A Novel Genetic Algorithm Developed on a Reduced Search Space for Optimal Redundancy Allocation in Multi-State Series-Parallel Systems

Mu-Xia Sun, Yan-Fu Li, Enrico Zio

A Novel Genetic Algorithm Developed on a Reduced Search Space for Optimal Redundancy Allocation in Multi-State Series-Parallel Systems

M. X. Sun & Y. F. Li
Chair on Systems Science and the Energetic Challenge, Fondation EDF, CentraleSupélec, France
E. Zio
Chair on Systems Science and the Energetic Challenge, Fondation EDF, CentraleSupélec, France
Energy Department, Politecnico di Milano, Milano, Italy

ABSTRACT: We consider the redundancy allocation problem (RAP) for multi-state series-parallel systems (MSSPSs), with the objective of maximizing system availability under cost constraint. We show that, to obtain the global optimum, an evolutionary algorithm (EA) only needs to search within a subset of the feasible region. To exploit this, a repair algorithm is designed in a genetic algorithm (GA), to ensure it generates individuals only within the boundary subset, without increasing its computational complexity. We also add a novel mutation operator to improve local search efficiency, via generalizing the local search operators that appear in previous MSSPS RAP literature. In benchmark test, our modified GA outperforms the comparative algorithms and finds a solution better than the published one.

1 INTRODUCTION

The redundancy allocation problem (RAP) is a well-known optimization problem of relevance for the design of industrial systems, for which high reliability required (Levitin, G. et al. 1998, Lisnianski, A. et al. 1996, etc). The objective is to allocate redundancies, for an optimal trade-offs between the reliability and the cost of the system, while all component and system constraints are satisfied (When repairs are considered, availability is used instead of reliability.). In principle, most RAPs are solvable by exact method such as branch-and-bound (Kuo, W. & Prasad 2000, Kuo & Rui 2007). However, the NP-hard nature of RAP makes it a challenge for practical applications and algorithms must be designed that are able to produce approximate optimal solutions with low computational time. Meta-heuristic methods, particularly evolutionary algorithms (EAs) such as genetic algorithms (GAs), have been successfully applied for this.

In reliability engineering, systems are modeled as binary-state systems (BSSs) or multi-state systems (MSSs). The former have only two possible states, perfect functioning and complete failure, whereas the latter, which allows system and components to have finite number of states from perfect functioning to failure.

With regards to system topology, series-parallel system (SPS) is one of the most common designs analyzed for redundancy allocation (Kuo & Rui 2007, Wang & Li 2012). In this paper, the availability maximization model and cost minimization model for MSSPS RAP are denoted as "max MSSPS RAP" and "min MSSPS RAP", respectively.

MSSPS RAP is more difficult than BSSPS RAP. According to Yalaoui et al. (2005), we regard BSSPS RAP is equivalent to a two-stage one-dimension knapsack problem, after a simple log-transformation, and thus it is NP-complete. The MSSPS is more difficult, even the availability evaluation is exponential in time. For efficiency, meta-heuristics appear more suitable for solving MSSPS RAP than exact approaches, since they can provide approximate optimal solutions in with much less time, even though they can still be rather time consuming for large systems. The min MSSPS RAP has been solved by meta-heuristics such as GA (Kuo & Rui 2007), Tabu search (TS) (Ouzineb et al. 2011), particle swarm optimization (PSO) (Wang & Li 2012), etc. On the contrary, max MSSPS RAP is not so frequently studied, e.g. in (Gupta & Agarwal 2006), where a GA with dynamic penalty function is presented.

We study max MSSPS RAP in this work. We start from analyzing the monotonicity of MSSPSs and, then, show that an EA only need to search within a subset of the feasible region. Then, we modify GA according to this particular property: a repair algorithm and a novel mutation operator are designed

to ensure that GA will only generate individuals within the subset.

The rest of this paper is organized as follows: in section II the formulation of max MSSPS RAP is presented; in section III the monotonicity property of MSSPS RAP and the research space reduction are presented; in section IV the repair algorithm, the novel mutation operator and the overall modified GA are presented; in section V our modified GA is tested on 3 benchmarks, comparing with the original GA and the published MSSPS RAP results; section VI concludes the study.

2 STATEMENT OF THE PROBLEM

2.1 *The MSSPS*

The structure of a MSSPS is shown in Fig.1. The MSSPS is composed of N subsystems connected in series. Each subsystem $i \in \{1,2,\dots,N\}$ is composed of functionally equivalent components in parallel and there are n_i versions of components available on the market. For each component version $j \in \{1,2,\dots,n_i\}$ in subsystem i ("version j in subsystem i" will be denoted as "version ij" in the rest of this paper), there are at most X_{ij} components available.

The states of the components in the system are assumed to be mutually *s*-independent. Each component version ij is characterized by its cost C_{ij} and state distribution S_{ij}, which is determined by the following sets:

$$G_{ij} = \{g_{k_{ij}} | g_{k_{ij}} \in \mathbb{R}_{\geq 0}, k_{ij} \in \mathbb{N}, k_{ij} \leq K_{ij}\} \quad (1)$$

$$P_{ij} = \{p_{k_{ij}} | p_{k_{ij}} \in \mathbb{R}_{\geq 0}, k_{ij} \in \mathbb{N}, k_{ij} \leq K_{ij}\} \quad (2)$$

where G_{ij} is the set of states and P_{ij} is the set of the corresponding probabilities of each component version ij. The states in G_{ij} are sorted in ascending order, with $g_0 = 0$ as "complete failure" and $g_{K_{ij}} = \max\{G_{ijh}\}$ as "perfect functioning".

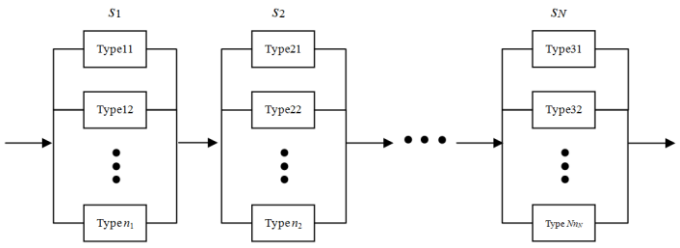

Figure 1. The MSSPS

The state of each component h of version ij is given by a random variable (RV) $G_{ijh} \sim S_{ij}$, i.e., G_{ijh}s are *i.i.d.* to S_{ij} for each ij. Then the state of each subsystem i is defined as:

$$G_i = \sum_{j=1}^{N_i} \sum_{h=1}^{x_{ij}} G_{ijh}, i \in \{1,2,\dots,N\} \quad (3)$$

where x_{ij} is the number of components of version ij. Then, the state of the whole system is defined as

$$G_s = \min_{i=1,2\dots,N} G_i \quad (4)$$

and, then, the availability of the system is defined as

$$A = \Pr(G_s \geq W) \quad (5)$$

where W is a RV denoting system demand, following an arbitrary discrete distribution defined by

$$G_D = \{d | d_k \in \mathbb{R}_{\geq 0}, k \in \mathbb{N}, k \leq K\} \quad (6)$$

$$P_D = \{p | p_k \in \mathbb{R}_{\geq 0}, k \in \mathbb{N}, k \leq K\} \quad (7)$$

where G_D is the set of demand states and P_D gives the probabilities for the respective states. After algebraic manipulations, the availability of the system can be written as the following formula

$$A = \sum_{k=1}^{K} \prod_{i=1}^{N} \Pr(G_i \geq W_k) p_k \quad (8)$$

Note that A can be directly calculated given each $\Pr(G_i \geq W_k)$, which can be evaluated by enumeration approaches such as universal generating function (UGF), as shown in (Wang & Li 2012).

2.2 *The MSSPS RAP*

Let us define the decision vector

$$\vec{x} = (x_{11}, x_{12}, \dots x_{ij}, \dots, x_{Nn_N}) \quad (9)$$

and the cost constraint

$$C(\vec{x}) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} c_{ij} x_{ij} \leq C_0 \quad (10)$$

Then, max MSSPS RAP is defined as:

Problem 2.1

$$\max A(\vec{x}) \quad (11)$$

s. t.

$$C(\vec{x}) \leq C_0 \quad (12)$$

$$x_{ij} \leq X_0 \quad (13)$$

$$x_{ij} \in \mathbb{N} \quad (14)$$

where C_0 is the maximum system cost and X_0 is the maximum number for all versions of components.

3 MONOTONICITY AND SEARCH SPACE REDUCTION

3.1 *Monotonicity*

Property 3.1

$A(\vec{x})$ is monotonically increasing on the feasible solution set F.

The monotonicity of MSSPS availability can be presented in partial order approach. Let us define the partial relation $\succcurlyeq$ on F as

$$\vec{x}' \succcurlyeq \vec{x}'' \ \textit{iff} \ \forall ij: x'_{ij} \geq x''_{ij}$$

then, we have

$$\forall \vec{x}' \, \forall \vec{x}'' : \vec{x}' \succcurlyeq \vec{x}'' \Rightarrow A(\vec{x}') \geq A(\vec{x}'')$$

i.e. $A(\vec{x})$ is monotone increasing on F.

3.2 *Search space reduction*

Let us define

$$B_c = \{\vec{x} \in F : \min(\{c_{ij} : x_{ij} < X_0\}) > d\},$$

where

$$d \stackrel{\text{def}}{=} C_0 - C(\vec{x})$$

Assume $\vec{x}' \in B_c$ and $\vec{x}'' \succ \vec{x}'$, then $C(\vec{x}'') > C_0$ since $C(\vec{x})$ is strictly monotone increasing on F (In partial order, " $\succ$" refers to "$\succcurlyeq and \neq$"). Hence, B_c is a subset of the boundary of the feasible region constrained by the cost.

Theorem 3.2

There is at least one global optimal solution for Problem 2.1 in the set B_c if $B_c \neq \emptyset$.

Proof

$$F \backslash B_c = \{\vec{x} \in F : \min(\{c_{ij} : x_{ij} < X_0\}) \leq d\}$$

Let us use the following algorithm to modify the number of components for one arbitrary feasible solution $\vec{x}'' \in F \backslash B_c$.

Step 1. find one component version ij s.t.

$$c_{ij} \leq d, x_{ij} < X_0$$

Step 2. $x_{ij} = x_{ij} + 1$

Step 3. update $\vec{x}''$ and d according to step 2.

Step 4. terminate if

$$\{ij : c_{ij} \leq d, x_{ij} < X_0\} = \emptyset$$

otherwise go back to step 1.

The algorithm will terminate in finite steps since $C(\vec{x})$ is monotonically increasing on F. Then, we shall have either

$$\{ij : x_{ij} < X_0\} = \emptyset$$

or

$$\min(\{c_{ij} : x_{ij} < X_{ij}\}) > d$$

The previous case is in contradiction with the condition $B_c \neq \emptyset$. In the latter case, we have $\vec{x}'' \in B_c$. Since there is always some x_{ij} increased in the previous algorithm, we conclude that

$$\forall \vec{x}' \, \exists \vec{x}'' : \vec{x}' \in F \backslash B_c, \vec{x}'' \in B_c \Rightarrow \vec{x}'' \succ \vec{x}'$$

Thus

$$\forall \vec{x}' \, \exists \vec{x}'' : \vec{x}' \in F \backslash B_c \; \vec{x}'' \in B_c \Rightarrow A(\vec{x}'') \geq A(\vec{x}')$$

according to Property 3.1. Then we immediately obtain

$$\max\{A(\vec{x}) : \vec{x} \in B_c\} \geq \max\{A(\vec{x}) : \vec{x} \in F \backslash B_c\}$$

according to the monotonicity and thus

$$\max\{A(\vec{x}) : \vec{x} \in B_c\} = \max\{A(\vec{x}) : \vec{x} \in F\}$$

4 ALGORITHM DESIGN

4.1 *The repair algorithm*

Here we present a randomized repair algorithm to fix any solutions $\vec{x} \notin B_c$ into B_c. The algorithm randomly selects a component from the current system design and removes it. The process repeats until $\vec{x}$ becomes feasible. Then, the algorithm randomly selects a component version from the following index set

$$\{ij : x_{ij} < X_{ij}, c_{ij} < d\}$$

and add x_{ij} by one. The process repeats until $\vec{x}$ enters B_c. The pseudo code is given in Algorithm 1.

Algorithm 1. Repair algorithm

$d = C_0 - \sum_{ij} x_{ij} c_{ij}$; $n = \sum_{ij} x_{ij}$;
while $d < 0$
    $*ij = \text{randi}(1, n, \{ij : x_{ij} > 0\})$;
    $x_{ij} = x_{ij} - 1$;
    $n = n - 1$;
    $d = d + c_{ij}$;
end while;
while $d \geq \min(\{c_{ij} : x_{ij} < X_{ij}\})$
    $** \; ij = \text{rand}(ij, \{ij : x_{ij} < X_{ij}, c_{ij} \leq d\})$;
    $x_{ij} = x_{ij} + 1$;
    $d = d - c_{ij}$;
end while;

* $\text{randi}(1, n, \{ij : x_{ij} > 0\})$ uniformly samples one component from all the n components installed on the current system and stores the component version of the sampled component.

** $ij = \text{rand}(ij, \{ij : x_{ij} < X_{ij}, c_{ij} \leq d\})$ uniformly samples the component version from the version set $\{ij : x_{ij} < X_{ij}, c_{ij} \leq d\}$ and stores it.

The time complexity of Algorithm 1 is $O(n^2)$.

In GA, new individuals are generated during reproduction. In this paper, the repair algorithm is only used after crossover, to ensure the entire population is within B_c before mutation is performed.

4.2 *The local-search mutation operator*

In literature, the operators designed for local search in MSSPS RAP include $(-1, +1)$, $(-1, +\beta)$ and $(-all, +\beta)$ (Ouzineb et al. 2008, Wang & Li 2012). In short, the $(-1, +1)$ operator deletes one existing component from the current solution and then adds one component of a different version onto the changed solution. It has also been used to replace the standard mutation operator in GA by Levitin et al. (1998). The $(-1, +\beta)$ and $(-\text{all}, +\beta)$ operators, they are restricted within a selected subsystem, in which "$-$all" refers to removing all components of a selected version and β is calculated to approximately compensate the availability difference caused by the "-1" or "$-$all" operation.

In this section, we provide a novel mutation operator for max MSSPS RAP, integrating the advantage of the above mentioned local search schemes. It assigns a uniform mutation probability p_m to each subsystem. When a subsystem i mutates, it applies a $(-\alpha, +\beta)$ operator onto itself. The $(-\alpha, +\beta)$ operator removes α components from a randomly selected component version ij and, then, adds β components to a randomly selected component version ij'. Detailed steps of the mutation operator are given in Algorithm 2.

Algorithm 2. Mutation operator

Step 1. determine the subsystems that will mutate according to p_m and store the indexes of these subsystems in an index set M_i;

Step 2. build a $1-1$ mapping

$$f: M_i \to f_M = \{1,2, \ldots |M_i|\}$$

by using a randomized disorder algorithm f on $(M_i \times f_M)$;

Step 3. select element k from f_M by ascending order, then decide the index i of the selected subsystem by

$$i = f^{-1}(k);$$

Step 4. select j and j' from the candidate component version set $\{ij\}$, respectively, according to uniform distribution;

Step 5. calculate $d = C_0 - C(\vec{x})$ for the current system;

Step 6. determine α by one realization of the RV

$$X_\alpha \sim U(1, x_{ij})$$

where $U(1, x_{ij})$ is a uniform integer distribution on the range (1,2,…, x_{ij});

Step 7. determine β by the following equation:

$$\beta = \left\lfloor \frac{d + \alpha \cdot c_{ij} x_{ij}}{c_{ij'}} \right\rfloor + \begin{cases} X, \text{when } k \neq |M_i| \\ 0, \text{when } k = |M_i| \end{cases}$$

where $X \sim Bern(0.5)$ is a Bernoulli trial;

Step 8. if $k \neq |M_i|$, go to Step 3; otherwise terminate the algorithm.

Note that in Step 7., β is used to compensate the reduction of system cost caused by the "$-\alpha$" operation; the Bernoulli trial X is used to allow the system having a 50% chance to slightly violate the cost constraint and another 50% chance to stay in B_c for $k < |M_i|$. When $k = |M_i|$, $X = 0$ to ensure the system remains in B_c after mutation, since it is the last subsystem to be mutated. To ensure the index of the last subsystem to be repaired $f^{-1}(|M_i|)$ is randomly selected, we use a randomized disorder algorithm in Step 2.

4.3 *The integer-coded elitism GA with repair algorithm and modified mutation operator*

In this paper, the repair algorithm and the mutation operator are integrated into an integer-coded elitism GA (*IEGA*). In *IEGA*, each bit in a chromosome represents the number of components of a certain version; for elitism, the parents will compete with the children in a tournament selection for survival, in which only the best half of the combined population will survive. For fitness evaluation, the value of the objective function of Problem 2.1 is calculated. The individuals with larger objective function values are considered to have better fitness rankings. To avoid selecting infeasible solutions for the next generation, a penalty term with large constant coefficient is added.

In our modified version of *IEGA,* named as *r-nm-IEGA*, the repair algorithm is inserted after the crossover and the original mutation operator is replaced by the novel mutation operator. The complete algorithm is given in the following Algorithm 3.

Algorithm 3. r-nm-IEGA

Step 1. Initialization: initialize the population size P_s, the maximum generation R_m, the length of each chromosome $L_c = \sum_{i=1}^{N} n_i$, the crossover probability p_c, and the mutation probability p_m; initialize the first population by assigning an integer number to each bit in the chromosome within the range $(0,1, \ldots X_0)$, and evaluate the fitness value of the initial population;

Step 2. Parents selection: select parents from the current population by binary tournament selection;

Step 3. Crossover: select pairs of parents for crossover according to p_c and, then, perform 2-point crossover between each pair of parents;

Step 4. Repair: each chromosome is repaired back to B_c by Algorithm 1 if it is in the initial population or has undergone crossover;

Step 5. Mutation: select chromosomes in current population for mutation according to p_m and, then, run the mutation operator defined in Algorithm 2;

Step 6. Fitness evaluation: compute the fitness values of the children population;

Step 7. Selection: combine the parents and children populations; build the new population of size P_s by running a binary tournament selection on the combined population;

Step 8. Termination: The algorithm terminates if the maximum generation R_m is reached.

5 BENCHMARK TESTS

The benchmark systems tested in this paper are **P1** (Levitin et al. 1998), **P2** (Ouzineb et al. 2008 & 2011) and **P3** (Levitin et al. 1997). In literature, **P1** and **P2** have been tested only for min MSSPS RAP; **P3** is the only benchmark that has been tested for the max MSSPS RAP (Gupta & Agarwal 2006). The problem type, constraint setting and best optima found on **P1**~ **P3** are listed in Table 1.

Table 1. The best results on the benchmark problems

	Constraint	*Found optimal*	*Problem Type*
P1	$A \geq 0.99$	$C = 8.18$ $A = 0.991$	Min
P2	$A \geq 0.99$	$C = 12.794$ $A = 0.992$	Min
P3	$C \leq 16$	$C = 15.982$ $A = 0.994$	Max

For **P1** and **P2**, we can build the respective maximization problem in the following way:
1. set the cost of the current best solution as maximal system cost;
2. set the current best solution as the reference for the maximization problem.
The new formulations for **P1** and **P2** are given in Table 2.

Table 2. Maximization formulations for **P1** and **P2**

	Constraint	*Reference Availability*
P1	$C \leq 8.18$	$A = 0.991$
P2	$C \leq 12.794$	$A = 0.992$

In this paper, *r-nm-IEGA* and *IEGA* are compared on **P1**~**P3**. On **P3**, the results published by Gupta & Agarwal (2006) are used for comparison. The sizes of the population and the maximum generation of *r-nm-IEGA*, *IEGA* and the penalty-guided GA (Gupta & Agarwal 2006) are given in Table 3. To find the best settings of p_c and p_m for *IEGA* and *r-nm-IEGA* in each benchmark, each algorithm is tested on a range of parameter values: for *r-nm-IEGA* they are $p_c = 0.1,0.2,\ldots,1$ and $p_m = 0.1,0.2,\ldots,1$ and for *IEGA* they are $p_c = 0.1,0.2,\ldots,1$ and $p_m = 0.02,0.04,\ldots,0.3$.
The mutation probabilities of *r-nm-IEGA* are much larger than those of the original GA, since our mutation operator works on subsystems instead of bits. Considering the local search scheme used in our mutation operator, it is straightforward to see that the mutation probability on each bit is still relatively small. The optimal parameters for each algorithm on each problem are listed in Table 4.

Table 3. Population settings

	r-nm-IEGA	*IEGA*	*Penalty-guided GA*
P_s	20	20	50
R_m	500	500	2000

Table 4. Optimal parameter settings

(p_c, p_m)	*r-nm-IEGA*	*IEGA*
P1	(0.6,0.5)	(0.8,0.14)
P2	(0.7,0.3)	(0.7,0.1)
P3	(0.5,0.4)	(0.5,0.2)

The results for **P1** and **P2** are given in Table 5, and show that *r-nm-IEGA* reaches the reference availabilities given in Table 2 within 500 generations while *IEGA* does not. Figure 1 shows the convergence curves of the best runs and the averages of the 10 runs of *r-nm-IEGA* and *IEGA* with the optimal p_c and p_m settings. From Figure 1, we can observe that:
1. the best run of *r-nm-IEGA* reaches the reference availability within about 60 runs while *IEGA* does not for 500 runs;
2. the average convergence curve of *r-nm-IEGA* is better than the convergence curve of the best run of *IEGA*.
Thus, we can conclude that *r-nm-IEGA* outperforms *IEGA* both in efficiency and robustness.

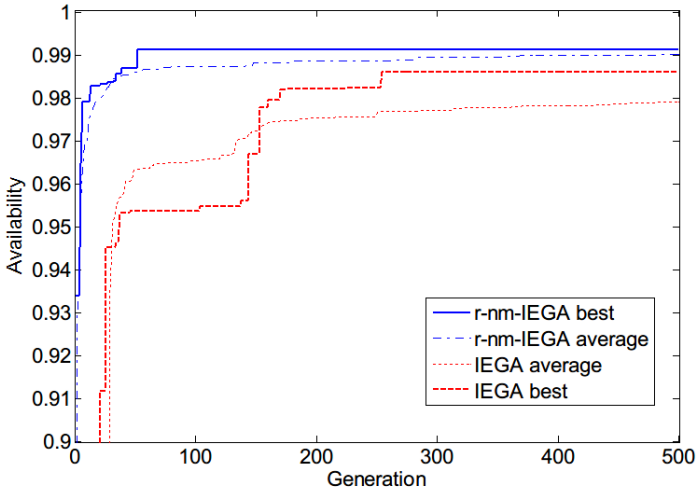

Figure 1. The convergence curves of **P1**

Table 5. Best optima obtained by the GAs

Best optima	*r-nm-IEGA*	*IEGA*	*Penalty-guided GA*
P1	0.991	0.986	/
P2	0.992	0.982	/
P3	0.997	0.986	0.994

Table 5 also shows that *r-nm-IEGA* obtains an optimal solution for **P3** that is better than the one published in (Gupta & Agarwal 2006). The details about the solutions are presented in Table 6.

Table 6. Best optimal solutions obtained by *r-nm-IEGA* and penalty guided GA (Gupta & Agarwal 2006) for **P3**

Sub	*r-nm-IEGA*	*Penalty-guided GA*
1	2(2)	5(1) 6(1) 7(2)
2	5(6)	3(2)
3	2(2) 3(1)	2(2) 3(1)
4	7(3)	5(2) 7(1)
5	4(3)	3(2) 4(1)

6 CONCLUSION

The mathematical properties of the max MSSPS RAP allow the optimal solution and approximate optimal solutions to be found within a subset of the boundary of the feasible region, which ensures the effectiveness of using a repair algorithm for the search space reduction. Considering the series-parallel structure of MSSPSs and the existing local search schemes in literature for min MSSPS RAP, a new mutation operator is designed for max MSSPS, which inherits the form of $(-x, +y)$ and retain the population inside the reduced search space after mutation. In benchmark tests our modified *r-nm-IEGA* is compared with the original GA, i.e. the *IEGA* and the published method. The results indicate the superiority of the proposed *r-nm-IEGA*.

REFERENCES

Gupta, R. & Agarwal, M. 2006. Penalty guided genetic search for redundancy optimization in multi-state series-parallel power system. Journal of Combinatorial Optimization 12(3): 257-277.

Kuo, W. & Rui, W. 2007. Recent advances in optimal reliability allocation. Computational Intelligence in Reliability Engineering: 1-36. Springer: Berlin.

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

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. et al. 1998. Redundancy optimization for series-parallel multi-state systems. Reliability, IEEE Transactions on, 47(2): 165-172.

Lisnianski, A. et al. 1996. Power system structure optimization subject to reliability constraints. Electric Power Systems Research 39(2): 145-152.

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. 2011. A heuristic method for non-homogeneous redundancy optimization of series-parallel multi-state systems. Journal of Heuristics, 17(1): 1-22.

Wang, Y. & Li, L. 2012. Heterogeneous redundancy allocation for series-parallel multi-state systems using hybrid particle swarm optimization and local search. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 42(2): 464-474.

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