	24	1	243.439	0	12	7	0.972	52
24	30	15	10	0.5	1000	0.995	50	30	20	30	3	552.091	5	20	19	0.9718	241
25	30	20	5	0.2	400	0.995	50	30	0	9	6	347.106	5	13	10	0.973	62
26	30	20	5	0.2	1000	0.995	50	30	0	12	8	944.347	5	23	9	0.9721	787
27	30	20	5	0.5	400	0.995	50	30	20	9	5	206.339	5	16	7	0.9709	33
28	30	20	5	0.5	1000	0.995	50	30	5	13	6	516.904	20	20	3	0.9739	406
29	30	20	10	0.2	400	0.995	50	30	0	9	5	317.561	15	11	9	0.9721	45
30	30	20	10	0.2	1000	0.995	50	30	5	13	7	913.106	25	19	15	0.9713	724
31	30	20	10	0.5	400	0.995	50	30	15	13	2	197.432	30	12	11	0.9722	27
32	30	20	10	0.5	1000	0.995	50	30	0	15	4	502.658	15	22	17	0.9726	373

Table 2: The results of perfect computation method with $\alpha = 10$

4.2.4.1. Crossover operator

The crossover selected in this article is a sequential crossover operator. This operator initially generates a crossover mask. Mask chromosome is simply a binary array which is of the same size of other chromosomes. Binary digits in crossover mask are generated randomly. Conventionally for zeros in the mask, corresponding genome of parent 1 will be transferred to the child and fro ones in the mask parent 2 will be copied. The second child will be produces in the same manner. For the second genome we need to pay attention to what explained in the above paragraph marked with **. In order to limit search criteria we want to avoid searching outcomes in which the number of repairers is equal to or greater than the number of components. For this reason during crossover operation in second genome (genome corresponding to the number of repairers) if through inheriting the genome from a parent the number of repairers becomes equal to or greater than the number of components then the

genome of that parent will not be inherited by the child. In this case the child will inherit the genome from the other parent.
The sequential crossover operator presented in figure 2.

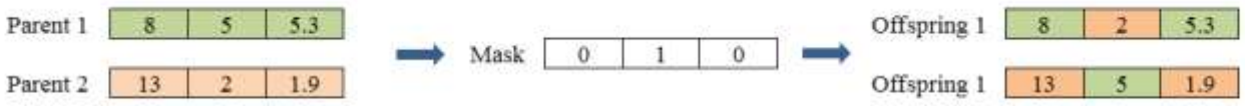


Figure 2: Crossover operator

4.2.4.2. Mutation operator

With this operator, first a genome will be randomly selected and then will be mutated. If the selected genome was corresponding to number of components or number repairers, then the content of the genome will be automatically increased or decreased by one unit (should not exceed upper or lower limits). In case the selected genome corresponds to alpha the content of the genome will be automatically increased or decreased by 0.1. In mutation operator as well. For this reason if the second genome will be selected to mutate and in this genome the number of repairers will equal to the number of components minus one, then the content of the genome can only decrease. The mutation operator illustrated in figure 3.

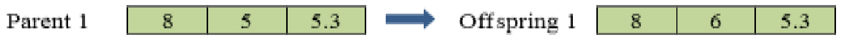


Figure 3: Mutation operator.

4.2.5. Penalty function

It is possible that crossover or mutation operators generate answers contradictory to the maximum available budget limit. In case the sum of components purchasing cost and the repairers fixed cost exceed the maximum available budget the limit will be contradicted. Since in such a case there will be no budget left to cover variable cost of repairers, the repair operation will never be practicable.
To penalize unjustifiable answers, we multiply the objective function of the unjustified answers by K (f=f×k) and the following formula is obtained:

$$k = \frac{1}{[D+1]^2}$$

In this formula D is the degree of exceeding the limit which can be calculated using the following formula:

$$D = n \times c_1 + m \times c_2 - k$$

The results of solving the presented problems are in tables 3 and 4.

5. Conclusion and further studies

In this article, using Monte Carlo simulation to compute objective functions, models for the problem of cold standby repairable redundancy allocation are presented. To reach the solution two different methods, perfect computation and genetic algorithm are utilized. Finally, in order to evaluate the performance of proposed methods, several numerical exampled were solved through using these methods. As these examples illustrate, by increasing the intervals between the failures, reliability and MTTF have considerably increased. Concerning the proposed methods for reliability and MTTF, genetic algorithm had a better performance comparing to perfect computation method. Genetic algorithm required shorter time to implement for reliability. But in problems of small size, MTTF can in some cases be solved faster using perfect computation. In general genetic algorithm had a better performance in this article and is recommended for further work.

	μ	C1	C2	C3	Budget	MTTF							Reliability						
						α	n	m	$MTTF_{max}$	CPU	$\overline{MTTF}$	σ	α	n	m	R_{max}	CPU	$\overline{Re}$	σ
1	20	15	5	0.2	400	5.1	11	8	247.03	163	245.28	1.144	6.4	17	8	0.9484	45	0.9472	0.0013
2	20	15	5	0.2	1000	1.6	14	9	625.46	493	619.37	4.534	26.2	19	21	0.9509	747	0.9484	0.0018
3	20	15	5	0.5	400	5	12	5	143.05	52	142.73	0.271	1.5	24	0	0.9501	33	0.9480	0.0012
4	20	15	5	0.5	1000	1.1	15	8	355.75	263	354.80	1.101	28.8	27	16	0.9536	232	0.9471	0.0038
5	20	15	10	0.2	400	3.8	12	6	219.98	77	219.65	0.428	18.4	22	0	0.9498	35	0.9477	0.0023
6	20	15	10	0.2	1000	0.9	15	8	603.59	476	601.09	2.710	23	30	13	0.9477	302	0.9472	0.0002
7	20	15	10	0.5	400	0.1	13	4	134.4	52	134.05	0.327	18.2	22	5	0.9491	33	0.9468	0.0017
8	20	15	10	0.5	1000	1.2	16	6	346.285	362	345.00	0.883	20	25	18	0.9492	105	0.9474	0.0015
9	20	20	5	0.2	400	0.5	10	7	204.397	127	203.84	0.726	3.9	14	12	0.9507	52	0.9492	0.0019
10	20	20	5	0.2	1000	1.2	14	10	592.454	777	589.93	2.451	25.9	16	12	0.9501	178	0.9478	0.0015
11	20	20	5	0.5	400	3.9	10	6	123.08	68	122.66	0.295	20.4	16	5	0.9494	20	0.9464	0.0014
12	20	20	5	0.5	1000	4.3	14	8	333.89	274	332.64	1.197	0	22	20	0.9506	136	0.9484	0.0013
13	20	20	10	0.2	400	3.7	10	6	180.891	193	180.59	0.285	27.7	17	3	0.9482	31	0.9467	0.0012
14	20	20	10	0.2	1000	1.7	14	8	572.503	324	565.68	4.507	3.8	25	1	0.9495	101	0.9483	0.0010
15	20	20	10	0.5	400	14.3	11	4	113.906	62	113.19	0.948	29	19	1	0.9483	15	0.9483	0.0029
16	20	20	10	0.5	1000	3.7	14	7	323.969	196	321.05	2.271	17.6	20	14	0.9508	60	0.9483	0.0002
17	30	15	5	0.2	400	10.1	13	8	176.34	150	175.13	0.908	14.8	22	3	0.9505	27	0.9479	0.0020
18	30	15	5	0.2	1000	4	18	11	476.358	725	472.84	2.293	9.6	17	16	0.9493	241	0.9474	0.0018
19	30	15	5	0.5	400	20.6	26	1	123.256	28	122.68	0.602	18.8	19	4	0.9509	21	0.9478	0.0019
20	30	15	5	0.5	1000	10.4	29	5	276.793	232	275.76	0.784	14.1	25	4	0.9503	58	0.9482	0.0013
21	30	15	10	0.2	400	3.6	13	6	157.732	112	157.07	0.638	25.1	22	6	0.9488	40	0.9475	0.0011
22	30	15	10	0.2	1000	3.2	19	9	454.274	471	452.65	1.285	15.8	29	28	0.9505	228	0.9488	0.0014
23	30	15	10	0.5	400	2.4	26	1	122.26	29	121.91	0.407	3.3	24	2	0.9506	18	0.9486	0.0013
24	30	15	10	0.5	1000	8.6	29	5	271.073	179	266.88	6.362	15.9	25	18	0.9499	101	0.9487	0.0009
25	30	20	5	0.2	400	4.7	11	8	144.239	101	143.56	0.632	10.7	15	8	0.948	27	0.9458	0.0023
26	30	20	5	0.2	1000	0.1	16	11	440.021	265	438.93	0.710	10.6	22	19	0.9504	210	0.9486	0.0030
27	30	20	5	0.5	400	10.3	13	4	98.6588	31	98.26	0.314	24	19	1	0.9478	35	0.9424	0.0033
28	30	20	5	0.5	1000	8.3	17	9	250.538	155	249.26	0.930	15	29	11	0.9492	81	0.9467	0.0007
29	30	20	10	0.2	400	9.7	11	6	127.665	55	127.18	0.553	18.1	15	6	0.949	35	0.9452	0.0025
30	30	20	10	0.2	1000	0.9	16	10	419.924	200	415.93	2.765	25.4	30	25	0.9487	302	0.9476	0.0014
31	30	20	10	0.5	400	3.1	16	2	94.8599	28	94.66	0.345	19.7	19	1	0.9473	23	0.9447	0.0018
32	30	20	10	0.5	1000	7.3	18	7	241.518	209	240.57	0.553	2.5	30	20	0.95	84	0.9476	0.0015

Table 3: The results of genetic algorithm with $\alpha = 5$

	μ	C1	C2	C3	Budget	MTTF							Reliability						
						α	n	m	$MTTF_{max}$	CPU	$\overline{MTTF}$	σ	α	n	m	R_{max}	CPU	$\overline{Re}$	σ
1	20	15	5	0.2	400	4.5	9	5	567.297	178	566.09	1.349	2.5	12	7	0.9724	34	0.9710	0.0014
2	20	15	5	0.2	1000	2.1	11	7	1301.04	141	1299.3	1.420	14	21	14	0.9722	336	0.9709	0.0009
3	20	15	5	0.5	400	7.3	9	4	304.471	45	303.64	0.608	8.8	18	14	0.9736	30	0.9714	0.0008
4	20	15	5	0.5	1000	1.4	12	5	735.54	230	728.50	5.280	16.4	18	6	0.973	296	0.9715	0.0016
5	20	15	10	0.2	400	3.8	9	4	531.194	86	527.41	2.398	9.1	13	4	0.9743	116	0.9712	0.0022
6	20	15	10	0.2	1000	0.8	12	5	1277.38	453	1266.5	6.094	10.7	18	16	0.9723	116	0.9717	0.0005
7	20	15	10	0.5	400	9.1	10	3	290.397	92	289.84	0.521	18.2	16	5	0.9725	29	0.9707	0.0007
8	20	15	10	0.5	1000	6.7	12	4	717.602	192	715.56	1.622	3.1	30	3	0.9738	73	0.9723	0.0012
9	20	20	5	0.2	400	3.6	8	5	499.22	136	496.59	2.686	25.2	13	6	0.9736	77	0.9709	0.0016
10	20	20	5	0.2	1000	0.8	10	8	1253.97	255	1250.1	3.802	1.4	20	3	0.9721	134	0.9711	0.0005
11	20	20	5	0.5	400	7.2	8	4	271.711	63	270.17	1.060	23.3	18	8	0.9728	23	0.9714	0.0008
12	20	20	5	0.5	1000	3.9	10	7	702.137	109	697.97	3.110	17.1	13	12	0.9745	98	0.9726	0.0012
13	20	20	10	0.2	400	7.2	8	5	458.235	110	456.50	1.599	18.3	15	7	0.9736	46	0.9723	0.0020
14	20	20	10	0.2	1000	3.8	11	5	1219.65	305	1216.5	3.664	2.6	24	9	0.9724	212	0.9720	0.0013
15	20	20	10	0.5	400	13.2	9	3	255.015	77	254.39	0.633	0.4	16	3	0.9735	23	0.9717	0.0004
16	20	20	10	0.5	1000	1	11	5	683.25	126	680.72	2.826	9.6	18	0	0.9737	114	0.9729	0.0008
17	30	15	5	0.2	400	6.5	10	6	404.559	111	403.54	0.939	0.4	12	10	0.9713	29	0.9710	0.0012
18	30	15	5	0.2	1000	1.1	13	7	1006.69	372	1003.2	5.824	20.4	23	7	0.9731	130	0.9715	0.0008
19	30	15	5	0.5	400	1.7	24	1	247.485	29	246.79	0.522	20.4	11	9	0.9721	16	0.9707	0.0012
20	30	15	5	0.5	1000	8.7	29	4	559.675	86	557.68	1.388	6	20	12	0.9729	142	0.9716	0.0014
21	30	15	10	0.2	400	4.6	10	5	373.282	74	370.32	2.235	16.5	14	12	0.9738	26	0.9720	0.0019
22	30	15	10	0.2	1000	1.9	13	6	981.729	212	977.38	4.758	28.6	28	1	0.9714	153	0.9709	0.0006
23	30	15	10	0.5	400	24.2	26	1	243.862	51	243.22	0.526	28.5	16	13	0.9725	30	0.9701	0.0015
24	30	15	10	0.5	1000	2.9	27	3	553.715	153	550.22	2.640	5.3	15	6	0.9723	67	0.9711	0.0011
25	30	20	5	0.2	400	0.9	9	6	350.499	99	349.29	0.891	4	13	12	0.9742	36	0.9709	0.0019
26	30	20	5	0.2	1000	1.7	12	8	952.649	230	945.61	6.465	5.8	26	14	0.9738	77	0.9723	0.0014
27	30	20	5	0.5	400	4	10	4	206.779	39	205.78	0.587	17.2	15	11	0.9736	24	0.9705	0.0017
28	30	20	5	0.5	1000	6.4	13	6	518.796	192	516.73	2.087	7.6	22	16	0.9740	54	0.9726	0.0010
29	30	20	10	0.2	400	2.7	9	5	320.37	25	318.96	1.124	13.2	19	2	0.9734	39	0.9713	0.0018
30	30	20	10	0.2	1000	1.6	13	6	922.05	253	918.24	3.883	1	19	9	0.9728	190	0.9714	0.0019
31	30	20	10	0.5	400	3.7	13	2	197.912	30	197.52	0.250	23.1	13	10	0.9728	17	0.9723	0.0003
32	30	20	10	0.5	1000	5.8	13	5	505.392	135	504.41	1.292	2	27	18	0.9728	87	0.9721	0.0008

Table 4: The results of genetic algorithm with $\alpha = 10$

6. References:

Ascher, H., & Feingold, H. (1987). Repairable system reliability. New York: Marcel Dekker Inc.

Chern MS.On the computational complexity of reliability redundancy allocation in a series system. Oper Res Lett 1992;11: 309-15.

Coit DW, Smith A. Optimization approaches to the redundancy allocation to the redundancy allocation problem for series-parallel systems. Proceedings of the fourth industrial engineering research conference, Nashville, TN, Nay 1995.

Fyffe, D.E., Hines, W.W. and Lee, N.K., "System Reliability Allocation and Computational Algorithm", IEEE Transactions on Reliability, (1968); Vol. 17, pp.64-69,

Goldberg D. Les algorithms genetiques. Paris: Addison-Wesley; 1994.

Gurov,S.V. and Utkin,L.V.Cold standby systems with imperfect and non-instaneous switchover mechanism. Microelectronics and reliability 1996,36(10):1425-1438.

J.H. Seoa, J.S. Jangb, D.S. Baia, Lifetime and reliability estimation of repairable redundant system subject to periodic alternation , Ajou University, Suwon 442-479, South Korea, (2003) 197–204;

Kuo, W., Prasad, V. R., Tillman, F. A., Hwang, C. 2001. Optimal reliability design fundamental and application. London: Cambridge University Press.

Kurien , KC., Reliability and availability analysis of repairable system using discrete event simulation, (1988), Ph. D. thesis, IIT, New Delhi

Marseguerra M, Zio E. Optimizing maintenance and repair policies via a combination of genetic algorithms and Monte Carlo simulation.*20133 Milano, Italy* (2000) 69–83

Misra KB. Reliability optimization of series-parallel system. IEEE Transaction on Reliability.1972; 21(21):230-8.

Serge V. Gurov and Levv. Utkin, Reliability of repairable systems with periodic modifications, Institutski per. 5., St Petersburg, 194018 Russia(1995);

Shankar, B.K. and Gururagan,M. Tow-unit cold standby system with imperfect repair and excessive availability period.Microelectronics and reliability1993,33(4):509-512.

Sharma U, Misra KB.An efficient algorithm to solve integer programming problems in reliability optimization.Int J Qual Reliab Manage 1990;7(5):44-56.

Zhang YL. An optimal geometric process model for a cold standby repairable system. Reliability Engineering and System Safety (1999);63:107–10.