International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Non-Linear Threshold Algorithm for the Redundancy Optimization of Multi-State Systems

Nabil Nahas
Département d'Administration,
Université de Moncton, Moncton (NB), Canada.
E-mail: nabil.nahas@umoncton.ca

Mustapha Nourelfath
Mechanical Engineering Department,
Laval University, Quebec, Canada.
Corresponding author: mustapha.nourelfath@gmc.ulaval.ca

(Received June 2, 2020; Accepted October 2, 2020)

Abstract
To improve system performance, redundancy is widely used in different kinds of industrial applications such as power systems, aerospace, electronic, telecommunications and manufacturing systems. Designing high performant systems which meet customer requirements with a minimum cost is a challenging task in these industries. This paper develops an efficient approach for the redundancy optimization problem of series-parallel structures modeled as multi-state systems. To reach the target system availability, redundancies are used for components among a list of products available in the market. Each component is characterized by its own availability, cost and performance. The goal is to minimize the total cost under a system availability constraint. Discrete levels of performance are considered for the system and its components. The extreme values of such performance levels correspond to perfect functioning and complete failure. A piecewise cumulative load curve represents consumer demand. System availability corresponds to the aptitude to fulfill this demand. The multi-state system availability evaluation uses the universal moment generating function technique. The proposed optimization algorithm is based on the non-linear threshold accepting metaheuristic, while using a self-adjusting penalty guided strategy. The obtained results demonstrate the approach efficiency for solving the redundancy optimization problem of multi-state systems. Its effectiveness is also tested using the classical redundancy optimization problem of binary-state systems. The algorithm is evaluated by comparison to the best known methods. For multi-state systems, it is compared to genetic algorithm and tabu search. For binary-state systems, it is compared to genetic algorithm, tabu search, ant colony optimization and harmony search. The obtained results demonstrate that the proposed approach outperforms these state-of-the-art benchmark methods in finding, for all considered instances, a high-quality solution in a minimum computational time.

Keywords- Reliability, Redundancy optimization, Series-parallel systems, Multi-state Metaheuristics.

1. Introduction

To improve system reliability, redundancy is widely used in different kinds of industrial applications including aerospace, power systems, electronic, telecommunications and manufacturing systems. The redundancy optimization problem (ROP) of a series-parallel system considers many sub-systems in series and for each sub-system, different component versions are connected in parallel. Thus, two design objectives are generally considered. The first objective is to find the number of redundant components and their location to ensure the required performance subject to resource constraints. The second objective is to minimize the cost under performance constraints. The current reliability optimization literature distinguishes between the ROP of multi-state systems (MSS), and the ROP of binary-state systems (BSS).

The ROP of MSS is more recent. Multi-state reliability modeling considers that a system and its

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

components may have more than two levels of performance varying from perfect functioning to complete failure. A review of MSS literature can be found, for example, in (Levitn, 2005; Levitin and Lisnianski, 2001; Lisnianski and Levitin, 2003; Ushakov and Levitin; 2002). The ROP for series-parallel MSS was initially presented in (Ushakov, 1987), where the universal generating function method was used for reliability evaluation (Ushakov, 1986). Many papers dealt with the development of heuristic methods based on metaheuristics for the ROP of series-parallel MSS (Kuo et al., 2001). These methods include genetic algorithms (Levitin et al., 1997; Levitin et al., 1998; Lisnianski et al., 1996), ant colony (Nourelfath et al., 2003), harmony search in (Nahas and Thien-My, 2010), a heuristic algorithm in (Ramirez and Coit, 2004), a coupling of tabu search and genetic algorithms in (Ouzineb et al., 2010), and artificial bee colony algorithm (Yeh and Hsieh, 2011). The present paper contributes to the literature on solution methods for the ROP of MSS. An efficient approach is developed, and compared to the state-of-the-art benchmark methods in terms of solution quality, computation time and implementation effort.

In the ROP of BSS (Chern, 1992), the system and its components are assumed to be either in a good state or in a failed state (Misra and Ljubojevic, 1973). Numerous optimization approaches and formulations have been proposed to solve the ROP under the binary-state assumption (Kuo and Prasad, 2000). These techniques can be classified as exact optimization approaches and non-exact or heuristic approaches. Exact techniques include dynamic programming (Bellman and Dreyfus, 1958; Fyffe et al., 1968; Nakagawa and Miyazaki, 1981; Yalaoui et al., 2005), mixed-integer and nonlinear programming (Tillman et al., 1977), and integer programming (Bulfin and Liu, 1985; Gen et al., 1993; Ghare and Taylor, 1969; Misra and Sharma, 1991). More recent exact methods are (Billionnet, 2008; Caserta and VoB, 2015a; Caserta and VoB, 2016; Onishi et al., 2007). Non-exact methods based on metaheuristics include genetic algorithms (Coit and Smith, 1996b; Painton and Campbell, 1995; Yokota et al., 1995; Yokota et al., 1996), tabu search algorithm (Kulturel-Konak et al., 2003), artificial neural networks (Nourelfath and Nahas, 2005), simulated annealing (Kim et al., 2006), ant colony optimization (Liang and Smith, 2004; Nahas et al., 2007; Nahas et al., 2008; Nourelfath et al., 2005; Zhao et al., 2007), and harmony search (Nahas and Thien-My, 2010). More recent heuristics are (Chang and Kuo, 2018; Liu, 2016). In (Agarwal and Gupta, 2006; Agarwal et al., 2010), the authors focused on solving highly constrained redundancy optimization problems in binary complex systems. As the ROP of binary-state series-parallel systems has been recently solved to optimality (see for example (Caserta and VoB, 2015b), it is solved in this paper just to illustrate the efficiency of our approach in comparison to other existing meta-heuristics approaches. The objective of proposed approach is rather to solve the more recent ROP of multi-state systems, which has never been solved to optimality.

All the above-mentioned approaches were (independently) proposed to solve either the ROP of binary-state systems, or the ROP of multi-state systems. When the proposed approaches are based on the same metaheuristic (for BSS and MSS), the operators and the algorithm steps are either different, or inefficient if used for both cases due to different modeling assumptions. Consequently, a method that is initially designed to solve the ROP of BSS cannot be used for MSS, and vice-versa. To the best of our knowledge, there is no existing method designed to solve both the ROP of series-parallel BSS and MSS. The present contribution develops a unified and efficient heuristic based on the non-linear threshold accepting (NLTA) metaheuristic (Nahas and Nourelfath, 2014), which consists in a simple deterministic iterative local search method. Recently, this algorithm was applied to efficiently solve the redundancy allocation problem considering multiple redundancy strategies (Nahas et al., 2018). The NLTA algorithm represents a variant of the conventional (linear) threshold accepting algorithm. Our approach also uses a self-adjusting penalty guided

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

strategy. The resulting method is efficient and very simple to implement. Numerical results for various test problems from previous reported research are presented, both for multi-state and binary cases. The performance of our approach is verified by comparing its results with those of the state-of-the-art benchmark methods, using all the existing benchmark problems.

The manuscript is structured as follows. Section 2 develops the method used for the availability evaluation of multi-state systems. Section 3 presents a description, and a mathematical formulation for the studied redundancy optimization problem. Section 4 presents the non-linear threshold accepting algorithm and the self-adjusting penalty guided strategy. Section 5 details the application of this approach to solve the formulated problem. Section 6 presents the numerical results. Section 7 concludes the paper.

2. Availability Evaluation for Multi-State Systems

To optimize the multi-state system redundancy, it is mandatory to develop an efficient evaluation procedure to estimate the availability of each series-parallel configuration. We consider a system with K states corresponding to different levels performance. For repairable MSS, availability is assessed by the probability to satisfy the demand (Levitn, 2005; Levitin et al., 1998; Ushakov, 1987; Xue and Yang, 1995):

$$A(t,D) = Pr\{G(t) \geq D\}, \tag{1}$$

where, $G(t)$ is the system performance at time t, and D is the demand.

The present paper uses universal moment generating function (UMGF) to calculate the MSS availability.

Consider an example of a power generation system. Reliability is considered as a measure of the aptitude of the system to meet the customer demand (D) by supplying an adequate amount of electrical energy (S). For repairable MSS, a steady-state availability A can be defined as $Prob(S \leq D)$. The following equations present the distribution of states probabilities, and the steady-state availability:

$$P_j = \lim_{t\to\infty}\left[Prob(S(t) = S_j)\right] \tag{2}$$

$$A = \sum_{S\geq D} P_j \tag{3}$$

It is convenient to distinguish, for a period T of operation, different intervals with specific durations $(T_1, T_2, \ldots, T_M)$. To each interval, corresponds a demand level $(D_1, D_2, \ldots, D_M)$, which needs to be fulfilled. In this context, the MSS availability is:

$$A = \frac{1}{\sum_{j=1}^{M} T_j}\sum_{j=1}^{M} Prob(S \geq D_j)T_j \tag{4}$$

2.1 Definition and Properties

Let S be a discrete random variable having J accessible states. Each state is characterized by a performance level S_j and its corresponding probability P_j. The UMGF of S is a polynomial function:

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

$$u(z) = \sum_{j=1}^{J} P_j z^{S_j} \quad (5)$$

The function $u(z)$ can be used to find the probabilistic characteristics of S. In particular, the availability A is:

$$Prob(S \geq D) = \Psi(u(z)z^{-D}) \quad (6)$$

such as the operative Ψ is distributive and defined by:

$$\psi(Pz^{\sigma-D}) = \begin{cases} P, & if \quad \sigma \geq D \\ 0, & if \quad \sigma < D \end{cases} \quad (7)$$

$$\psi\left(\sum_{j=1}^{J} P_j z^{S_j-D}\right) = \sum_{j=1}^{J} \Psi(P_j z^{S_j-D}) \quad (8)$$

As an illustrative example, let consider single components such as each component has a nominal functioning state with performance S_i, and a total failure state with performance 0. The availability of each component is $A_i = Prob(S = S_i)$, and $Prob(S = 0) = 1 - A_i$. The UMGF of such binary state component is:

$$u_i(z) = (1 - A_i)z^0 + A_i z^{S_i} = (1 - A_i) + A_i z^{S_i} \quad (9)$$

In the next subsection, we define some basic composition operators to evaluate the MSS availability of a series-parallel system. This is done by using simple algebraic operations on the individual UMGF of components depending on their interactions (Ushakov, 1986).

2.2 Composition Operators

Let consider a subsystem m containing J_m components connected in parallel. In this case, the performance of the system is the sum of the performances of its components. The following operator defines the UMGF of parallel components:

$$u_p(z) = \Gamma(u_1(z), u_2(z), \ldots, u_n(z)), \qquad where \quad \Gamma(g_1, g_2, \ldots, g_n) = \sum_{i=1}^{n} g_i.$$

The Γ operator is a product of the individual UMGF of components. We have:

$$u_p(z) = \prod_{j=1}^{J_m} u_j(z).$$

For binary state components whose individual UMGF are defined in equation (9), we have:

$$u_p(z) = \prod_{j=1}^{J_m} \left(1 - A_j + A_j z^{S_i}\right).$$

In series system, the component having the least performance constitutes the bottleneck and defines the system performance. For example, the UMGF of two series components is evaluated using the

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

following operator:

$$\eta\big(u_1(z),u_2(z)\big)=\eta\left(\sum_{i=1}^{n}P_iz^{a_i},\sum_{j=1}^{m}Q_jz^{b_j}\right)=\sum_{i=1}^{n}\sum_{j=1}^{m}P_iQ_jz^{\min\{a_i,b_j\}}.$$

It is possible to express the UMGF of the series-parallel configuration by recursively applying the operators Γ and η on individual UMGF components (Levitn, 2005). The UMGF approach is convenient for numerical implementation and in high dimension combinatorial optimization contexts. It is advantageously used in this paper to solve the ROP of MSS.

3. The ROP of Multi-State Systems

Notation

t	: time
D	: required performance level or demand
$G(t)$	: output performance of the system at time t
H_i	: number of versions available for a component in subsystem i
h_i	: version index of a component belonging to subsystem i
h	: vector $(h_1,\ldots,h_n)$
$G(h_i)$	: minimal capacity of a component of version h_i
$A(h_i)$	: availability of a component of version h_i
$C(h_i)$	: cost of a component of version h_i
R_i	: number of versions available for a component in subsystem i
r_i	: number of parallel components in subsystem i
A_0	: minimum required level of MSS availability
r	: vector $(r_1,\ldots,r_n)$
$C(h,r)$	: total cost of series-parallel MSS
$A(h,r,D,t)$	: availability of series-parallel MSS

Each subsystem i is composed of many identical components in parallel. Several components versions are available in the market. Each version h_i has different characteristics: nominal capacity $G(h_i)$, availability $A(h_i)$ and cost $C(h_i)$. The subsystem i is defined by the version number h_i $(1\le h_i\le H_i)$ and the number of parallel components r_i $(1\le r_i\le R_i)$. The total cost is $C=\sum_{i=1}^{n}r_i\times C(h_i)$. We consider homogeneous structures, which means that only one component type is selected to provide redundancy. The decision variables are defined by the vector (**h**,**r**) specifying the MSS structure, with $\mathbf{h}=(h_1,\cdots,h_n)$ and $\mathbf{r}=(r_1,\cdots,r_n)$.

The goal of MSS availability optimization is to minimize the system cost with respect to a required availability level A_0 (Lisnianski et al., 1996):

$$\text{Minimize}\quad C(h,r) \tag{10}$$

$$\text{Subject to}\quad A(h,r,D,t)\ge A_0 \tag{11}$$

In the classical binary-state case, each component version has a different reliability, cost and weight. The goal of the considered optimization problem is to maximize the system reliability with respect to weight and cost limitations.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

4. The Non-Linear Threshold Accepting Algorithm

Notation

w	: parameter which decreases during the NLTA process
w_0	: positive parameter
$G(w)$	: acceptance function
Δw	: decreasing rate of w
T_0	: initial threshold
N	: maximum number of iterations
m	: counter of iterations
T_m	: threshold parameter at iteration m
S_0	: initial solution
S	: current solution
S'	: neighbor solution
S_m	: solution at iteration m
$f(.)$	: objective function of a given solution
G'	: ratio $f(s)/f(s')$
$NB(\mathbf{X})$	: neighborhood of solution **X**
TC	: system total cost
TW	: system total weight
$C(h,r)_{pen}$	: penalized objective function (MSS cost)
ρ	: random number from $[0,1]$
R_{max}	: maximum reliability
R_{av}	: average reliability
$R_{syst-pen}$	: penalized objective function (BSS reliability)

4.1 The Conventional Threshold Acceptance Algorithm

The threshold accepting (TA) algorithm is a simple local search algorithm developed in (Dueck and Scheuer, 1990). Like any local search algorithm, TA algorithm starts by a feasible solution (S_0), explores the neighborhood of the current solution and moves to a neighbor solution (S) if it improves the objective function or if the deterioration (i.e. $f(S) - f(S_0)$) is less than a threshold parameter T_m. The parameter T_m is decreasing by a constant value Δt at each iteration. The pseudo-code of the method, for a minimization model, is given below:

Step 1. Construct an initial solution S_0
Step 2. Initialize T_0 and N
Step 3. Set $m = 0$
Step 4. While $m < N - 1$
- Create randomly a neighbor solution S' and set $\Delta = f(S') - f(S_m)$
- If $\Delta < T_m$ then set $S_{m+1} = S'$ else $S_{m+1} = S_m$

Step 5. Set $T_{m+1} = T_m - \Delta t$ and $m = m + 1$
Step 6. End While

4.2 Extension

A new acceptance rule is used to extend the conventional threshold acceptance algorithm. To introduce this rule, we first define the following acceptance function:

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

$$G(w) = \frac{1}{\sqrt{1+(w/w_0)^2}} \quad (12)$$

where, w_0 is a positive parameter, and w is a parameter which decreases during the execution of the algorithm.

The resulting extension is similar to simulated annealing and threshold accepting algorithm because during the search process it may accept low-quality solutions according to a different acceptance criterion. Initially, parameter w is set to adequately high value. Since, the acceptance function $G(w)$ (equation (12)) is inversely proportional to w, any decrease in w will increase the value of acceptance function, thus making it hard for a bad solution to be accepted. For any new solution S_0 generated, it may be accepted for one of two reasons. The new solution is improvement over the current one, i.e. $f(S) < f(S_0)$, or it is accepted based on the comparison between $G(w)$ and G_0, where G_0 is the ratio between the objective values of the current solution and the new one ($G_0 = f(S)/f(S0)$) . Whenever an iteration changes, or a new solution is accepted, it reduces the value of w by a constant value Δw. The parameters w_0 and Δw are used to regulate the decay rate and thereby the search process. Manipulating the values of these parameters controls the convergence of $G(w)$. Having an acceptance function that decreases in a nonlinear way may allow for a better explorative ability of the algorithm. One can remark that Equation (12) resembles the transfer function of a low-pass filter (RC-filter) used in electronics to reduce the amplitude of signals with higher frequencies. The following steps present the proposed algorithm for a minimization problem:

Step 1. Construct a solution S_0
Step 2. Initialize w_0, w, Δw and N (maximum number of iterations)
Step 3. While (number of iterations $< N$)
- Choose a neighbor solution S'
- $G' = \frac{f(S)}{f(S')}$
- $G(w) = \frac{1}{\sqrt{1+(w/w_0)^2}}$
- If $G' \geq 1$ or $G(w) < G'$ then set $S = S'$ and $w = w - \Delta w$

Step 4. End While

Note that the objective function of a given solution is penalized in our approach. In what follows, we demonstrate that this simple method, coupled with a penalty guided strategy, efficiently solves hard instances of the ROP of MSS and BSS. Before presenting the computational results, let us explain first the rationale behind using this non-linear function.

4.3 Why the Proposed NLTA Approach is Successful?

While inheriting the characteristics of TA and simulated annealing, our algorithm use a non-linear acceptance function. To answer the above question, we need then to better understand the role of the non-linear decay rate on the behavior of the algorithm.

High Jump Competition Metaphor

As a metaphor, let consider the process encountered in high jump competition. The high jump is a track and field athletics event in which competitors must jump over a horizontal bar placed at measured heights. In competition the winner is the person who cleared the highest height. An

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

important remark that has inspired our work is that the bar is raised in a non-linear way. For example, it can be raised by 5 cm at the beginning, then by 3 cm after, etc. Let consider a set of N competitors that are chosen from a large number of persons. A competitor is chosen if he performs the best jump or if his jump is within an acceptable range. This range is larger at the beginning but lowered in a non-linear way as indicated above. In our metaphor, the persons participating to the high jump competition represent the search space. Choosing between the N competitors corresponds to a number N of iterations. In this metaphor, it is more probable to select a good competitor if the bar is raised in a non-linear way. Also, for more adaptability to competitors performance, it should be better to evaluate the decay depending on previous performance of the competitors.

Non-linearity and Adaptability
The metaphor above highlights that non-linearity and adaptability are desirable in the search process of good solutions (competitors). The non-linear decay rate adapts to the quality of the best solution more easily than a linear one. In fact, the NLTA adjusts the decay rate at every iteration. On the one hand, substantial improvements are observed, the decrease of the decay rate is also high. On the other hand, slower improvements in the best solution imply slower reactions of the decay rate. Unlike the classical (linear) TA, the NLTA should never become greedy, since the decay rate is adapted to the quality of the best obtained solution. These characteristics and the penalization strategy have allowed for an efficient solution of the ROP of BSS and MSS as described in the next section.

5. Application to the ROP of Multi-State Systems

To apply the proposed NLTA approach, we need to define the neighborhood space and the way the objective function is penalized.

5.1 Neighboring Solutions

Since we deal with homogeneous series-parallel MSS, each subsystem i contains a number of identical components. The following procedure is used to define the neighborhood solutions:

- Step 1. Choose randomly a subsystem i
- Step 2. Generate randomly a number ρ from $[0,1]$
- Step 3. If $\rho \geq 0.5$ then
  Change the component version of subsystem i
  (a) Choose randomly a version number h_i $(1 \leq h_i \leq H_i)$ from available choices
  (b) Set the version h_i to the component of subsystem i
- Step 4. If $\rho < 0.5$ then
  Change the number of components connected in parallel
  (a) Choose randomly a number r_i $(1 \leq r_i \leq R_i)$
  (b) Set the number r_i to the component of subsystem i
- Step 5. End

5.2 Penalizing the Objective Function

For this problem, infeasible solutions are also accepted to force the penalized search towards the feasible solutions area. That is, the proposed penalized objective function is:

$$C(h,r)_{pen} = C(h,r) + \frac{A_0}{A(h,r,D,t)} \times \alpha \quad (13)$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

The value of α is updated as follows:

- If all a solutions are infeasible, set α to $\alpha \times c_1$.
- If all previous b solutions are feasible, set α to $\alpha \times c_2$

where, c_1 and c_2 are positive numbers.

For the ROP of BSS, the move applied to the current solution consists in either adding or subtracting one component. A subsystem is randomly selected. Then, the version of the added of subtracted component is randomly selected from the available versions. Also, infeasible solutions are accepted to force the penalized search towards the feasible solutions area.

The penalty functions are introduced to encourage the algorithms to explore both feasible and infeasible regions. Searching near the boundaries have proven beneficial. Gendreau et al. (1994) introduced the concept of adding infeasible solutions in the initial population. However, to penalize these solutions, they used self-adjusting penalties. Weights increase if, during the last few iterations, only infeasible solutions are found. They decrease if all recent solutions are feasible. In (Coit and Smith, 1996a; Coit and Smith, 1996b), the authors proposed GA to solve the RAP and found that penalizing the infeasible solutions benefits in improving final feasible solutions. Similar techniques are applied in (Kulturel-Konak et al., 2003; Nahas et al., 2014; Ouzineb et al., 2008). Using the same concept, the penalty weights have been adjusted in our proposed approach.

6. Computational Results

The algorithms were implemented in MATLAB on a computer with a processing speed of 2 Ghz.

6.1 ROP of Multi-State Systems

6.1.1 Test Problems

Four benchmarks are used. The first three examples were introduced in (Levitin et al., 1998), (Lisnianski et al., 1996) and (Levitin et al., 1997). The fourth example was introduced in (Ouzineb et al., 2008).

Table 9 and 10 present the data of Example 1. Three different values of the availability index A_0 have been used to generate three variations.

Table 11 and 12 present the data of Example 2. The reliability index A_0 has been set to eight different values to create eight variations of the problem. Considering possible discounts, the cost of the component is defined as a function of the number μ of components purchased:

$$C(\mu) = \begin{cases} C^* & \mu \le m_1 \\ \gamma_1 \times C^* & m_1 \le \mu \le m_2 \\ \gamma_2 \times C^* & \mu \ge m_2 \end{cases} \quad (14)$$

where, $C^* = C_{ij}$ is the cost per unit, and m_1, m_2, γ_1, γ_2 are shown in Table 13 for each component type

Table 14 and 15 present the data of Example 3. The availability index A0 has been set to three different values to create three variations of the problem. Table 16 completes the data of Example 4, for which three different values of the availability index A0 have been used to create three variations of the problem.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

6.1.2 Parameter Settings

For this problem, the user-specified parameters Δw, w_0 and w have been tuned by following the same tuning procedure used for binary-state systems. The found values to be most appropriate were: $1/w_0 = 0.0085$, $w = 50$ and $\Delta w = 0.0001$. In addition, preliminary experiments showed that the best results are obtained when the initial value of α is set to 10 with $a = 5$, $b = 5$, $c_1 = 0.99$ and $c_2 = 1.01$.

6.1.3 Comparison Results

The algorithm is evaluated by comparison to the existing benchmark methods: genetic algorithms (GA) (Levitin et al., 1998; Levitin et al., 1997; Lisnianski et al., 1996) and tabu search (Ouzineb et al., 2008). Table 1 shows the total cost of the NLTA solution for each of the 17 instances. The total cost for the best solutions found in 10 trials and the average cost, as well as the standard deviations (St. D) of the 10 solutions and the average cost, the standard deviations (St.D) of the 10 solutions and the average computation time. From this table, we can see that the standard deviation is very low and the average computation time does not exceed 200 seconds.

Table 1. Solutions found by NLTA (10 trials).

Problem	**Nonlinear Threshold Accepting Algorithm**				
	A_0	**Optimal cost**	**Average cost**	**St.D**	**Average time (sec.)**
Example 1	0.9	5.986	6.16	0.1	60
	0.96	7.303	7.43	0.13	
	0.99	8.328	8.4	0.13	
Example 2	0.91	14.886	14.96	0.09	78
	0.92	15.075	15.1	0.1	
	0.94	17.805	17.87	0.04	
	0.95	20.049	20.064	0.02	
	0.96	21.155	21.18	0.04	
	0.97	21.907	21.91	0.01	
	0.98	22.656	22.67	0.02	
	0.99	24.305	25.08	0.5	
Example 3	0.975	16.45	16.49	0.03	150
	0.98	16.52	16.58	0.04	
	0.99	17.05	17.09	0.02	
Example 4	0.975	11.241	11.24	0.01	200
	0.98	11.369	11.37	0.01	
	0.99	12.764	13.06	0.26	

Table 2 shows the total cost of the best solution for each of the 17 instances. The average (Av.) and the standard deviations (St.D) of the best solutions found in 10 trials are also presented in Table 2. This table shows that:

- In 6 of the 17 test cases, the NLTA outperforms the genetic algorithm, while it provides the same results in the other cases.
- In all cases tested, the results of the NLTA are in par with the best known results.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 2. Comparison results.

Problem	A_0	TS	GA	NLTA
		Optimal cost	Optimal cost	Optimal cost
Example 1	0.9	**5.986**	6.348	**5.986**
	0.96	**7.303**	7.571	**7.303**
	0.99	**8.328**	**8.328**	**8.328**
Example 2	0.91	**14.886**	15.812	**14.886**
	0.92	**15.075**	16.035	**15.075**
	0.94	**17.805**	**17.805**	**17.805**
	0.95	**20.049**	**20.049**	**20.049**
	0.96	**21.155**	**21.155**	**21.155**
	0.97	**21.907**	**21.907**	**21.907**
	0.98	**22.656**	**22.656**	**22.656**
	0.99	**24.305**	**24.305**	**24.305**
Example 3	0.975	**16.45**	**16.45**	**16.45**
	0.98	**16.52**	**16.52**	**16.52**
	0.99	**17.05**	17.095	**17.05**
Example 4	0.975	**11.241**	**11.241**	**11.241**
	0.98	**11.369**	12.608	**11.369**
	0.99	**12.764**	**12.764**	**12.764**

The above results clearly highlight that our approach outperforms the existing methods. The reasons of this performance are discussed in the next subsection.

6.2 ROP of Binary-State Systems

6.2.1 Test Problems

The results of the algorithm are compared to the results reported in the literature for the 33 problem instances introduced by (Fyffe et al., 1968; Nakagawa and Miyazaki, 1981). The system consists of 14 subsystems. In Appendix A, Table 17 provides all the input data for the examples studied in this paper, with r_{ij}, c_{ij} and w_{ij} are (respectively) the reliability, the cost and the weight of component j for subsystem i.

6.2.2 Parameter Settings

In meta-heuristic approaches, like simulated annealing and threshold accepting algorithm, to achieve high performance, it is essential to adjust a number of input parameters. In the case of NTAA, there are three user-specified input parameters: Δw, w_0 and w. Parameter values most favorable to the considered optimization model need to be explored. Therefore, similar to common practices used in the literature, 10 simulations were performed for each case, where value of one input parameter was explored while keeping others constant and some statistical information about the average evolution of the key parameters was collected. Multiple tests were carried out, setting with the available budget $C = 130$ and the allowed weight constraint $W = 191$, in order to identify the suitable values for the parameters. Maximum number of iterations (i.e. 2×10^6) was used as stopping criteria. By performing these experiments, the most suitable values for the parameters were: $1/w_0 = 0.04$, $w = 30$ and $\Delta w = 1.4 \times 10^{-5}$.

6.2.3 Comparison Results

Benchmark methods include genetic algorithm (GA) (Coit and Smith, 1996b), tabu search (TS) (Kulturel-Konak et al., 2003), the ant colony optimization (ACO) (Liang and Smith, 2004) and the

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Harmony search algorithm (HSA) (Nahas and Thien-My, 2010). The obtained results for 10 trials are shown in Table 3. This table presents the system reliability of the NLTA solution for each instance. The maximum (R_{max}) for the best solutions found in 10 trials and the average (R_{av}), the standard deviations (St.D) of the 10 solutions and the average computation times. From this table, we can see that the standard deviation is very low and the average computation time does not exceed 87 seconds for any instance. Figure 1 shows the convergence of the best solution of the NLTA for $C = 130$ and $W = 191$. From this figure, we can see that not more than 15×10^5 iterations are necessary for the algorithm to converge. Indeed, this number is much lower than the size of the search space (larger than 7.6×10^{33} for this instance example).

Table 4 presents the comparison results of the NLTA and the best solutions found in previous papers: GA (Coit and Smith, 1996b), TS (Kulturel-Konak et al., 2003), ACO (Liang and Smith, 2004) and HSA (Nahas and Thien-My, 2010). To illustrate the efficiency of the proposed approach (NLTA), the following errors are defined:

$$Error(\%) = 100 \times \frac{R_{max}(NLTA) - R_{max}(ALG)}{R_{max}(ALG)} \quad (15)$$

where, $R_{max}(NLTA)$ represents the best result obtained by NLTA and R_{max} (ALG) represents the best result obtained by another approach, i.e. GA, TS, ACO or HSA.

Figure 2 presents the errors obtained by NLTA relatively to the GA, TS, ACO and HSA. Table 4 and these figures clearly highlight that NLTA outperforms the other approaches.

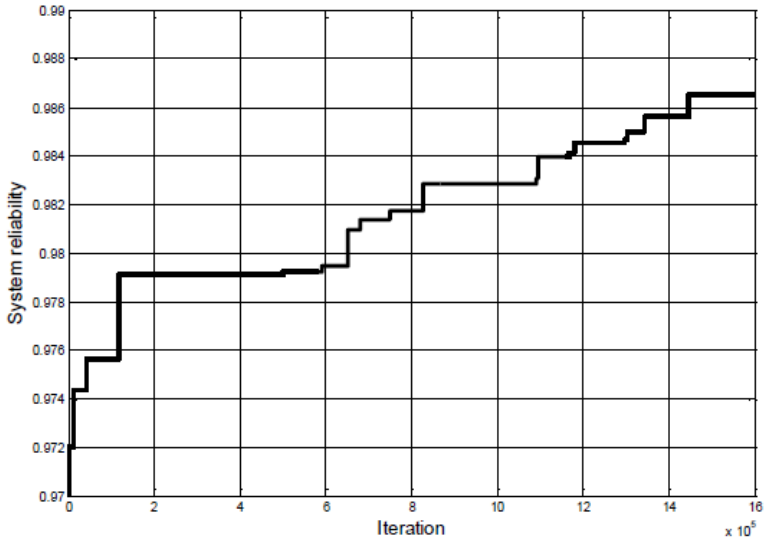

Figure 1. Solution convergence.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 3. Solutions found by NLTA (10 trials).

W	Nonlinear threshold Accepting Algorithm			
	$R_{\max}$	R_{av}	St.D. ($\times 10^{-3}$)	Average time (sec.)
191	0.986811	0.986463	0.40	73
190	0.986416	0.986161	0.25	67
189	0.985922	0.985628	0.26	64
188	0.985378	0.985193	0.46	87
187	0.984688	0.984425	0.30	65
186	0.984176	0.984093	0.25	64
185	0.983505	0.983304	0.16	65
184	0.982994	0.982868	0.33	64
183	0.982256	0.981975	0.19	64
182	0.981518	0.981343	0.39	80
181	0.981027	0.980500	0.24	64
180	0.980290	0.979723	0.39	62
179	0.979273	0.978711	0.34	67
178	0.978380	0.978231	0.39	65
177	0.977596	0.977401	0.15	62
176	0.976690	0.976516	0.14	57
175	0.975708	0.975624	0.09	61
174	0.974926	0.974833	0.07	55
173	0.973827	0.973708	0.06	54
172	0.973027	0.973026	0.25	52
171	0.971929	0.971929	0.00	50
170	0.970760	0.970760	0.28	54
169	0.969291	0.969181	0.45	43
168	0.968125	0.968125	0.44	57
167	0.966335	0.966147	0.49	42
166	0.965042	0.964845	0.27	41
165	0.963712	0.963400	0.40	48
164	0.962422	0.962188	0.89	36
163	0.960642	0.960341	0.07	33
162	0.959188	0.958938	0.28	38
161	0.958034	0.957386	0.35	33
160	0.955714	0.955643	0.25	41
159	0.954564	0.953798	0.67	24

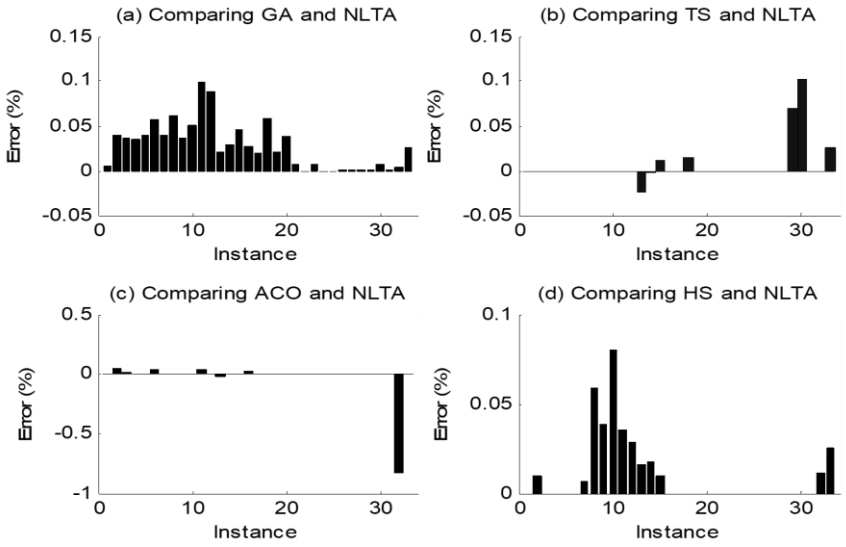

Figure 2. Comparing NLTA and four other metaheuristics.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 4. Comparison of the NLTA and the best known solutions.

W	NLTA	GA	TS	ACO	HSA
	R_{max}	R_{max}	R_{max}	R_{max}	R_{max}
191	**0.986811**	0.98675	**0.986811**	0.986745	**0.986811**
190	**0.986416**	0.98603	**0.986416**	0.985905	0.986316
189	**0.985922**	0.98556	**0.985922**	0.985773	**0.985922**
188	**0.985378**	0.98503	**0.985378**	0.985329	**0.985378**
187	**0.984688**	0.98429	**0.984688**	**0.984688**	**0.984688**
186	**0.984176**	0.98362	**0.984176**	0.983801	**0.984176**
185	**0.983505**	0.98311	**0.983505**	**0.983505**	0.983436
184	**0.982994**	0.98239	**0.982994**	**0.982994**	0.982417
183	**0.982256**	0.98190	**0.982256**	0.982206	0.981876
182	**0.981518**	0.98102	**0.981518**	0.981468	0.980729
181	**0.981027**	0.98006	**0.981027**	0.980681	0.980681
180	**0.980290**	0.97942	**0.980290**	**0.980290**	0.980009
179	0.979273	0.97906	**0.979505**	**0.979505**	0.979115
178	0.978380	0.97810	**0.978400**	**0.978400**	0.978208
177	**0.977596**	0.97715	0.977474	**0.977596**	0.977498
176	**0.976690**	0.97642	**0.976690**	0.976494	**0.976690**
175	**0.975708**	0.97552	**0.975708**	**0.975708**	**0.975708**
174	**0.974926**	0.97435	0.974788	**0.974926**	**0.974926**
173	**0.973827**	0.97362	**0.973827**	**0.973827**	**0.973827**
172	**0.973027**	0.97266	**0.973027**	**0.973027**	**0.973027**
171	**0.971929**	0.97186	**0.971929**	**0.971929**	**0.971929**
170	**0.970760**	**0.97076**	**0.970760**	**0.970760**	**0.970760**
169	**0.969291**	0.96922	**0.969291**	**0.969291**	**0.969291**
168	**0.968125**	**0.96813**	**0.968125**	**0.968125**	**0.968125**
167	**0.966335**	**0.96634**	**0.966335**	**0.966335**	**0.966335**
166	**0.965042**	**0.96504**	**0.965042**	**0.965042**	**0.965042**
165	**0.963712**	**0.96371**	**0.963712**	**0.963712**	**0.963712**
164	**0.962422**	**0.96242**	**0.962422**	**0.962422**	**0.962422**
163	**0.960642**	**0.96064**	0.959980	**0.960642**	**0.960642**
162	**0.959188**	0.95912	0.958205	**0.959188**	**0.959188**
161	**0.958034**	**0.95803**	**0.958034**	**0.958034**	**0.958034**
160	**0.955714**	0.95567	**0.955714**	0.963712	0.955604
159	**0.954564**	0.95432	0.954325	**0.954564**	0.954325

6.3 Discussion

In general, it is not always easy to justify why a given metaheuristic provides good results to solve a given problem. It is however widely recognized that a potentially good approach should seeks to balance between diversification and intensification strategies. Diversification explores the search space by examining unvisited regions, while intensification exploits a local region to look more closely around good solutions identified during the past search. Our method tends to find the right balance between diversification and intensification.

For the redundancy optimization problems, the best known solution methods are based on metaheuristic approaches such as GA, TS, ACO and HSA. The algorithm proposed in this paper is based on a simple iterative local search method. The impressive results found by this algorithm give rise to an important question: Why such simple method performs very well for the ROP? Our method has two important characteristics: the use of a self-adjusting penalty strategy, and the non-linearity of the acceptance rule. To analyze the effect of these characteristics, additional experiments were conducted to quantify their contributions to the algorithm efficiency.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 5. Results of the NLTA without penalty: case of binary-state systems.

Problem	**NLTA without penalty**	
	R_{max}	R_{av}
1	0.986745	0.98645
2	0.986316	0.98567
3	0.985922	0.98516
4	0.985378	0.98487
5	0.984688	0.98423
6	0.984146	0.98302
7	0.98277	0.9822
8	0.982994	0.98171
9	0.98222	0.98138
10	0.980803	0.9803
11	0.98045	0.97968
12	0.979556	0.97878
13	0.979273	0.97774
14	0.97838	0.97732
15	0.97663	0.97594
16	0.97669	0.97577
17	0.975478	0.97457
18	0.974926	0.97415
19	0.973827	0.97314
20	0.973027	0.97221
21	0.971929	0.97114
22	0.97076	0.97026
23	0.969291	0.96842
24	0.967499	0.96702
25	0.966335	0.96565
26	0.964872	0.96436
27	0.963712	0.95161
28	0.961331	0.9571
29	0.960642	0.95953
30	0.957925	0.93808
31	0.958035	0.94547
32	0.955367	0.94084
33	0.9536	0.94915

Table 6. Results of the NLTA without penalty: case of multi-state systems.

Problem	A_0	**NLTA without penalty**	
		Optimal cost	Average cost
	0.9	6.198	6.295
Example 1	0.96	7.498	7.6551
	0.99	8.328	8.503
	0.91	14.88	14.88
	0.92	15.075	15.075
	0.94	17.805	17.805
	0.95	20.049	20.049
Example 2	0.96	21.336	21.646
	0.97	21.956	21.956
	0.98	22.706	22.706
	0.99	24.354	24.354
	0.975	16.457	16.776
Example 3	0.98	16.548	16.939
	0.99	17.06	17.330
	0.975	11.319	11.676
Example 4	0.98	11.369	11.966
	0.99	12.764	13.039

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

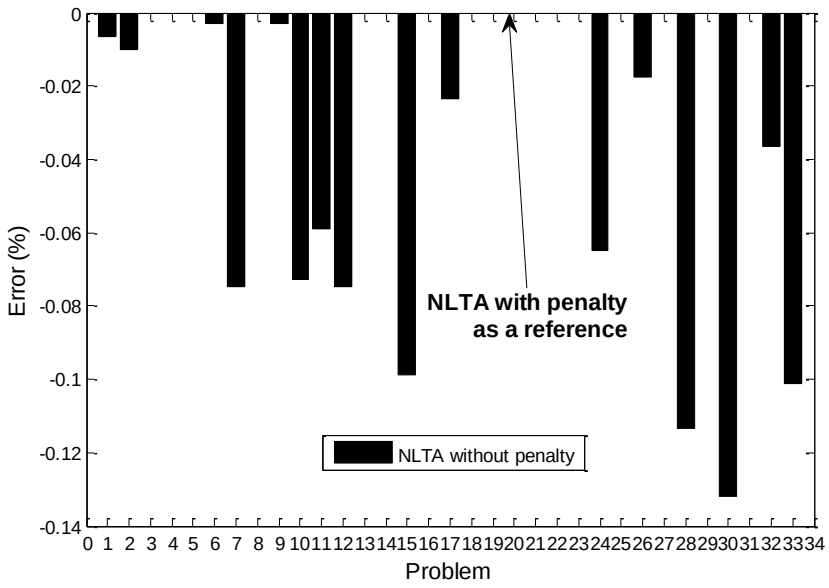

Figure 3. Effect of the penalty on NLTA in case of binary-state systems.

Tables 5 and 6 present the results obtained without the penalty strategy for the ROP of binary and multi-state systems, respectively. Figures 3 and 4 present the errors between the NLTA versions with and without penalties and show that the used penalty strategy plays an important role in solving the ROP.

Furthermore, to analyze the importance of the non-linearity, we tested the conventional threshold accepting algorithm (Dueck and Scheuer, 1990), with and without penalty strategies. Tables 7 and 8 present the comparison results, while Figures 5 and 6 sketch the errors and highlight two important points. First, the linear threshold algorithm without penalty strategy provides relatively bad solutions. Second, the introduction of the penalty strategy into the linear threshold algorithm improves the solution quality, but the overall efficiency remains much less competitive in comparison to existing methods. This means that the efficiency of our method comes not only from the use of the self-adjusting penalty strategies, but from the non-linear form of the acceptance rules as well. Under such a non-linear control of the search process speed, the intensification is also encouraged, and the solution acceptance depends on the value of the fraction between the objective value of the current solution and the value of its neighbor.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

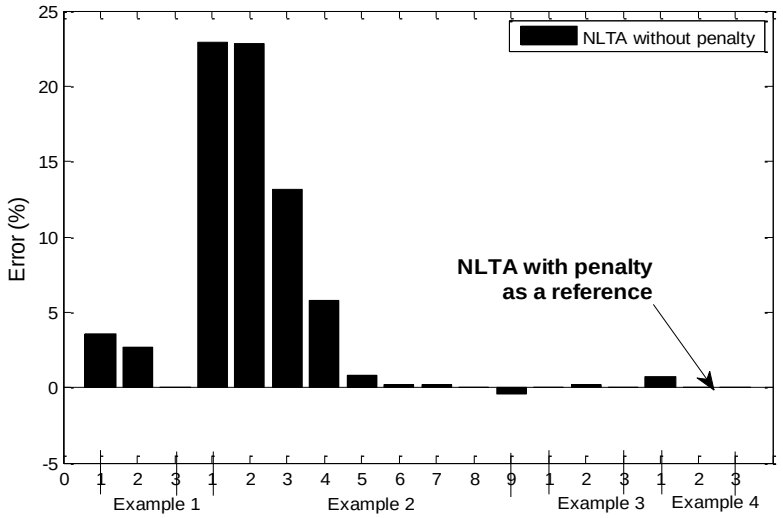

Figure 4. Effect of the penalty on NLTA in case of multi-state systems.

Table 7. Results obtained by TA with and without penalty in case of binary-state systems.

Problem	Threshold with penalty		Threshold without penalty	
	R_{max}	R_{av}	R_{max}	R_{av}
1	0.98674	0.98628	0.98671	0.98607
2	0.98631	0.98542	0.98631	0.98533
3	0.98592	0.98508	0.98592	0.98486
4	0.98537	0.98420	0.98537	0.98438
5	0.98468	0.98398	0.98468	0.98359
6	0.98399	0.98321	0.98414	0.98310
7	0.98350	0.98275	0.98350	0.98255
8	0.98276	0.98205	0.98241	0.98213
9	0.98187	0.98119	0.98225	0.98125
10	0.98123	0.98052	0.98146	0.98058
11	0.98045	0.97947	0.98040	0.97911
12	0.98000	0.97899	0.98000	0.97901
13	0.97927	0.97845	0.97927	0.97813
14	0.97833	0.97724	0.97838	0.97768
15	0.97759	0.97658	0.97759	0.97647
16	0.97669	0.97584	0.97645	0.97585
17	0.97542	0.97463	0.97488	0.97457
18	0.97492	0.97442	0.97492	0.97428
19	0.97380	0.97343	0.97380	0.97326
20	0.97302	0.97217	0.97302	0.97193
21	0.97192	0.97148	0.97192	0.97100
22	0.97076	0.96954	0.97076	0.97008
23	0.96929	0.96850	0.96929	0.96868
24	0.96812	0.96725	0.96812	0.96684
25	0.96633	0.96560	0.96578	0.96538
26	0.96487	0.96394	0.96487	0.96427
27	0.96371	0.96261	0.96371	0.96268
28	0.96179	0.96098	0.96179	0.96098
29	0.95989	0.95888	0.96017	0.95920
30	0.95918	0.95796	0.95861	0.95764
31	0.95675	0.95634	0.95681	0.95592
32	0.95566	0.95448	0.95566	0.95439
33	0.95328	0.95224	0.95342	0.95238

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 8. Results obtained by TA with and without penalty in case of multi-state systems.

Problem	A_0	Threshold with penalty		Threshold without penalty	
		Optimal cost	Average cost	Optimal cost	Average cost
	0.9	6.198	6.4699	6.262	6.7735
Example 1	0.96	7.303	7.5564	7.303	7.6069
	0.99	8.328	8.7134	8.328	8.6668
	0.91	19.107	20.854	18.293	21.686
	0.92	16.507	19.572	18.516	21.695
	0.94	17.857	21.580	20.146	22.676
	0.95	20.255	22.080	21.215	22.831
Example 2	0.96	21.155	22.759	22.446	23.652
	0.97	22.423	22.952	22.535	23.827
	0.98	22.960	24.235	23.57	24.368
	0.99	24.451	26.199	25.174	26.741
	0.975	16.457	16.588	16.45	16.980
Example 3	0.98	16.52	16.713	16.676	16.946
	0.99	17.057	17.141	17.06	17.267
	0.975	11.241	13.975	16.25	18.204
Example 4	0.98	11.516	15.723	11.369	17.316
	0.99	12.911	16.145	12.764	19.836

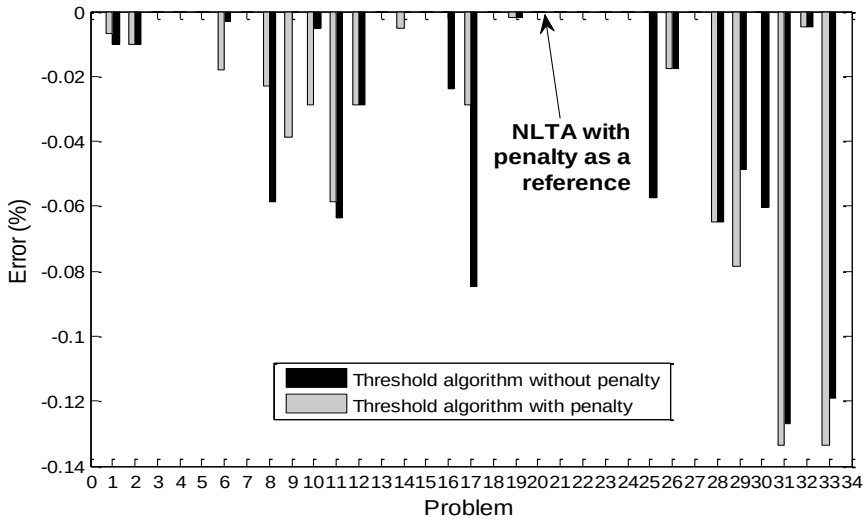

Figure 5. Comparing threshold algorithm with and without penalty in case of binary-state systems.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

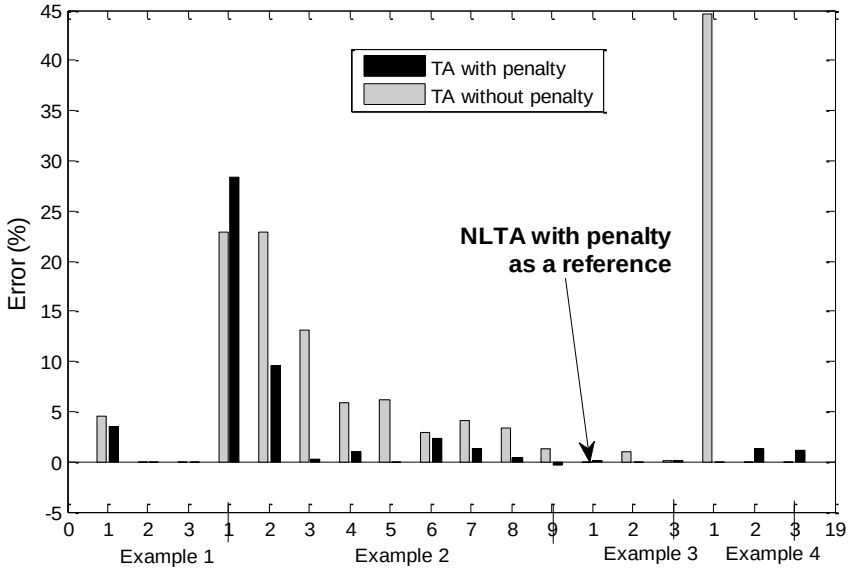

Figure 6. Comparing threshold algorithm with and without penalty in case of multi-state systems.

7. Conclusion

In this paper, we developed an evaluation/optimization approach to solve the redundancy optimization problem of series-parallel multi-state systems. This approach provides a powerful tool to reliability designers, when different varieties of components can be considered to achieve high-availability for complex multi-state systems. Extensive numerical results for the test problems from previous research highlight the advantages of the proposed approach in terms of solution quality, computation time and implementation effort. Using existing benchmark problems, it was found that this approach out-performs the state-of-the-art benchmark methods dealing with the ROP of multi-state systems. Not only the proposed approach is shown to be efficient, but it is also simple to implement. Two important features characterise the paper contribution and justify the efficiency of the proposed approach. First, the availability evaluation method uses the universal moment generating function technique, which is convenient for numerical implementation in high dimension combinatorial optimization. Second, the optimization algorithm is adaptable and uses a non-linear acceptance function. Such non-linearity and adaptability are desirable in the search process of good solutions. Furthermore, by using a self-adjusting penalty guided strategy, this algorithm is encouraged to explore both feasible and infeasible regions.

To achieve high availability levels for series-parallel multi-state systems, this paper considered that each subsystem was composed of identical components (i.e. homogeneous systems). It was also assumed that the characteristics of components are known and deterministic. While the extension to heterogeneous systems is straightforward, taking into account parameters uncertainties may require further methodological development. We are currently working on these issues, and on the extension of our algorithm as general-purpose method to deal with others combinatorial optimization problems encountered in the context of multi-state system reliability.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Appendix A. Input data for the studied examples.

The data for multi-state systems are presented in Tables 9-16 and those of binary-state systems examples are in Table 17.

Table 9. Data of the components available in the market (Example 1).

Subsystem i	Component version j	Availability A_{ij}	Cost C_{ij} (mln $)	Performance G_{ij} (%)
1	1	0.97	0.52	50
	2	0.964	0.62	80
	3	0.98	0.72	80
	4	0.969	0.89	100
	5	0.96	1.02	150
2	1	0.967	0.516	20
	2	0.914	0.916	50
	3	0.96	0.967	50
	4	0.953	1.367	75
3	1	0.959	0.214	60
	2	0.97	0.384	90
	3	0.959	0.534	180
	4	0.96	0.614	200
	5	0.97	0.783	200
	6	0.96	0.813	240
4	1	0.989	0.683	25
	2	0.979	0.645	25
	3	0.98	0.697	30
	4	0.96	1.19	70
	5	0.98	1.26	70

Table 10. Parameters of the cumulative load demand curve (Example 1).

Demand level (%)	100	80	50	20
Duration (h)	4260	800	1200	2496

Table 11. Data of the components available in the market (Example 2).

Subsystem i	Component version j	Availability A_{ij}	Cost C_{ij} (mln $)	Nominal Performance G_{ij} (%)
1	1	0.99	1.117	25
	2	0.996	1.31	25
	3	0.995	1.903	35
	4	0.961	1.64	35
	5	0.993	2.122	50
	6	0.957	1.91	50
	7	0.942	1.722	50
	8	0.991	2.591	72
	9	0.951	2.001	72
	10	0.986	3.284	100
	11	0.979	3.095	100

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 11 continued…

2	1	0.967	4.01	40
	2	0.914	3.45	40
	3	0.96	4.35	55
	4	0.953	4.84	78
	5	0.92	4.21	78
	6	0.95	5.8	93
	7	0.948	6.55	110
3	1	0.967	0.636	25
	2	0.952	0.448	35
	3	0.973	0.868	35
	4	0.972	0.964	50
	5	0.949	0.678	50
	6	0.988	1.096	50
	7	0.966	1.358	72
	8	0.954	1.298	72
	9	0.945	1.81	100
4	1	0.987	0.614	12.5
	2	0.985	0.883	25
	3	0.961	0.745	25
	4	0.98	0.963	30
	5	0.958	0.885	30
	6	0.974	1.338	45
	7	0.982	1.445	45

Table 12. Parameters of the cumulative load demand curve (Example 2).

Demand level (%)	100	80	40
Duration (h)	20	30	50

Table 13. Parameters of the cost function with discounts (Example 2).

Component	m_1	m_2	γ_1	γ_2
1	3	5	0.9	0.8
2	3	3	0.85	0.85
3	3	3	0.95	0.95
4	2	6	0.95	0.9

Table 14. Data of the components available in the market (Example 3).

Subsystem i	Component version j	Availability A_{ij}	Cost C_{ij} (mln $)	Performance G_{ij} (%)
1	1	0.98	0.59	120
	2	0.977	0.535	100
	3	0.982	0.47	85
	4	0.978	0.42	85
	5	0.983	0.4	48
	6	0.92	0.18	31
	7	0.984	0.22	26
2	1	0.995	0.205	100
	2	0.996	0.189	92
	3	0.997	0.091	53
	4	0.997	0.056	28
	5	0.998	0.042	21

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 14 continued...

3	1	0.971	7.525	100
	2	0.973	4.72	60
	3	0.971	3.59	40
	4	0.976	2.42	20
4	1	0.977	0.18	115
	2	0.978	0.16	100
	3	0.978	0.15	91
	4	0.983	0.121	72
	5	0.981	0.102	72
	6	0.971	0.096	72
	7	0.983	0.071	55
	8	0.982	0.049	25
	9	0.977	0.044	25
5	1	0.984	0.986	128
	2	0.983	0.825	100
	3	0.987	0.49	60
	4	0.981	0.475	51

Table 15. Parameters of the cumulative load demand curve (Examples 3 and 4).

Demand level (%)	100	80	50	20
Duration (h)	4203	788	1228	2536

Table 16. Data of the components available in the market (Example 4).

Subsystem i	**Component** version j	**Availability** A_{ij}	**Cost** C_{ij} (mln $)	**Performance** G_{ij} (%)
1	1	0.932	1.59	27.3
	2	0.998	0.515	27.7
	3	0.983	0.225	49.8
	4	0.927	3.22	52.5
	5	0.959	4.02	62
	6	0.955	4.27	66.4
	7	0.984	3.67	84.6
	8	0.918	4.63	90.7
	9	0.939	1.01	97
	10	0.988	0.779	124
	11	0.984	3.13	129
2	1	0.989	0.05	35.9
	2	0.923	1.29	44.7
	3	0.9	0.204	51.4
	4	0.946	2.22	63.2
	5	0.917	0.872	68.8
	6	0.962	1.83	81.8
	7	0.994	0.294	82
	8	0.984	2.81	115
3	1	0.931	3.62	34.7
	2	0.95	0.475	41
	3	0.911	1.17	41.4
	4	0.956	0.793	43.6
	5	0.966	3.74	48.6
	6	0.992	4.59	59.6
	7	0.929	1.74	66.2
	8	0.968	1.72	91.9
	9	0.901	1.3	121

International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 1, 416-441, 2021
https://doi.org/10.33889/IJMEMS.2021.6.1.026

Table 16 continued…

4	1	0.915	2.49	25.1
	2	0.908	0.078	28.8
	3	0.928	1.37	50.2
	4	0.944	4.47	129
5	1	0.908	1.55	34.9
	2	0.98	4.92	64.3
	3	0.964	2.65	108
	4	0.924	4.72	126
6	1	0.965	3.22	24.8
	2	0.927	2.89	47.3
	3	0.986	3.41	58.8
	4	0.983	1.92	107
	5	0.991	4.51	120
	6	0.954	4.58	125

Table 17. Component parameters for testing problem.

i	r_{i1}	c_{i1}	w_{i1}	r_{i2}	c_{i2}	w_{i2}	r_{i3}	c_{i3}	w_{i3}	r_{i4}	c_{i4}	w_{i4}
1	0.95	2	5	0.93	1	4	0.91	2	2	0.90	1	3
2	0.95	2	8	0.94	1	10	0.93	1	9			
3	0.92	4	4	0.90	3	5	0.87	1	6	0.85	2	7
4	0.87	4	6	0.85	5	4	0.83	3	5			
5	0.95	3	5	0.94	2	4	0.93	2	3			
6	0.99	3	5	0.98	3	4	0.97	2	5	0.96	2	4
7	0.94	5	9	0.92	4	8	0.91	4	7			
8	0.91	6	6	0.90	5	7	0.81	3	4			
9	0.99	3	9	0.97	2	8	0.96	4	7	0.91	3	8
10	0.90	5	6	0.85	4	5	0.83	4	6			
11	0.96	5	6	0.95	4	6	0.94	3	5			
12	0.90	5	7	0.85	4	6	0.82	3	5	0.79	2	4
13	0.99	3	5	0.98	2	5	0.97	2	6			
14	0.99	6	9	0.95	5	6	0.92	4	7	0.90	4	6

Confilct of Interest
The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgement
The authors would like to thank the reviewers and the editor for their valuable and helpful recommendations to improve the paper.

References

Agarwal, M., & Gupta, R. (2006). Genetic search for redundancy optimization in complex systems. *Journal of Quality in Maintenance Engineering*, *12*(4), 338–353.

Agarwal, M., Aggarwal, S., & Sharma, V.K. (2010). Optimal redundancy allocation in complex systems. *Journal of Quality in Maintenance Engineering*, *16*(4), 413–424.

Bellman, R., & Dreyfus, S. (1958). Dynamic programming and the reliability of multicomponent devices. *Operations Research*, *6*(2), 200–206.

Billionnet, A. (2008). Redundancy allocation for series-parallel systems using integer linear programming. *IEEE Transactions on Reliability*, *57*(3), 507–516.

Bulfin, R.L., & Liu, C.Y. (1985). Optimal allocation of redundant components for large systems. *IEEE Transactions on Reliability*, *R-34*(3), 241–247.

Caserta, M., & VoB, S. (2015a). A discrete-binary transformation of the reliability redundancy allocation problem. *Mathematical Problems in Engineering*, *2015*, 1–6.

Caserta, M., & VoB, S. (2015b). An exact algorithm for the reliability redundancy allocation problem. *European Journal of Operational Research*, *244*(1), 110–116.

Caserta, M., & VoB, S. (2016). A corridor method based hybrid algorithm for redundancy allocation. *Journal of Heuristics*, *22*(4), 405–429.

Chang, K.H., & Kuo, P.Y. (2018). An efficient simulation optimization method for the generalized redundancy allocation problem. *European Journal of Operational Research*, *265*(16), 1094–1101.

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

Coit, D.W., & Smith, A.E. (1996a). Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering*, *30*(4),895–904.

Coit, D.W., & Smith, A.E. (1996b). Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, *45*(2), 254–260.

Dueck, G., & Scheuer, T. (1990). Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, *90*(1), 161–175.

Fyffe, D.E., Hines, W.W., & Lee, N.K. (1968). System reliability allocation and a computational algorithm. *IEEE Transactions on Reliability*, *R-17*(2), 64–69.

Gen, M., Ida, K., Tsujimura, Y., & Kim, C.E. (1993). Large-scale 0-1 fuzzy goal programming and its application to reliability optimization problem. *Computers & Industrial Engineering*, *24*(4), 539–549.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problems. *Management Science*, *40*(10), 1276–1290.

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

Kim, H., Bae, C., & Park, D. (2006). Reliability and redundancy optimization using simulated annealing algorithms. *Journal of Quality in Maintenance Engineering*, *12*(4), 354–363.

Kulturel-Konak, S., Smith, A.E., & Coit, D.W. (2003). Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions*, *35(*6), 515–526.

Kuo, W., & Prasad, R. (2000). An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability*, *49*(2), 176–187.

Kuo, W., Rajendra, V.P., Tillman, F.A., & Hwang, C.L. (2001). Optimal reliability design: fundamentals and applications. *Cambridge University Press, Cambridge*, UK.

Levitin, G., & Lisnianski, A. (2001). A new approach to solving problems of multi-state system reliability optimization. *Quality and Reliability Engineering International*, *47*(2), 93–104.

Levitin, G., Lisnianski, A., & Elmakis, D. (1997). Structure optimization of power system with different redundant elements. *Electric Power Systems Research, 43*(1), 19–27.

Levitin, G., Lisnianski, A., Ben-Haim, H., & Elmakis, D. (1998). Redundancy optimization for series-parallel multi-state systems. *IEEE Transactions on Reliability*, *47*(2), 165–172.

Levitn, G. (2005). *Universal generating function in reliability analysis and optimization*. Springer, Berlin.

Liang, Y.C., & Smith, A.E. (2004). An ant colony optimization algorithm for the redundancy allocation problem (rap). *IEEE Transactions on Reliability*, *53*(3), 417–423.

Lisnianski, A., & Levitin, G. (2003). *Multi-state system reliability: assesment, optimization and applications*. World Scientific, Singapore.

Lisnianski, A., Levitin, G., Ben-Haim, H., & Elmakis, D. (1996). Power system structure optimization subject to reliability constraints. *Electric Power Systems Research*, *39*(2), 145–152.

Liu, Y. (2016). Improved bat algorithm for reliability-redundancy allocation problems. *International Journal of Security and its Applications*, *10*(2), 1–12.

Misra, K.B,. & Sharma, U. (1991). An efficient algorithm to solve integer-programming problems arising in system-reliability design. *IEEE Transactions on Reliability*, *40*(1), 81–91.

Misra, K.B., & Ljubojevic, M.D. (1973). Optimal reliability design of a system: a new look. *IEEE Transactions on Reliability*, *R-22*(5), 255–258.

Nahas, N., & Nourelfath, M. (2014). Nonlinear threshold accepting meta-heuristic for combinatorial optimisation problems. *International Journal of Metaheuristics*, *3*(4), 265-290.

Nahas, N., & Thien-My, D. (2010). Harmony search algorithm: application to the redundancy optimization problem. *Engineering Optimization*, *42*(9), 845–861.

Nahas, N., Darghouth, M.N., Karaa, A.Q., & Nourelfath, M. (2018). Non-linear threshold algorithm based solution for the redundancy allocation problem considering multiple redundancy strategies. *Journal of Quality in Maintenance Engineering*, *25*(3), 397-411.

Nahas, N., Nourelfath, M., & Ait-Kadi, D. (2007). Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series-parallel systems. *Reliability Engineering and System Safety*, *92*(2), 211–222.

Nahas, N., Nourelfath, M., & Ait-Kadi, D. (2008). Ant colonies for structure optimisation in a failure prone series-parallel production system. *Journal of Quality in Maintenance Engineering*, *14*(1), 7–33.

Nahas, N., Nourelfath, M., & Gendreau, M. (2014). Selecting machines and buffers in unreliable assembly/disassembly manufacturing networks. *International Journal of Production Economics*, *154*, 113–126.

Nakagawa, Y., & Miyazaki, S. (1981). Surrogate constraints algorithm for reliability optimization problems with two constraints. *IEEE Transactions on Reliability*, *R-30*(2), 175–180.

Nourelfath, M., & Nahas, N. (2005). Artificial neural networks for reliability maximization under budget and weight constraints. *Journal of Quality in Maintenance Engineering*, *11*(2), 139–151.

Nourelfath, M., Nahas, N., & Ait-Kadi, D. (2005). Optimal design of series production lines with unreliable machines and finite buffers. *Journal of Quality in Maintenance Engineering*, *11*(2), 121–138.

Nourelfath, M., Nahas, N., & Zeblah, A. (2003). An ant colony approach to redundancy optimization for multi-state system. *International Conference On Industrial Engineering And Production Management (IEPM'2003)*. Porto, Portugal.

Onishi, J., Kimura, S., James, R.J., & Nakagawa, Y. (2007). Solving the redundancy allocation problem with a mix of components using the improved surrogate constraint method. *IEEE Transactions on Reliability*, *56*(1), 94–101.

Ouzineb, M., Nourelfath, M., & Gendreau, M. (2008). Tabu search for the redundancy allocation problem of homogenous series-parallel multi-state systems. *Reliability Engineering & System Safety*, *93*(8), 1257–1272.

Ouzineb, M., Nourelfath, M., & Gendreau, M. (2010). An efficient heuristic for reliability design optimization problems. *Computers & Operations Research*, *37*(2), 223–235.

Painton, L., & Campbell, J. (1995). Genetic algorithms in optimization of system reliability. *IEEE Transactions on Reliability*, *44*(2), 172–178.

Ramirez, M., & Coit, D.W. (2004). A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems. *Reliability Engineering and Systems Safety*, *83*(3), 341–349.

Tillman, F.A., Hwang, C.L., & Kuo, W. (1977). Determining component reliability and redundancy for optimum system reliability. *IEEE Transactions on Reliability*, *R-26*(3), 162–165.

Ushakov, I. (1986). Universal Generating Function. *Soviet Journal Computer Systems Science*, *24*(5), 118–129.

Ushakov, I. (1987). Optimal standby problems and a universal generating function. *Soviet Journal Computer Systems Science*, *25*(4), 79–82.

Ushakov, I., Levitin, G., & Lisnianski, A. (2002). Multi-state system reliability: from theory to practice. *Proceedings of the Third International Conference on Mathematical Methods in Reliability (MMR'2002)*, Trondheim, Norway, 635–638.

Xue, J., & Yang, K. (1995). Dynamic reliability analysis of coherent multi-state systems. *IEEE Transactions on Reliability*, *44*(4),683–688.

Yalaoui, A., Chatelet, E., & Chu, C. (2005). A new dynamic programming method for reliability redundancy allocation in a parallel-series system. *IEEE Transactions on Reliability*, *54*(2), 254–261.

Yeh, W.C., & Hsieh, T.J. (2011). Solving reliability redundancy allocation problems using an artificial bee colony algorithm. *Computers & Operations Research*, *38*(11), 1465–1473.

Yokota, T., Gen, M., & Ida, K. (1995). System reliability optimization problems with several failure modes by genetic algorithm. *Journal of Japan Society for Fuzzy Theory and Systems*, *7*(1), 177–185.

Yokota, T., Gen, M., & Li, Y.X. (1996). Genetic algorithm for nonlinear mixed-integer programming and its applications. *Computers and Industrial Engineering*, *30*(4), 905–917.

Zhao, J.H., Liu, Z., & Dao, M.T. (2007). Reliability optimization using multiobjective ant colony system approaches. *Reliability Engineering & System Safety*, *92*(1), 109–120.