Comparing Parallel Simulated Annealing, Parallel Vibrating Damp Optimization and Genetic Algorithm for Joint Redundancy-Availability Problems in a Series-Parallel System with Multi-State Components

Mani Sharifi[a,*], Morteza Mousa Khani[b], Arash Zaretalab[c]

[a] *Assistant Professor, Faculty of Industrial & Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[b] *Associate Professor, Faculty of Management & Accounting, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[c] *MSc, Faculty of Industrial & Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

Received 10 October, 2012; Revised 19 March, 2013; Accepted 20 May, 2013

Abstract

In this paper, we study different methods of solving joint redundancy-availability optimization for series-parallel systems with multi-state components. We analyzed various effective factors on system availability in order to determine the optimum number and version of components in each sub-system and consider the effects of improving failure rates of each component in each sub-system and improving reliability of each sub-system. The target is to determine optimum values of all variables for improving the availability level and decreasing the total cost of the system. At first, the exact values of variables are determined using a mathematical model; then, the results of SA-Parallel, VDO-Parallel and genetic algorithms are compared with the exact solution.

Keywords: SA; VDO; Parallel; Availability; Multi-state; Redundancy.

1. Introduction

The effective factors in system availability in general include component types in each sub-system, the number of components in each sub-system and the level of improving failure rates for each component in each sub-system, as well as improving reliability of each sub-system. The third factor concerns decreasing the failure rate of each component and increasing the reliability of subsystems by spending money. For example, if we consider more skilled workers in each subsystem, the reliability of the sub-system and in turn its availability will increase.

In traditional models, the components and subsystems have only two states: working, not working (failed). But it is obvious that they may be in the states between the two above states. These states are caused by slight failures and operational defects. These systems are called multi-state systems (MSS) (Barlow & Wu, 1987; Boedigheimer et al, 1994; Lisianaski et al, 2003; Zuo et al, 2006; Zuo et al, 2007).

In this regard, efforts have been made to optimize the level of redundancy in the various subsystems of a system. Levitin et al (1998) developed a model to determine the optimal version of subsystem components in a multi-state series-parallel system. Coit and Ramirez Marquez (2004) introduced a heuristic method for multi-state Redundancy Allocation Problem (RAP), and Tian and Zuo (2006) provided a method based on physical programming and genetic algorithm for multi-state RAP. In both models, RAP was the only factor in the improvement of system operation, but Tian et al (2009) introduced a modern method for optimizing series-parallel systems. In doing so, they provided two options for improving system reliability and availability: optimal redundancy allocation in various subsystems and improving the reliability of components through affecting functional rates. Furthermore, Tian et al (2005) presented a joint reliability-redundancy optimization method for multi-state series-parallel systems inspired by the method described above. In their method, transition rates affect component state distributions and redundancy is considered as a variable of the problem. After Chern (1972) proved that RAP belongs to NP hard problems, Mehta heuristic algorithms are used increasingly in the area. In this regard, Gen and Kim (1999) used a hybrid genetic algorithm to optimize reliability problems and Konak et al (2003) took advantage of using the tabu search for RAP. Liang et al (2004) solved RAP using the

[*] Corresponding author E-mail: mani.sharifi@yahoo.com

ant colony algorithm while Chen and Liang (2007) solved series-parallel RAP by VNS. Moreover, Lisnianski et al. (2008) described Universal Generating Function (UGF) for RAP in fuzzy and crisp situations.
In the same vein, in the present paper we use Simulated Annealing (SA) -parallel, Vibrating Damp Optimization (VDO) -parallel and genetic algorithms for solving series-parallel RAP as we calculate system availability by UGF. Comparing the results with other algorithms, it seems that the presented algorithms are effective.

Nomenclature

n :	Number of sub-systems
n_i :	Number of component(s) in sub-systems i
H_i :	Number of versions in sub-systems i
n_{ih} :	Number of components for version h in sub-system i
M_i :	Maximum of components variety in sub-system i
M_{si} :	Maximum states in subsystem i
M :	Maximum states for series-parallel system
λ_{ji} :	Failure rate from state j to state i
μ_{ji} :	Repair rate from state j to state i
$p_i(t)$:	The probability that component is in i in time T
k_i :	Number of technical and organizational activities in sub-system i
k_{si} :	Number of available organizational activities in sub-system i
Tk^i_{hk} :	A binary variable which equals 1 when technical activity k in sub-system i for version h is done
Tk^i_k :	A binary variable which equals 1 when organizational activity k in sub-system i is done
X_{ih} :	Vector for component version i in sub-system i
X :	Variables vector
$a_{k,j,l}$:	Represented factor for effects of activity k on transition rate from state j to state i
g^i_{hi} :	Operation rate according to state i for component version j in sub-system i
$U^i(Z)$:	U function for sub-system i
$U(Z)$:	U function for series-parallel system
$\otimes_\phi$:	UGF operator
A :	System availability
A_0 :	System availability constraint
D :	Demand level
C^i_h :	Unit cost for organizational activity on component version h for sub-system i
C^{Ti}_{kh} :	Unit cost for technical activity k on component version h for sub-system i
C^{Ti}_{kh0} :	Total cost for technical activity k on component version h for sub-system i
C^i_{com} :	Cost of sub-system i components
C^i_{act} :	Cost of sub-system i activities
C^i :	Cost of subsystem i
$C(X)$:	System cost
UGF :	Universal generation function
GA :	Genetic algorithm

The remainder of this paper is organized as follows. Section 2 describes the system definition under study along with the related mathematical model. Section 3 presents the mathematical model. Section 4 presents the proposed solution procedures. The experimental design and the numerical example appear in this Section. Finally, the paper is concluded in Section 5.

2. System Definition

2.1. Assumptions

The assumptions of this paper regarding multi-state systems are as follows:

- Components are independent,
- In different states, components have different operational rates,
- Operation rate of a parallel sub-system is equal to the sum of operation rates of all components,
- Operation rate of a series-parallel system is equal to minimum operation rates of sub-systems.

2.2. State diagram and Marcovian model:

In multi-state systems, the first thing to calculate is the state probability distribution that depends on failure and repair rates of components. The transition rates can be estimated by the historical data obtained from past failure and repair times. In this paper, we concentrate on the differential equation obtained from Marcovian model. Random process models such as Marcov and semi Marcov models can be used to illustrate a multi-state component as shown in Figure1.

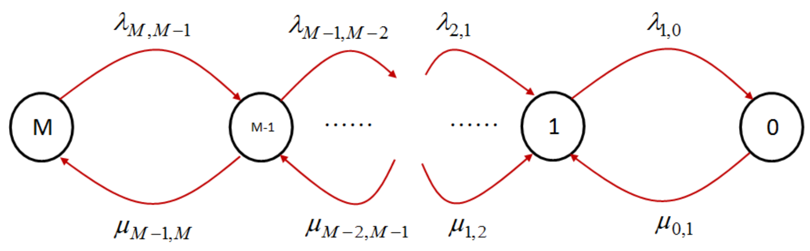


Fig.1. The state space diagram for a repairable system.

Assume that each component has $(M+1)$ state(s) and transitions are possible between two adjacent states. So a system of differential equation must be solved to determine the state probability distributions at the time t. The systems of differential equations are as follows:

$$\begin{aligned}
&\frac{d\ p_0(t)}{dt}=-\mu_{0,1}\ p_0(t)+\lambda_{1,0}\ p_1(t)\\
&\frac{d\ p_1(t)}{dt}=\mu_{0,1}\ p_0(t)+\\
&\lambda_{2,1}\ p_2(t)-(\lambda_{1,0}+\mu_{1,2})\ p_1(t)\\
&\ \vdots \qquad\qquad \vdots\\
&\frac{d\ p_{M-1}(t)}{dt}=\mu_{M-2,M-1}\ p_{M-2}(t)+\\
&\lambda_{M,M-1}\ p_M(t)-(\lambda_{M-1,M-2}+\mu_{M-1,M})\ p_{M-1}(t)\\
&\frac{d\ p_M(t)}{dt}=\mu_{M-1,M}\ p_{M-1}(t)-\lambda_{M,M-1}\ p_M(t)
\end{aligned} \tag{1}$$

And the initial conditions are:

$$\begin{aligned}
&p_i(0)=0 \quad ; \quad i=1,2,\dots,M-1\\
&p_M(0)=1
\end{aligned} \tag{2}$$

Now it is necessary to discuss the effects of technical and organizational activities on the system reliability. When technical or organizational activities are done, the transition rate changes in order to improve the system availability. Generally, with these activities, the failure rate decreases and the repair rate increases. Assume that TK_k is a binary variable showing whether activity k is done or not. Now to determine the new values of transition rate, we have:

$$if \quad TK_k=1 \quad :\begin{cases}\lambda_{ij}=a_{k,i,j}\times\lambda_{ij}\\ \mu_{ji}=a_{k,j,i}\times\mu_{ji}\end{cases} \tag{3}$$

In this paper two different kinds of activities are considered:

- Activities at the component level called technical activities,
  - Activities at the sub-system level called organizational activities.

TK_{hk}^{i} is the technical activity that presents activity k on component version h on sub-system i. The cost of this type of activities depends on the number of affected components. TK_k^{i} is the organizational activity that presents activity h on sub-system i and affects all components in sub-system i. The cost of this kind of activity is independent of the number of sub-system components.

2.3. System variables

For each sub-system, different versions of components are available. We consider that for a specific sub-system, different versions of components have the same available states. Each version of components is defined and categorized by operation rates of different states, transition rates and unit costs.
Each sub-system can have different kinds of components. In addition to the number of components, the type of components in sub-systems must be determined. The maximum variety of component types and component versions in sub-system i is H_i. The number of component version h on sub-system i is n_{ih}. So for sub-system i, $(1\le i\le N)$ the variables $(n_{i1},n_{i2},\dots,n_{i,H_i})$ must be determined.
In addition, vector $(TK_{h1}^{i},TK_{h2}^{i},\dots,TK_{hk_i}^{i})$ is used to illustrate the set of technical activities done on components version h. TK_{hk}^{i} is a binary variable. If activity k for sub-system i is performed for components version h then $TK_{hk}^{i}=1$, otherwise $TK_{hk}^{i}=0$.

$$X_{ih}=(n_{ih},TK_{h1}^{i},TK_{h2}^{i},\dots,TK_{hk_i}^{i})$$

The system design variables vector related to components version h in sub-system i, $(1\le i\le N, 1\le h\le H)$ shown as X_{ih} is as follows:

$$X_{ih}=(n_{ih},TK_{h1}^{i},TK_{h2}^{i},\dots,TK_{hk_i}^{i}) \tag{4}$$

To illustrate the set of technical activities on sub-system i, we use $(TK_{k_1}^{i},TK_{k_2}^{i},\dots,TK_{k_{si}}^{i})$. In this vector,

$TK_{hk}^{i}, \left(1 \le i \le N, 1 \le k \le k_i + k_{si}\right)$ is a binary variable. If activity i is performed on the sub-system, then $TK_k^i = 1$, otherwise $TK_k^i = 0$.

So, for a series-parallel multi-state system with N sub-system and H_i version of component of sub-system i, the system design vector is as follows:

$$X = \begin{pmatrix} X_{11},\dots,X_{1H_1},TK_{k_1+1}^{1},\dots, \\ TK_{k_1+k_{s_1}}^{1},X_{21},\dots,X_{2H_2},TK_{k_2+1}^{2},\dots,TK_{k_2+k_{s_2}}^{2}, \\ \dots,X_{N1},\dots,X_{NH_N},TK_{k_N+1}^{1},\dots,TK_{k_N+k_{s_N}}^{1} \end{pmatrix} \tag{5}$$

In a vector of variables, we can identify the availability of a multi-state system as follows:

- For each version of components in a sub-system, identify the transition rate based on technical and organizational activities,
- For each version of components in a sub-system, identify state distributions based on transition rates using Marcov model,
- Identify the system operation distribution based on the state distribution using UGF.

2.4. *System cost:*

$$\begin{aligned} C^i &= C_{com}^i + C_{act}^i \\ C_{com}^i &= C_0^i + \sum_{h=1}^{H_i} n_{ih}\, c_h^i \\ C_{act}^i &= \sum_{k=1}^{K_i} TK_{hk}^i \left(c_{kh0}^{Ti} + n_{ih}\, c_{kh}^{Ti}\right) + \sum_{k=k_i+1}^{K_i+K_{si}} TK_k^i\, c_{k0}^{Ti} \end{aligned} \tag{6}$$

The total cost of the system is equal to the sum of sub-system costs as follows:

$$C(X) = \sum_{i=1}^{N} C^i \tag{7}$$

2.5. *UGF method*

In 1986, Ushakov (1987) presented the general concepts of universal generation function used for the system reliability evaluation. Taking it a step further, Levitin and Lisnianski made some studies of using UGF for analyzing series-parallel and parallel-series multi-state systems (Levitin and Lisnianski et al.1996).

With regard to multi-state systems, it should be pointed out that when we study a multi-state component, for each performance mode of component a probability is considered. This increases the system performance state, but the classic method cannot be used to calculate the system reliability. Therefore, to calculate the system reliability and availability, there was a tendency for using mathematical methods. Using the UGF method for analyzing a multi-state system decreases the system state very much [17]. We use the UGF method to calculate the system availability as follows:

$$\begin{aligned} U_h^i(z) &= p_{h,0}^i(t) z^0 + p_{h,1}^i(t) z^{g_{h,1}^i} + \dots + p_{h,M_i}^i(t) z^{g_{h,M_i}^i} \\ U^i(z) &= \otimes_\phi \left\{u_1^i(z), u_2^i(z), \dots, u_{H_i}^i(z)\right\} = \sum_{j=0}^{M_{si}} p_j^i(t) z^{g_j^i} \\ U(z) &= \otimes_\phi \left\{U^1(z), U^2(z), \dots, U^N(z)\right\} = \sum_{j=0}^{M_s} p_j(t) z^{g_j} \\ A(w) &= \sum_{j=1}^{M_s} p_j . 1\left(g_j \ge w\right) \end{aligned} \tag{8}$$

3. Multi-State Series-Parallel System Optimization Model

In this section, the multi-state series-parallel system optimization model with n series subsystem with parallel components is presented. The objective function is to minimize the system cost in order to satisfy the system availability.

$$\begin{aligned} &Min \quad C(X) \\ &s.t: \quad A \ge A_0 \\ &\qquad n_{ih} \ge 0 \quad ;1 \le i \le N, 1 \le h \le H_i \\ &TK_{hk}^i = 0\, or\, 1 \quad ;1 \le i \le N, 1 \le h \le H_i, 1 \le k \le k_i \\ &TK_k^i = 0\, or\, 1 \quad ;1 \le i \le N, K_i + 1 \le k \le k_i + K_{si} \end{aligned} \tag{9}$$

4. Solving Approach

4.1. *SA-Parallel algorithm*

In 1983, Kirkpatrick et al. (1983) developed simulated annealing (SA) algorithm which is a single point Meta-heuristic algorithm. In this algorithm, first, we obtain a random solution and introduce it to a predefined neighborhood and get another solution as the neighbor of the present solution. Then, if the new solution has a better objective function than the old solution, it replaces the old solution. But if the new solution does not have a better objective function than the old solution, it still has the opportunity to replace the old solution under a probability. This probability defined under an acceptance function is as follows:

$$Uniform\,(0,1) < e^{-\Delta / t(it)} \tag{10}$$

In equation number (10), the temperature in each repeat of algorithm is $t(it)$, and Δ is defined as follows:

$$\Delta = f_2 - f_1 \tag{11}$$

In equation number (11), f_2 is the new solution and f_1 is the old solution. We can normalize the Δ as follows:

$$\Delta = \frac{f_2 - f_1}{f_1} \tag{12}$$

Clearly, this acceptance function which depends on the temperature is each repeat $t(it)$ of the SA algorithm.

After this stage, the old solution remains or is replaced by the new solution. In both situations, the result is again sent to the neighborhood function to get a new solution and this procedure repeats till the algorithm satisfies the stop conditions.

Now we present an improved version of the SA present that is called SA-Parallel. In the SA-Parallel the algorithm starts with generating more than one point (random solution) and these solutions are sent to the neighborhood function to generate new solutions. The number of these points is $npop$. Also to produce more improvement, we can generate more than one new solution from each old solution. We show the number of new solutions obtained from an old solution as $nmove$. So, in each iteration of the algorithm, the number of new solutions is $npop \times nmove$. Then we rank all new solutions and select the first $npop$ solutions. These new best solutions are again sent to the neighborhood function to generate $nmove$ new solutions. This procedure repeats until the algorithm satisfies the stop conditions. To compare the new solutions with the old ones, each new solution is compared with the same rank old solution.

The structures of the solutions if the algorithm is Meta-heuristic are as follows:

- Structure related to redundancy: $Position.\mathrm{Re}dundancy.n$,
- Structure related to technical activities: $Position.tkh$,
- Structure related to organizational activities: $Position.tk$.

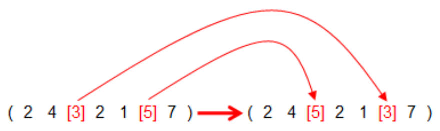
4.1.1. Neighborhood function:

One of the most important bases in the SA algorithm is neighborhood function. Regarding this function, all solutions are categorized into two categories. The first category contains the solutions under SWAP or REVERSION. The second category contains the solutions under MUTATION.

4.1.1.1. First category

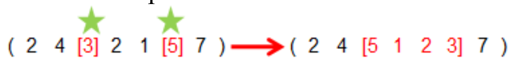
SWAP

In the SWAP condition, for each old solution in the neighborhood function, two elements of solution are selected randomly and then replaced by each other as follows:



REVERSE

In this condition in each old solution, two random elements are selected and then the places of the elements between these two points are reversed as follows:

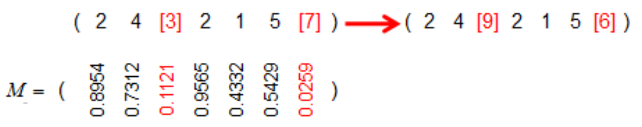


Thus in the operators of the first category, entering new variables is impossible and only the places of the variables changed. But in the second category, it is possible to enter a new variable into a solution.

4.1.1.2. Second category

MUTATION

In this category, a new matrix named M is generated with the same rank of the solution matrix. In the new matrix, each array is filled with a random variable between 0 and 1. Then in the solution matrix, the variables with a correspondence amount less than 0.2 in matrix M are replaced with a random new variable as follows:



In each new set of solutions in the SA algorithm, some of the results are obtained from the first category and the other results from the second category. So we have a wide range of results in each iteration. In Figure 2, the pseudo code of the neighborhood function is presented.

The flowchart of the SA-Parallel algorithm is as follows in Figure 3:

```
$e = Uniforme(0,1)$
    if $e \le 0.5$
        Swap
        Reverse
    Else
        Mutation
$End.$
```

Fig. 2. The pseudo code of neighborhood function

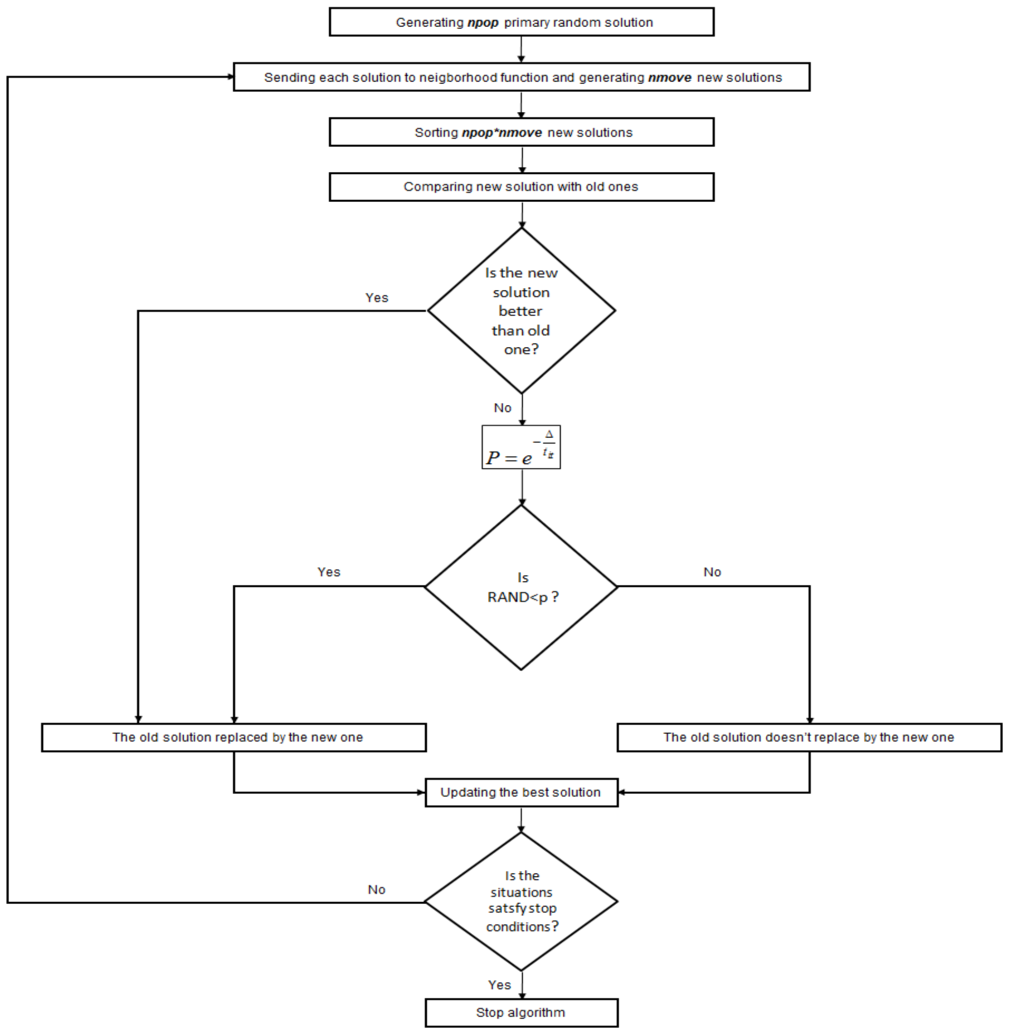

Generating *npop* primary random solution
Sending each solution to neigborhood function and generating *nmove* new solutions
Sorting *npop*nmove* new solutions
Comparing new solution with old ones
Is the new solution better than old one?
Yes
No
$P = e^{-\frac{\Delta}{t_z}}$
Is RAND<p ?
Yes
No
The old solution replaced by the new one
The old solution doesn't replace by the new one
Updating the best solution
Is the situations satsfy stop conditions?
No
Yes
Stop algorithm

Fig. 3. The flowchart of the SA-Parallel algorithm

4.2. VDO-Parallel algorithm

The VDO algorithm developed by Mahdizade and Tavakoli-Moghadam (2009) is a single point Meta-heuristic algorithm which is very similar to SA. In this paper the performance of this algorithm is evaluated in reliability problems. The main difference between SA and VDO is their acceptance function. This function in the VDO algorithm is as follows:

$$P = 1 - e^{-\left(\frac{A^2}{2\sigma^2}\right)} \tag{13}$$

The parameter A in each iteration of the algorithm is updated as:

$$A = A_0\, e^{-\left(\frac{\gamma t}{2}\right)} \tag{14}$$

A_0, γ and σ must be predetermined. The steps of the VDO-Parallel algorism are similar to the SA-Parallel.

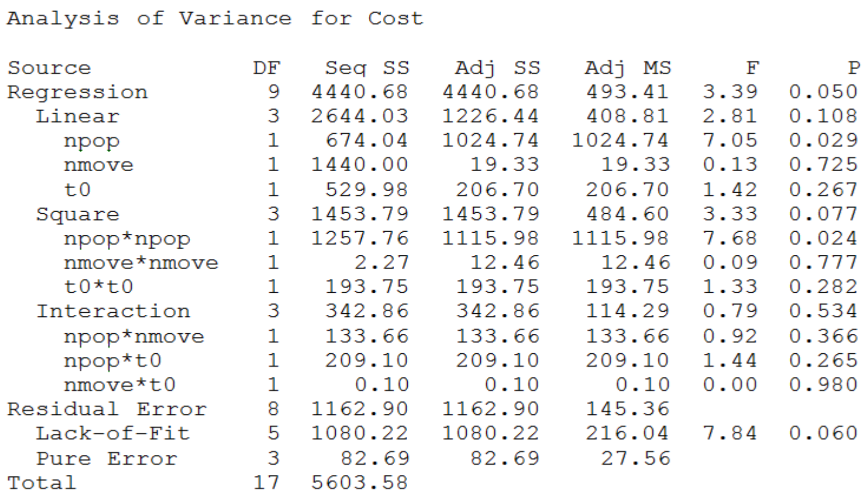
4.3. Parameter tuning

We use the RSM methodology for tuning the SA and VDO parameters. It is similar to problem 1 used by Levitin et al. (2009) in 2009. The results are presented in Figures 4 and 5 along with the information in Tables 1 and 2. Also stop condition in both algorithms is 30 generation without improve

Table 1
SA-Parallel parameters

	Lower value	Upper value
npop	3	5
nmove	10	20
t_0	1000	10000

t_0 is the starting temperature.



```
Analysis of Variance for Cost

Source            DF   Seq SS   Adj SS   Adj MS     F      P
Regression         9  4440.68  4440.68   493.41  3.39  0.050
  Linear           3  2644.03  1226.44   408.81  2.81  0.108
    npop           1   674.04  1024.74  1024.74  7.05  0.029
    nmove          1  1440.00    19.33    19.33  0.13  0.725
    t0             1   529.98   206.70   206.70  1.42  0.267
  Square           3  1453.79  1453.79   484.60  3.33  0.077
    npop*npop      1  1257.76  1115.98  1115.98  7.68  0.024
    nmove*nmove    1     2.27    12.46    12.46  0.09  0.777
    t0*t0          1   193.75   193.75   193.75  1.33  0.282
  Interaction      3   342.86   342.86   114.29  0.79  0.534
    npop*nmove     1   133.66   133.66   133.66  0.92  0.366
    npop*t0        1   209.10   209.10   209.10  1.44  0.265
    nmove*t0       1     0.10     0.10     0.10  0.00  0.980
Residual Error     8  1162.90  1162.90   145.36
  Lack-of-Fit      5  1080.22  1080.22   216.04  7.84  0.060
  Pure Error       3    82.69    82.69    27.56
Total             17  5603.58
```

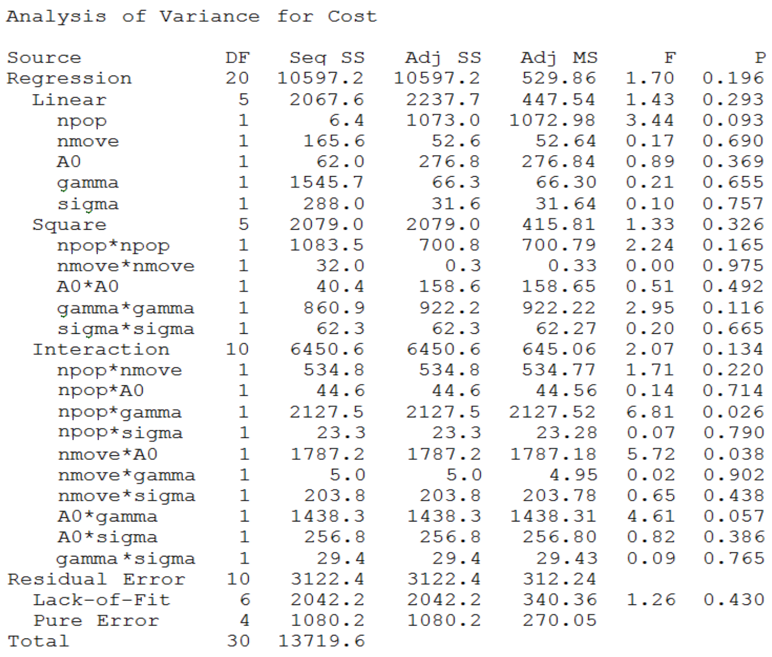
Fig. 4. ANOVA for SA

Table 2
VDO-Parallel parameters

Parameter	Optimal value
Npop	5
Nmove	20
t0	1000

Table 3
VDO-Parallel parameters

	Lower value	Upper value
npop	3	5
nmove	10	20
A_0	1000	10000
γ	0.01	0.001
σ	4	10



```
Analysis of Variance for Cost

Source            DF   Seq SS   Adj SS   Adj MS     F      P
Regression        20  10597.2  10597.2   529.86  1.70  0.196
  Linear           5   2067.6   2237.7   447.54  1.43  0.293
    npop           1      6.4   1073.0  1072.98  3.44  0.093
    nmove          1    165.6     52.6    52.64  0.17  0.690
    A0             1     62.0    276.8   276.84  0.89  0.369
    gamma          1   1545.7     66.3    66.30  0.21  0.655
    sigma          1    288.0     31.6    31.64  0.10  0.757
  Square           5   2079.0   2079.0   415.81  1.33  0.326
    npop*npop      1   1083.5    700.8   700.79  2.24  0.165
    nmove*nmove    1     32.0      0.3     0.33  0.00  0.975
    A0*A0          1     40.4    158.6   158.65  0.51  0.492
    gamma*gamma    1    860.9    922.2   922.22  2.95  0.116
    sigma*sigma    1     62.3     62.3    62.27  0.20  0.665
  Interaction     10   6450.6   6450.6   645.06  2.07  0.134
    npop*nmove     1    534.8    534.8   534.77  1.71  0.220
    npop*A0        1     44.6     44.6    44.56  0.14  0.714
    npop*gamma     1   2127.5   2127.5  2127.52  6.81  0.026
    npop*sigma     1     23.3     23.3    23.28  0.07  0.790
    nmove*A0       1   1787.2   1787.2  1787.18  5.72  0.038
    nmove*gamma    1      5.0      5.0     4.95  0.02  0.902
    nmove*sigma    1    203.8    203.8   203.78  0.65  0.438
    A0*gamma       1   1438.3   1438.3  1438.31  4.61  0.057
    A0*sigma       1    256.8    256.8   256.80  0.82  0.386
    gamma*sigma    1     29.4     29.4    29.43  0.09  0.765
Residual Error    10   3122.4   3122.4   312.24
  Lack-of-Fit      6   2042.2   2042.2   340.36  1.26  0.430
  Pure Error       4   1080.2   1080.2   270.05
Total             30  13719.6
```

Fig. 5. ANOVA for VDO

Table 4
VDO-Parallel parameters

Parameter	Optimal value
Npop	5
Nmove	20
A0	100
Gamma	0.001
Sigma	10

4.4. Numerical examples

To illustrate the method introduced above, 6 numeric examples are provided in this section. Assume a system with two series sub-systems. In sub-system one, three different versions of components are available and in the second sub-system, four different components are available. The components in the first sub-system have three working states of 0, 1 and 2 while the components in the second sub-system have two working states of 0 and1. Transitions are only available in the adjacent states.

The operation rates, transition rates, component costs in each version of subsystem 1 are presented in Table 5. Assume that in the first sub-system, 8 different activities are available numbered from 1 to 8. The actions numbered 1 to 5 are technical activities while the activities numbered 6, 7 and 8 are organizational activities. The effects of technical activities in the three versions of components on the first subsystem are presented in Tables 6, 7 and 8. The pre cost of components in the first sub-system is fixed on 50.

For subsystem 2, the operation rates, transition rates, and component costs of all versions are reported in Table 9. Assuming that in the second sub-system 4 different activities are available, the actions numbered 1 to 3 are technical activities and activity number 4 is an organizational activity. The effects of technical activities in the four versions of components on the second subsystem are presented in Tables 10, 11, 12 and 13. The pre cost of components in the second sub-system is fixed on 60.

In this example, we assume that the operation rate of each sub-system is equal to the sum of the components operation rates and the operation rate of the system is equal to the minimum operation rates of the two sub-systems. We calculate the availability and cost of the system and only a unique demand level w is available (Tian & Zuo, 2009).

Table 5
Components data for sub-system 1

Version	c_h^i	g_{h1}^i	g_{h2}^i	$\lambda_{1,0}$	$\lambda_{2,1}$	$\mu_{0,1}$	
1	18	30	60	0.04	0.05	0.4	0.6
2	25	50	100	0.08	0.09	0.4	0.5
3	40	60	120	0.05	0.06	0.4	0.7

Table 6
Costs and effects of actions on version 1 components in sub-system 1

Action k	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,2,1}$	$a_{k,0,1}$	$a_{k,1,2}$
1	0.1	1.0	1	0.9	1	1
2	0.4	1.5	1	0.8	1	1
3	0.8	3.1	0.9	0.8	1	1
4	0.0	4.0	0.8	0.7	1	1
5	2.0	0.4	1	1	1.5	1
6	6.4	0.0	1	1	1.2	1.5
7	8.0	0.0	1	1	1.5	1.5
8	10.6	0.0	1	1	2.0	3.0

Table 7
Costs and effects of actions on version 2 components in sub-system 1

Action k	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,2,1}$	$a_{k,0,1}$	$a_{k,1,2}$
1	0.1	1.0	1	1	1	1
2	0.5	1.5	1	1	1	1
3	0.9	3.1	1	0.8	1	1
4	0.0	4.0	0.9	0.7	1	1
5	2.0	0.4	1	1	1.6	1
6	6.4	0.0	1	1	1.2	1
7	8.0	0.0	1	1	1.6	1.2
8	10.6	0.0	1	1	2.5	2.0

Table 8
Costs and effects of actions on version 3 components in sub-system 1

Action k	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,2,1}$	$a_{k,0,1}$	$a_{k,1,2}$
1	0.1	1.0	1	1	1	1
2	0.4	1.5	1	1	1	1
3	1	3.1	1	1	1	1
4	0.0	4.2	0.9	1	1	1
5	2.0	0.5	1	1	1.4	1
6	6.4	0.0	1	1	1.2	1.2
7	8.0	0.0	1	1	1.4	1.4
8	10.6	0.0	1	1	2.0	3.2

Table 9
Components data for sub-system 2

Version h	c_h^i	g_{h1}^i	$\lambda_{1,0}$	$\mu_{0,1}$
1	30	80	0.05	0.30
2	35	100	0.06	0.35
3	60	150	0.03	0.45
4	80	180	0.02	0.40

Table 10
Costs and effects of actions on version 1 components in sub-system 2

Action k	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,0,1}$
1	0.4	0.8	0.9	1
2	0.0	3.2	0.6	1
3	1.8	2.4	1	2.2
4	30	0.0	0.9	2.4

Table 11
Costs and effects of actions on version 2 components in sub-system 2

Action k	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,0,1}$
1	0.4	0.8	0.9	1
2	0.0	3.2	0.6	1
3	1.8	2.8	1	2.1
4	30	0.0	1	2.2

Table 12
Costs and effects of actions on version 3 components in sub-system 2

Action k	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,0,1}$
1	0.4	1	1	1
2	0.0	3.2	0.6	1
3	1.8	2.6	1	1.6
4	30	0.0	1	1.8

Table 13
Costs and effects of actions on version 4 components in sub-system 2

Action K	$c_{kho}^{Ti}\left(c_{ho}^{Ti}\right)$	c_{kh}^{Ti}	$a_{k,1,0}$	$a_{k,0,1}$
1	0.4	1	1	1
2	0.0	3.2	1	1
3	1.8	2.6	1	2.2
4	30	0.0	1	1.6

Example 1

In this example, $w = 500$ and $A_0 = 0.9$. Table 14 present the results of genetic, SA-Parallel, and VDO-Parallel algorisms and the fitness convergence graphs of the SA-Parallel and VDO-Parallel are depicted in Figures 6.

Example 2

In this example, $w = 500$ and $A_0 = 0.95$. The results of genetic, SA-Parallel, and VDO-Parallel algorisms arepresented in Table 15. Figures 7show the fitness convergence graphs of the SA-Parallel and VDO-Parallel.

Table 14
The results of the algorithms in example 1

Subsystem	Version	Redundancy	Actions	Availability	Cost	Algorithm
1	1	0				
1	2	6	8			
1	3	0				
2	1	4	1,3	0.9231	531	GA
2	2	3	2,3			
2	3	0				
2	4	0				
1	1	4	5			
1	2	2	1,2,4			
1	3	1	1,2,3	0.9134	486.7	SA-Parallel
2	1	0				
2	2	2				
2	3	2				
2	4	0				
1	1	4	5			
1	2	2	1,2,4			
1	3	1	1,2,3	0.9134	486.7	VDO-Parallel
2	1	0				
2	2	2				
2	3	2				
2	4	0				

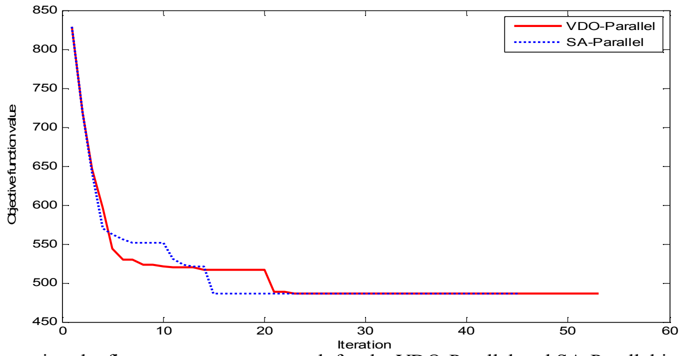


Fig. 6. Comparing the fitness convergence graph for the VDO-Parallel and SA-Parallel in example 1

Table 15
The results of the algorithms in example 2

Subsystem	Version	Redundancy	Actions	Availability	Cost	Algorithm
1	1	0				
1	2	6	8			
1	3	0				
2	1	3	3	0.9510	559.2	GA
2	2	0				
2	3	3	3			
2	4	0				
1	1	0	8			
1	2	6	8			
1	3	0				
2	1	5	4	0.9514	538.8	SA-Parallel
2	2	2	4			
2	3	0				
2	4	0				
1	1	1				
1	2	4				
1	3	3				
2	1	1		0.9500	475.7	VDO-Parallel
2	2	0				
2	3	4				
2	4	0				

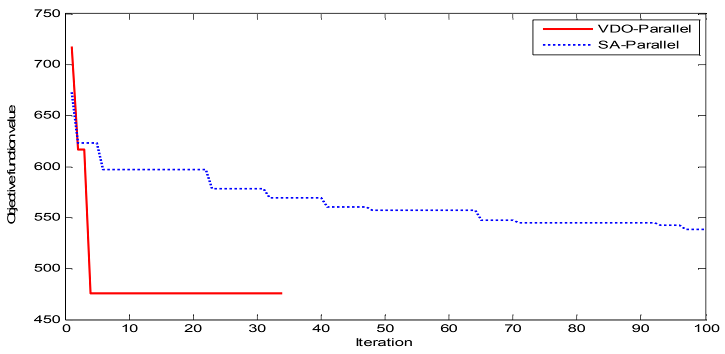


Fig. 7. Comparing the fitness convergence graph for the VDO-Parallel and SA-Parallel in example 2

Example 3

In this example, $w = 500$ and $A_0 = 0.99$. The results of genetic, SA-Parallel, and VDO-Parallel algorisms are reported in Table 16 and the fitness convergence graphs of the SA-Parallel and VDO-Parallel are shown in Figures 8.

Example 4

In this example, $w = 1000$ and $A_0 = 0.9$. The results of genetic, SA-Parallel, and VDO-Parallel algorisms are shown in Table 17 and the fitness convergence graphs of the SA-Parallel and VDO-Parallel are shown in Figures 9.

Table 16
The results of the algorithms in example 3

Subsystem	version	Redundancy	Actions	availability	Cost	Algorithm
1	1	0				
1	2	8				
1	3	0				
2	1	8	2,3,4	0.9906	627.1	GA
2	2	0				
2	3	0				
2	4	0				
1	1	1				
1	2	4				
1	3	1				
2	1	2		0.99714	493	SA-Parallel
2	2	3				
2	3	1				
2	4	0				
1	1	4				
1	2	2				
1	3	1				
2	1	0		0.99512	497	VDO-Parallel
2	2	3				
2	3	2				
2	4	0				

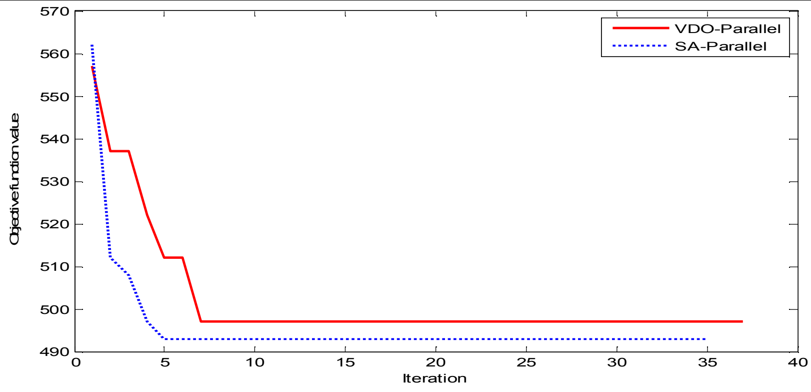


Fig. 8. Comparing the fitness convergence graph for the VDO-Parallel and SA-Parallel in example 3

Table 17
The results of the algorithms in example 4

Subsystem	version	Redundancy	Actions	availability	Cost	Algorithm
1	1	6	1	09038	940	GA
1	2	6	3			
1	3	2				
2	1	2	4			
2	2	9	4			
2	3	1	1,4			
2	4	0				
1	1	7	8	0.9020	905.8	SA-Parallel
1	2	7	8			
1	3	0				
2	1	0				
2	2	8	3			
2	3	3				
2	4	0				
1	1	3	1,6,8	0.90	294.9379	VDO-Parallel
1	2	8	1,6,8			
1	3	1	6,8			
2	1	1				
2	2	8				
2	3	1				
2	4	2				

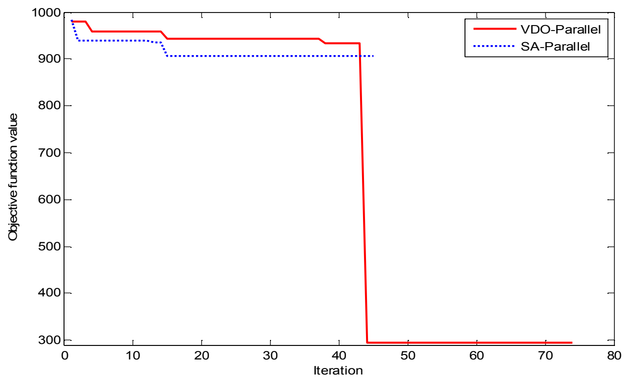


Fig. 9. Comparing the fitness convergence graph for the VDO-Parallel and SA-Parallel in example 4

Example 5

In this example, $w = 1000$ and $A_0 = 0.95$. The results of genetic, SA-Parallel, and VDO-Parallel algorisms are presented in Table 18 and the fitness convergence graphs of the SA-Parallel and VDO-Parallel are shown in Figures 10.

Example 6

In this example, $w = 1000$ and $A_0 = 0.99$. Table 19 show the results of genetic, SA-Parallel, and VDO-Parallel algorisms, and the fitness convergence graphs of SA and VDO algorisms are illustrated in Figures 11.

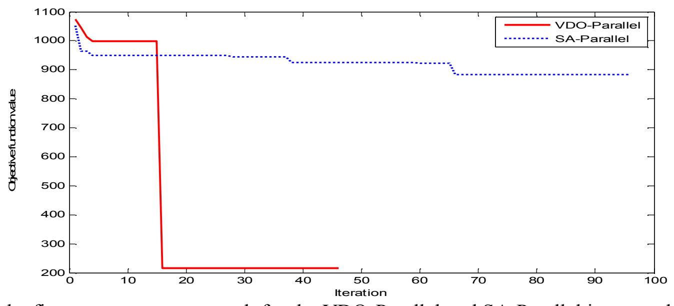


Fig. 10. Comparing the fitness convergence graph for the VDO-Parallel and SA-Parallel in example 5

Table 18
The results of the algorithms in example 5

Subsystem	version	Redundancy	Actions	availability	Cost	Algorithm
1	1	5	8			
1	2	8	8			
1	3	1	8			
2	1	4	4	09539	957.6	GA
2	2	2	4			
2	3	2	3,4			
2	4	2	4			
1	1	0	1,6			
1	2	10	5,6			
1	3	2	6			
2	1	1	3	0.95	883.9938	SA-Parallel
2	2	2	1			
2	3	9				
2	4	0	1			
1	1	6	2			
1	2	9				
1	3	1	4			
2	1	6	1	0.9599	215.1572	VDO-Parallel
2	2	8				
2	3	1				
2	4	0				

Table 19
The results of the algorithms in example 6

Subsystem	version	Redundancy	Actions	availability	Cost	Algorithm
1	1	6	1,8			
1	2	9	8			
1	3	0				
2	1	8	1,4	09925	987.3	GA
2	2	5	3,4			
2	3	1	4			
2	4	0				
1	1	2	1,2,3,6,8			
1	2	7	1,6,8			
1	3	3	6,8			
2	1	0		0.99	299.9471	SA-Parallel
2	2	8	4			
2	3	3	4			
2	4	1	2,3,4			
1	1	0	2,5,6			
1	2	0	1,2,3,5,6			
1	3	10	1,3,4,5,6			
2	1	10	1,3,4	0.99	227.8364	VDO-Parallel
2	2	10	2,4			
2	3	10	2,4			
2	4	9	2,3,4			

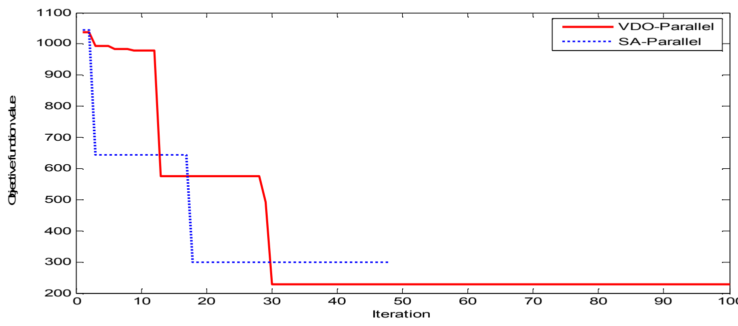


Fig. 11. Comparing the fitness convergence graph for the VDO-Parallel and SA-Parallel in example 6

As mentioned above, in example 1, the results of SA-Parallel and VDO-Parallel algorithms are the same and both of them are better than the genetic algorithm. The only difference between the two parallel algorithms is the number of iterations. In the SA-Parallel, after 45 iterations the algorithm reaches to convergence but the number of iterations in the VDO-Parallel is 53. In the examples number 2, 4, 5 and 8, the VDO-Parallel yields better results than the SA-Parallel algorithm. Only in example number 3, the results of the SA-Parallel are better than those of the VDO-Parallel algorithm. Of course in all examples, both of the proposed parallel algorithms have better results than the genetic algorithm.

5. Conclusion and Further studies

All in all, in this paper we showed that parallel meta-heuristic algorithms like SA-parallel and VDO-Parallel have a better performance in the series-parallel multi-state systems than the ordinary ones such as genetic. Differences between the acceptance functions of SA and VDO algorithms enable them to avoid local solutions. Using this approach for the SA-parallel and VDO-Parallel improves the ability of these two algorithms. Of course, operation of these two algorithms depends on neighborhood function structures. Using the approach, one can improve the ability of other meta-heuristic algorithms.
In this paper we determined these decision variables: 1) Number of assigned components in each sub-system of each system, 2) The version of assigned components in each sub-system of each system, and 3) The type of technical and organizational activities.

References

[1] Barlow RE, Wu AS., (1978), Coherent systems with multi-state components. Mathematics of Operations Research; 3(4):275–81

[2] Boedigheimer RA, Kapur KC., (1994), Customer-driven reliability models for multi- state coherent systems. IEEE Transactions on Reliability; 43(1):46–50

[3] Chern, M.S. (1972) On the computational complexity of reliability redundancy allocation in a series system. Operations Research Letters,11, 309–315

[4] Coit, D.W. and Smith, A.E. (1996) Reliability optimization of series parallel systems using a genetic algorithm. IEEE Transactions on Reliability, 45(2), 254–260

[5] Ding, Y. Lisnianski A., (2008), Fuzzy universal generating functions for multi-state system reliability assessment Fuzzy Sets and Systems 159, 307 – 324 .

[6] Kim,J. Gen, M., Ga-based Reliability Design :state-of-the-Art survey .computers & industrial engineering , 37,151-155.

[7] Kirrkpatrick, S. Gelatt,C,D. Vecchi,Jr,M ,P. (1983), Optimization by Simulated Annealing. Science, 13 May 1983, Volume 220, Number 4598.

[8] Konak, S. SMITH A and COIT D, (2003), Efficiently solving the redundancy allocation problem using tabu search. IIE Transactions, 35, 515–526.

[9] Levitin G, Lisnianski A, Ben Haim H, Elmakis D. (1998), Redundancy optimization for series–parallel multistate systems. IEEE Transactions on Reliability; 47(2):165–72

[10] Liang Y, Chen Y. (2007), Redundancy allocation of series-parallel systems using a variable Neighborhood search algorithm . Reliability Engineering and System Safety 92, 323–331

[11] Liang, Y., (2004), Associate Member, IEEE and Smith, A Senior Member, IEEE An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP) IEEE TRANSACTIONS ON RELIABILITY, VOL. 53, NO. 3, SEPTEMBER, 417.

[12] Lisnianski A, Levitin G, Ben-Haim H, Elmakis D. (1996), Power system structure optimization subject to reliability constraints. Electr Power Sys Res; 39(2):145–52.

[13] Lisnianski A, Levitin G. (2003), Multi-state system reliability: assessment, optimiza- tion and applications. Singapore: World Scientific.

[14] Mehdizadeh, E., Tavakkoli-Moghaddam., R., (1999), VIBRATION DAMPING OPTIMIZATION ALGORITHM FOR AN IDENTICAL PARALLELMACHINES SCHEDULING PROBLEM. The 2nd International Conference of Iranian Operations Research SocietyMay 20-22, 2009 – Babolsar, Iran.

[15] Ramirez-Marquez JE, Coit DW., (2004), A heuristic for solving the redundancy allocation problem for multi-state series–parallel systems. Reliability Engineering and System Safety; 83(3):341–9.

[16] Tian Z , Levitin G ,Zuo M A., (2009), joint reliability–redundancy optimization approach for multi-state series–parallel systems. Reliability Engineering and System Safety 94 1568–1576.

[17] Tian Z, Zuo MJ, Huang H. (2005), Reliability–redundancy allocation for multi-state series–parallel systems. In: Proceedings of the 2005 European safety and reliability conference, Tri City, Poland,. p. 1925–30

[18] Tian Z, Zuo MJ. (2006), Redundancy allocation for multi-state systems using physical programming and genetic algorithms. Reliability Engineering and System Safety; 91(9):1049–56.

[19] Ushakov I. (1987), Optimal standby problem and a universal generating function. Sov J Comput Sys Sci 1987;25(4):61–73.

[20] ZuoMJ, TianZ, Huang H. (2007), An efficient method for eliability evaluation of multi-state networks given all minimal path vectors. IIE Transactions; 39(8):811–7

[21] Zuo, MJ,, Tian, Z. (2006), Performance evaluation for generalized multi-state k-out-of-n systems. IEEE Transactions on Reliability; 55(2):319–27.