Chapter 16
Cost Optimization and High Available Heterogeneous Series-Parallel Redundant System Design Using Genetic Algorithms

Walid Chaaban, Michael Schwarz, and Josef Börcsök

Abstract Heterogeneous redundant series-parallel systems allow the mixing of components within the same subsystem. This diversity feature may improve the overall characteristics of the system compared with the homogeneous case in term of less susceptibility against so called common-cause failures and reduced cost. That means they guarantee longer availability and are quite suitable for systems that are designed to perform continuous processes. But the main challenging task is to determine the optimal design that corresponds to the minimal investment costs and satisfies the predefined constraints. This kind of combinatorial optimization tasks is perfectly solved using heuristic methods, since those approaches showed stability, powerfulness, and computing effectiveness in solving such matters. This task is more complex than the homogeneous case since the search space is getting larger due to the fact that every component available and that can be deployed in a subsystem has to be taken into account. This fact leads definitely to greater chromosome length and makes the search more time consuming. The algorithm has been implemented in Matlab and three different existing models (Levitin, Lisnianski, and Ouzineb) have been considered for a comparison with the homogeneous case and for validation purposes.

Keywords Common cause failure (CCF) • Genetic algorithms • Heterogeneous series-parallel systems • Redundancy allocation problem (RAP) • Universal moment generating function (UMGF)

W. Chaaban (✉) • M. Schwarz • J. Börcsök
Department of Computer Architecture and System Programming, University of Kassel, Willhelmshöher Allee. 71, Kassel 34121, Germany
e-mail: walid.chaaban@uni-kassel.de; m.schwarz@uni-kassel.de; j.boercsoek@uni-kassel.de

16.1 Introduction

Heuristic search methods represent powerful and effective means in solving combinatorial optimization problems since they do not require any additional information compared with the classical optimization methods and they accelerate the search towards objective or convergence through their parallel performed exploration and exploitation of the search space.

One well known combinatorial optimization task solved using such heuristic approaches is the Redundancy Allocation Problem (RAP) also referred to as Redundancy Optimization Problem (ROP) which consists of determining best series-parallel system designs in terms of redundancy depth on different subsystems level corresponding to minimal investment costs and that satisfies at the same time the predefined constraints and system design requirement specifications like availability, weight, volume and etc.

The Redundancy Allocation Problem (RAP) or Redundancy Optimization Problem (ROP) [1] is a single objective optimization and can often be encountered in many applications areas of the safety engineering world like electrical power systems and in the consumer electronic industry where system designs are mostly assembled using standard certified component types with different characteristics, e.g., reliability, availability, nominal performance, cost, etc. This matter has been intensively studied over last two decades and has been classified as a complex nonlinear integer programming combinatorial problem, where deterministic or conventional mathematical optimization approaches become ineffective by means of computational effort and quality of solution [1].

Using heuristic and metaheuristic search methods, e.g. Genetic Algorithms (GAs), Tabu Search (TS), Simulated Annealing, etc., in solving such kind of combinatorial optimization problems aims to determine an optimal or near optimal, also called pseudo-optimal solution, to the proposed RAP, i.e. to find the best or at least one acceptable solution that satisfies the constraint(s). However, these approaches have shown instead how powerful and effective they are in finding high qualitative solutions for the addressed kind of problems, especially when the search space corresponding to the problem becomes too large and where conventional classical optimization methods become ineffective and useless. This kind of problems was first introduced by Ushakov [1] and has been further analyzed by Levitin and Lisnianski et al. [2–4], Ouzineb [5–7] and many others.

This paper deals with the cost optimization of heterogeneous structured series assembled systems, where mixing of components or usage of non-identical components within the same subsystem is allowed. This feature, compared with the homogeneous case, includes more complexity to the task because the corresponding search or solution space becomes larger, since every component available on the market has to be taken into account. It represents a single objective optimization (cost function) subject to one constraint which is the availability of the system.

The remainder of the paper is organized as follows. Section 16.2 gives a short introduction into heterogeneous series-parallel multi-states configurations and a brief overview on the advantages obtained through mixing of components in addition to a short comparison with homogeneous systems. Section 16.3 shortly discusses genetic algorithms and its different operators while chromosomal encoding and random generating of solution candidates are implemented in Sect. 16.4. In Sect. 16.5 a detailed formulation of the optimization problem, which is solved using heuristic traditional GA genetic techniques, is presented. Section 16.6 deals with the Universal Moment Generating Function (UMGF), also called the *Ushakov*-transform, which represents the function used for the determination and evaluation of the availability of the different redundant structures. Section 16.7 reports different numerical and experimental results, evaluations and graphical representations obtained by the implemented GA algorithm for different analyzed models which will be compared with previously published evaluations in terms of efficiency, solution quality and accuracy in addition to algorithm computation speed and convergence time. Finally concluding remarks are resumed in Sect. 16.8.

16.2 Homogeneous vs. Heterogeneous Series-Parallel Configurations

In order to improve or to increase the system's reliability and provide longer operation time, safety system designers may introduce different parallel technologies into a system also called redundancy [8]. Including homogeneous components redundancy is a great and effective technique to achieve a desired level of reliability in binary state systems or to increase the availability of multi-state systems. Reliability analysis have shown that the availability of homogeneous redundant structures or systems is extremely affected by common cause failure (CCF) that cannot be ignored since the CCF is the simultaneous failure of all components of the same type due to a common cause (CC), which leads in homogeneous redundant structures definitely to the failure of complete subsystems consisting of identical components causing herewith a total system failure. Common cause events may arise from environmental loads (humidity, temperature, vibration, shock, etc.), errors in maintenance and system design flaws [9]. In order to partly overcome this kind of facing problems and avoid total system failure subject to CCF heterogeneous redundancy is used.

The main concept of heterogeneous or non-homogeneous structures consists of the mixture of non-identical components within the same subsystem. That means that all non-identical components with the same functionality available on the market and which can be deployed in a redundant manner within the same subsystem have to be taken into account in this case, the fact that would definitely enlarge the size of the search space of feasible solutions and increase the exploration and hence the convergence time towards acceptable solutions.

The main advantages and benefits of components mixing lie in the improvement of the availability of the whole system and reducing the effect of common cause failure in addition of introducing flexibility and diversification into redundant system design through the allowed multiple component choice.

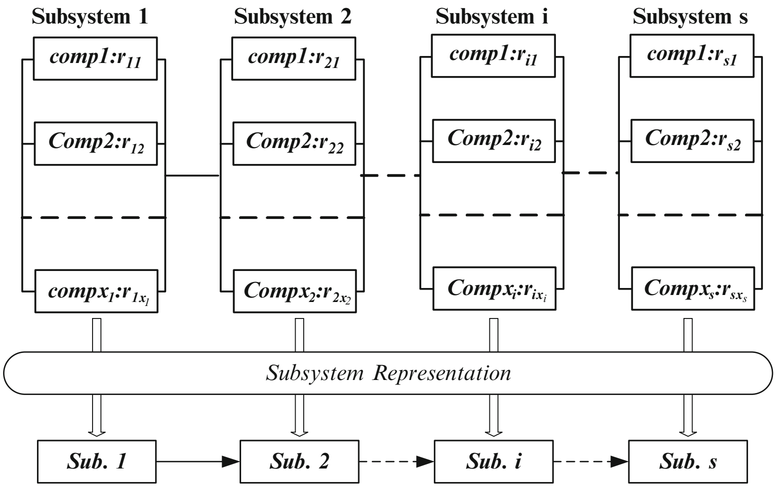

Fig. 16.1 Heterogeneous series-parallel configuration consisting of s-nodes or subsystems

Figure 16.1 represents a heterogeneous series-parallel multi-state system consisting of s subsystems, which are connected serially.

In Fig. 16.1 r_{ij} represents the reliability of a component of version j within the subsystem i. In the case of a homogeneous configuration the reliabilities of all components within the same subsystem are the same since subsystems consists of identical components, i.e. all r_{ij}'s are equal for the same i, which must not be the case in heterogeneous systems since they mostly deal with non-identical components on the same stage.

For a brief explanation, in addition to a short mathematical computation that shows why series-parallel configurations are more suitable and studied than parallel-series configurations, the reader should refer to [1, 10]. This is due to the fact that the overall reliability or availability of a system in a series-parallel configuration is better than the corresponding parallel-series configuration using the same set of components.

16.3 Genetic Algorithms

Genetic algorithms (GAs) are biologically inspired metaheuristic search and optimization routines that mimic the act of self-evolution concept of natural species that has been first laid by Charles Darwin. Nowadays, they are frequently used in many engineering and mathematical fields like optimization, self-adaptiveness, artificial intelligence, machine learning, etc. As computational efforts and speeds have been increasingly improved over the last decade, GAs have been expanded to cover a wide variety of applications including numerical and combinatorial optimization tasks in engineering like the one discussed in the recent work. For further fundamentals and detailed information on genetic algorithms, the reader should refer to [11, 12].

GAs represent iterative self-adaptive stochastic techniques based on the concept of randomness. They mimic the process of the natural evolution of species. GAs have become very popular and widely used over the last decade and are very well suited as universal or common techniques for solving combinatorial optimization problems, e.g. multimodal functions (many peaks and local optima), the very well-known TSP (Travelling Salesman Problem) and redundancy allocation problems like the one discussed in this paper and many other matters [13].

GAs differ from normal optimization and search methods in four fundamental different ways [12]:

- The first difference is that GAs require an encoding of the parameter set in so called solution candidates, also called chromosomes according to the biological genetic terminology.
- Another difference is that GA starts the search from a start (initial) population and not from a single point like classical deterministic algorithms.
- GAs use information provided directly by the objective function and do not require any additional information or auxiliary knowledge like derivatives, gradients, etc.
- GAs apply probabilistic transition rules or operators (crossover, mutation) and not deterministic ones.

As mentioned before the search procedure starts from a random generated population of chromosomes that are encoded according to the addressed problem (binary, integer, decimal, etc.) conducting herewith a simultaneous search in many areas of the feasible solution space at once. The encoding of solutions constitutes the most difficult and challenging task of GAs and the evolving procedure from one population to the next is referred to as generation.

After each generation the new generated solutions are decoded and evaluated in terms of fitness with the help of the objective function. The fitness value of a chromosome represents a measure for its quality (fitness) and represents the decision maker of the selection operator since the probability of selection of chromosomes is proportional to its corresponding fitness value. A general overview of the genetic cycle is given in Fig. 16.2.

The genetic run process terminates when at least one of its predefined termination criterions is met, e.g., when the predefined maximum number of generations or repetitions N_{rep} or a specific number of successive runs without any solution's improvement is reached, or for example when the current solution satisfies the predefined requirement specifications or constraints.

Three main operators, also called genetic operators, will be executed during one genetic cycle, hence the selection, crossover or recombination and the mutation operator. These operators are shortly discussed in the following subsections.

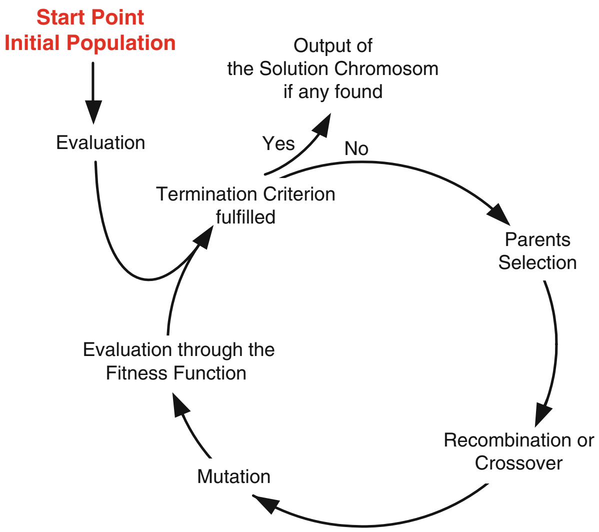

Fig. 16.2 General overview of the genetic process

16.3.1 Selection or Reproduction: Operator

Outgoing from a start or an initial population of different solution candidates the selection operator is used to randomly select or choose individuals or chromosomes which will reproduce and help building the next population during the genetic cycle. This operator represents an artificial version of natural selection, the Darwinian principle of the survival of the fittest, which drives the evolution towards optimization. Since the selection probability is proportional to relative fitness, chromosomes with higher fitness have better chance or higher probability to survive into next population, while chromosomes with bad or lower fitness will die off. This phenomenon will improve the population's average fitness from one population to the next.

There are many selection methods, some of them are listed in the following [14]

- Roulette Wheel selection,
- Tournament selection,
- Rank selection,
- etc.

16.3.2 Recombination or Crossover: Operator

Whereas the selection operator determines which chromosomes of the recent population are going to reproduce, the crossover operator performs jumps between the different solution subspaces enabling the exploration of new areas of the solution space and avoiding herewith premature convergence in addition to the exchange of some basic characteristics or genetic materials and inheriting these properties to the offsprings which will join next populations. The crossover occurs with a predefined crossover rate or probability of crossover p_c. There are many crossover techniques used in genetic algorithms [1, 14, 15] like the one-point crossover, two-point crossover, uniform and half uniform crossover and many other crossover techniques.

In the following the one-point crossover operator is shortly discussed. For this purpose two parent chromosomes *Parent1* and *Parent2* are selected. Afterwards a pseudorandom real number is generated in [0; 1]. If and only if the generated number is less or equal p_c both parents will undergo crossover; otherwise they will have to be recopied in the population and wait to undergo the next operator, hence the mutation operator.

In case 2 chromosomes undergo crossover, a crossover point or position depending on the length of the chromosome is randomly selected on both selected parent chromosomes. This step is done by a pseudorandom integer number generator that generates numbers in the interval $[1; l_c - 1]$, where l_c is the length of the vector representing the chromosome. All data beyond that point in either chromosome will be swapped between the two parent organisms which will result in two new individuals called offsprings or children chromosomes. The one-point crossover technique is depicted in Fig. 16.3.

Figure 16.4 shows a pseudo code which resumes the crossover procedure.

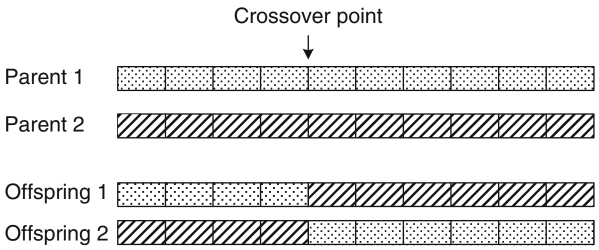

Fig. 16.3 One-point crossover technique

```
% Crossover(Parent1,Parent2,$p_c$)
if (rand(1)<=$p_c$)
     CrossPos=randi($l_c$);
     [child1,child2]=crossover(Parent1,Parent2,CrossPos)
end
```

Fig. 16.4 Pseudocode—one-point crossover

Chromosom before mutation: 1 0 **0** 0 1 0 1 0 1 1 **1** 1 1

Chromosom after mutation : 1 0 **1** 0 1 0 1 0 1 1 **0** 1 1

Fig. 16.5 Mutation of a binary encoded chromosome

Fig. 16.6 Pseudocode—mutation operator

```
% Mutation(Chromosom,$p_m$)
for i=1:length(chromosome)
    if (rand(1)<=$p_m$)
      chromosome(i)=mutate(chromosome(i),i)
    end
end
```

16.3.3 Mutation Operator

After crossovering parent chromosomes the resulting offsprings or children undergo mutation with a low mutation rate or probability p_m. The mutation operator introduces diversity into the GA algorithm and inserts small disturbance into the properties (genes) of the proposed solutions avoiding herewith premature convergence into local maxima. Mutation also helps recovering loss that might have been caused by crossover. After the mutation process has been accomplished, the new resulting mutated chromosomes constitute the new next population. The mutation of a binary encoded chromosome consists of inverting the randomly selected bit or position like depicted in Fig. 16.5, while a pseudo code which represents the mutation routine is shown in Fig. 16.6.

16.4 Chromosomal Encoding and Random Generating of Solution Candidates

Genetic algorithms are population based combinatorial optimization approaches where populations consist of a predefined number of solution candidates, also called chromosomes or individuals. These candidates represent vectors of encoded information, referred to as genetic materials, which will be decoded using the fitness or objective function in order to find the optimal (minimum or maximum) or near optimal solution of the addressed problem, whereas the recent approaches for solving the RAP problem are based on the Universal Moment Generating Function (UMGF) for estimating the availability of multi-state systems [2, 5–7, 16]. The encoding of chromosomes represents one of the major challenges faced in the context of genetic computing.

With regard to such problems that are dealt with in this paper, i.e., in the case of heterogeneous redundant structures the chromosome length corresponding to a system is definitely longer than chromosome corresponding to the homogeneous case since each component version or type available on the market that may be deployed in a subsystem has to be taken into account, whereas in homogeneous systems, the chromosome length is equal to twice the number of subsystems, since only one component type or version is allowed on each stage. The chromosomes are integer encoded and each element x_{ij} of the chromosome vector corresponds to the number of components of version j used in subsystem i. The chromosome dimension or length l_c is given therefore through:

$$l_c = \sum_{i=1}^{s} J_i . \tag{16.1}$$

where s is the total number of subsystems or stages and J_i the total number of component versions available on the market that can be used on stage or in subsystem i. For more clarification and to gain a deeper insight it is important to mention at this point that two components of different versions or non-identical components connected in parallel are supposed to perform the same task or function. The difference lies in the technical data (reliability/availability, nominal performance and etc.) in addition to purchasing costs.

For example let us consider a system consisting totally of four subsystems with the following version vector $m = [4\ 6\ 8\ 5]$ representing the number of components available on the market for each subsystem. The chromosome length results in this case as the sum of all elements of the version vector and would give according to Eq. (16.1) a total chromosome length of $4 + 6 + 8 + 5 = 23$ and the chromosome encoding or vector X will look like in the following:

$$X = ([x_{11} \ldots x_{14}], [x_{21} \ldots x_{26}], [x_{31} \ldots x_{38}], [x_{41} \ldots x_{45}]) . \tag{16.2}$$

where x_{ij} denotes as mentioned previously the number of components of type j used in subsystem i.

For generating chromosomes or solution candidates according to the addressed optimization problem, discussed in this paper, a pseudo random number generator is used. Since events should happen at random but some events or numbers within the chromosomal encoding should have a higher probability of occurrence or happening than others, e.g. zeros which means that no components of this kind are used, a weighted pseudo random number generator is used.

As mentioned above, the step of generating numbers with predefined probabilities of occurrence in addition to the limitation of the maximum number of totally used components within each subsystem would reduce the search hypervolume. This would increase the computation speed drastically towards convergence and make the algorithm more efficient and time consuming.

16.5 Cost Optimization and Redundancy Allocation: Task Formulation

The cost optimization problem of heterogeneous series-parallel redundant systems discussed in this work deals with the determination of optimal redundant designs and the level of redundancy (redundancy allocation) to use in each subsystem that corresponds to the minimal total purchasing costs of the system and which satisfies at the same time the predefined availability constraints. This kind of optimization gives a rise to safety vs. economics conflicts resumed in the following two points [17]:

Choice of components: choosing high reliable components guarantees high system availability but may be largely non-economic due to high purchase prices; whereas choosing less reliable components for lower costs on one hand may decrease the availability of the system and increase drastically the accident costs on the other hand.

Choice of redundancy configuration: choosing highly redundant configurations increases definitely the reliability and availability of the system and is accompanied at the same time with higher purchase costs caused by additional equipment units required to improve individual subsystems reliabilities.

The previously described aspects of safety system design call for compromise choices which optimize system operation in view of recommended safety and longer operation time or budget constraint. As mentioned before this paper deals with customizing a GA for budgetary optimization and redundancy determination of multi state systems under a given availability constraint. This problem is considered as a single objective optimization and can be mathematically formulated as to minimize the cost function $C_{sys}(X)$ (objective function) of the whole system given by [1, 5–7, 16]:

$$C_{sys}(X) = \sum_{i=1}^{s} \sum_{j=1}^{m_i} c_{ij} x_{ij}. \tag{16.3}$$

where c_{ij} being the cost of component of type j in subsystem i and x_{ij} the number of components of type j used in subsystem i. m_i is the number of component choices available on the market which may be deployed in subsystem i. The (cost) objective function represented in Eq. (16.3) results over the sum of the purchasing costs of all components used in system that should meet at the same time the system specified availability constraint, which implies that the total availability of the system $A_{sys}(X)$ must satisfy a minimum level of availability required A_0 (inequality or availability constraints)

$$A_{sys}(X) \geq A_0. \tag{16.4}$$

Based on the UMGF or the *Ushakov*-transform, the total availability of the system $A_{sys}(X)$ is estimated as a function of system structure, performance and availability characteristics of its constituting components.

For a detailed overview of the UMGF in computing the availability of series-parallel systems the reader should refer to [2, 3, 5–7, 10, 17, 18].

16.6 Universal Moment Generating Function

The UMGF, also referred to as *Ushakov*-transform according to *I. Ushakov* (mid 1980s) or *u*-function, is a polynomial representation of the different states corresponding to a component or a system. For a great understanding of the mathematical fundamentals of the UMGF the reader should refer to [17]. For example, the *u*-transform $u_j(z)$ of a component or a random variable j having M different discrete states is given by

$$u_j(z) = \sum_{m=1}^{M} p_m z^{W_m}. \tag{16.5}$$

In Eq. (16.5) p_m is the probability that the nominal performance of the component or value of the discrete random variable j at state m equal to W_m.

Since in the recent study it is assumed that the used components are binary state, i.e., have two particular states (perfect working or complete failing), the *Ushakov*-polynomial representation of a binary state component j is given by

$$u_j(z) = \sum_{m=1}^{2} p_m z^{W_m} = \left(1 - A_j\right) z^0 + A_j z^{W_j} = \left(1 - A_j\right) + A_j z^{W_j}. \tag{16.6}$$

In Eq. (16.6) A_j represents the probability that the component is available (perfect functioning) and delivers a nominal performance of W_j ($\Pr[W_m = W_j] = A_j$) while $(1 - A_j)$ represents the probability of unavailability or failing (system not available, i.e., $\Pr[W_m = 0] = 1 - A_j$). The 0 power factor in the failing state results from the absence of delivered performance in this state.

In order to determine the *u*-function of an entire series-parallel system for availability computation purposes, two different basic operators will be respectively applied [5, 6, 10, 17, 18]. These two operators also called composition operators applied respectively [17] will be implemented separately in the following subsections.

16.6.1 *Γ-Operator: Ushakov Transform of Parallel Configurations*

The *Γ*-composition operator is used to determine the *u*-function of parallel systems. Suppose a system consisting of x_i parallel connected components, the corresponding *u*-function is given by the following equation

$$u_{parallel}(z) = \Gamma\left(u_1(z), u_2(z), \ldots, u_{x_i}(z)\right). \tag{16.7}$$

where the total performance or the structure function $f(W_1,W_2,\ldots,W_{xi})$ is given by the sum of the performances or capacities of the individual components or elements as described in the following equation

$$f\left(W_1, W_2, \ldots, W_{x_i}\right) = \sum_{i=1}^{xi} W_i. \tag{16.8}$$

For a pair of parallel connected components with the corresponding u-functions $u_1(z)$ and $u_2(z)$ given according to Eq. (16.5) the resulting *u*-function $u_{parallel}(z)$ of the entire parallel system is given by

$$\begin{aligned} u_{parallel}(z) &= \Gamma\left(U_1(z), U_2(z)\right) \\ &= \sum_{i=1}^{n} P_{1i} z^{W_{1i}} \sum_{j=1}^{m} P_{2j} z^{W_{2j}} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} P_{1i} P_{2j} z^{W_{1i}+W_{2j}}. \end{aligned} \tag{16.9}$$

In Eq. (16.9), *n* and *m* represent the number of states of the components 1 and 2. W_{1i} and W_{2j} are respectively the nominal performances of the components 1 and 2 at states *i* and *j*, which occur with the respective probabilities P_{1i} and P_{2j} ($i = 1 \ldots n$ and $j = 1 \ldots m$). All this means that the *Γ*-operator is nothing else than the polynomial product of the individual *u*-functions corresponding to all parallel connected components and therefore Eq. (16.9) can be represented as

$$u_{parallel}(z) = \prod_{e=1}^{x_i} u_e(z). \tag{16.10}$$

For a subsystem *i* within a series-parallel configuration consisting of x_i different parallel connected binary state components, whose UMGF representation is given by Eq. (16.6), the *u*-function according to Eq. (16.10) is written in the form

$$u_{parallel}(z) = \prod_{e=1}^{x_i} \left[\left(1 - A_{ij}\right) + A_{ij} z^{W_{ij}}\right]. \tag{16.11}$$

j represents the index corresponding to the version of the component used in case of non-homogeneity. Suppose that all parallel connected components are identical, Eq. (16.11) is rewritten as

$$u_{parallel}(z) = \left[\left(1 - A_{ij}\right) + A_{ij} z^{W_{ij}}\right]^{x_i}. \tag{16.12}$$

Using the binomial theorem the power representation of Eq. (16.12) can be expanded in a sum of the form

$$\begin{aligned} u_{parallel}(z) &= \sum_{k=0}^{x_i} \binom{x_i}{k} \left(A_{ij} z^{W_{ij}}\right)^k \left(1 - A_{ij}\right)^{x_i - k} \\ &= \sum_{k=0}^{x_i} \binom{x_i}{k} A_{ij}^k \left(1 - A_{ij}\right)^{x_i - k} z^{k W_{ij}} \\ &= \sum_{k=0}^{x_i} \alpha_{ik} z^{k W_{ij}} \end{aligned} \tag{16.13}$$

with

$$\alpha_{ik} = binm\left(k, A_{ij}, x_i\right) = \binom{x_i}{k} A_{ij}^k \left(1 - A_{ij}\right)^{x_i - k} \tag{16.14}$$

and where the binomial coefficients are given by

$$\binom{x_i}{k} = \frac{x_i!}{k!\,(x_i - k)!}. \tag{16.15}$$

That means that the u-function or the polynomial z-representation of each parallel subsystem consisting of x_i non-identical components (heterogeneous case) can be computed according to Eq. (16.11). If all x_i or a specific number x_n of components are identical, some simplification can be made using the binomial theorem according to Eq. (16.13).

16.6.2 η-Operator: Ushakov Transform of Series Configurations

In order to determine the u-function of a system consisting of s elements or components connected in series, the η-operator is applied according to

$$u_{series}(z) = \eta\left(u_1(z), u_2(z), \ldots, u_s(z)\right). \tag{16.16}$$

In the case of series connected components, the element with the minimal or least performance becomes the bottleneck of the system. This system is the decision maker about the total system performance or productivity. Therefore the structure or the performance function f is given by

$$f\left(W_1, W_2, \ldots, W_s\right) = \min\left(W_1, W_2, \ldots, W_s\right). \tag{16.17}$$

For a pair of components $u_1(z)$ and $u_2(z)$ connected in series and according to Eq. (16.17) the resulting u-function of the system $u_{series}(z)$ is given by

$$\begin{aligned} u_{series}(z) &= \eta\left(u_1(z), u_2(z)\right) \\ &= \eta\left(\sum_{i=1}^{n} P_{1i} z^{W_{1i}}, \sum_{j=1}^{m} P_{2j} z^{W_{2j}}\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} P_{1i} P_{2j} z^{f\left(W_{1i}, W_{2j}\right)}. \end{aligned} \tag{16.18}$$

Replacing the structure function according through Eq. (16.17), the u-function of two series connected components will be given by

$$u_{series}(z) = \sum_{i=1}^{n}\sum_{j=1}^{m} P_{1i} P_{2j} z^{\min\left(W_{1i}, W_{2j}\right)}. \tag{16.19}$$

16.6.3 Ushakov Transform of Series-Parallel Systems: (Γ, η)

As mentioned previously, in order to determine the u-function of the entire series-parallel MSS both composition operators Γ and η have to be performed respectively like depicted in Fig. 16.7.

In Fig. 16.7 the individual s parallel systems consisting of non-identical binary state components will be replaced by single elements having multi-states u-functions ($u_1(z)$ … $u_s(z)$) determined by the Γ-composition operator according to Eq. (16.11). Afterwards the η-composition operator will be applied on the resulting system consisting of the s-multi-states components connected in series which leads to the u-function $u_{sys}(z)$ of the entire system that would be computed with the help of Eq. (16.16) like in the following

$$\begin{aligned} u_{sys}(z) &= \eta\left(u_1(z), u_2(z), \ldots, u_s(z)\right) \\ &= \eta\left(\sum_{k=0}^{x_1} \alpha_{1k} z^{kW_{1j}}, \sum_{k=0}^{x_2} \alpha_{2k} z^{kW_{2j}}, \ldots, \sum_{k=0}^{x_s} \alpha_{sk} z^{kW_{sj}}\right) \\ &= \sum_{m=0}^{M} \delta_m z^{W_m}. \end{aligned} \tag{16.20}$$

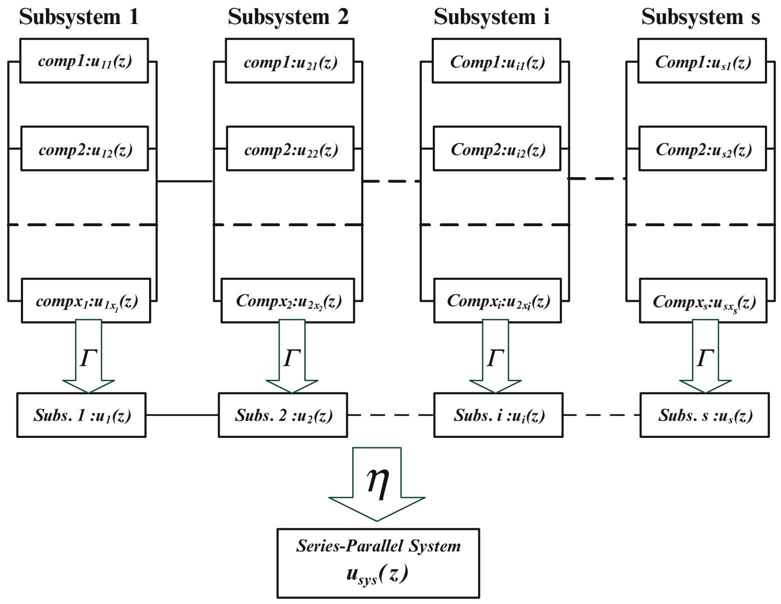

Fig. 16.7 Determination of the $u(z)$-function for the entire series-parallel system

δ_m and W_m are real numbers determined according to Eq. (16.19). The evaluation of the probability that the entire system satisfies a specific level of performance W_0 is given by the sum over all coefficients δ_m that correspond to a nominal performance W_m greater or equal W_0. The resulting sum represents the availability $A_{sys}(X)$ corresponding to the series-parallel system, whose design is represented by X and is given by

$$A_{sys}\left(W_0\right) = \Pr\left(W_m \geq W_0\right) = \sum_{W_m \geq W_0} \delta_m. \tag{16.21}$$

Given K different demand levels represented by W_0^k where $k = 1 \ldots K$, which should be satisfied over different operation intervals T_k the total availability $A_{sys}(X)$ of the system is obtained by the sum of the instantaneous availabilities corresponding to the different demand levels divided by the total operation or mission time and is given by

$$\begin{aligned} A_{sys}(X) &= \tfrac{1}{T}\sum_{k=1}^{K}\sum_{W_m \geq W_0^k} \delta_m T_k \\ &= \frac{1}{\sum_{k=1}^{K} T_k}\sum_{k=1}^{K}\sum_{W_m \geq W_0^k} \delta_m T_k. \end{aligned} \tag{16.22}$$

16.7 Tuning Parameters and Experimental Results: Validation

The simple genetic algorithm with some modifications in the context of its operators was used in this work. The algorithm has been implemented in Matlab which provides uniform pseudorandom number generators and powerful matrix and vector operations and allows great visualization and graphical representation. The three different models, which have been analyzed in the homogeneous case in a previous publication [10, 18], have been treated again in the heterogeneous case in order to show the effect of mixing of components on investment cost reduction and hence getting safer systems subject to CCFs for lower cost than the homogeneous case and for same given constraint factors. The used components are assumed to be binary state (perfect working or totally failing). The models and data, as mentioned previously, have been taken from [5–7], are listed below:

- Lev4_4_6_3
- Lev5_5_9_4
- Ouz6_4_11_4

For a brief understanding of the decoding of the denotation of the individual models it is referred to Ouzineb in [5–7]. The purchasing price, reliability and nominal performance capacity for the components corresponding to the upper listed systems are supposed to be known and can be retrieved from a list with technical data (excel sheets).

The algorithm starts by retrieving the data of the analyzed problem from the appropriate sheet and by random generating a so called initial population of size Pop_{size} that has been set to 100 chromosomes. The integer encoded chromosomes constituting the initial population have been generated in such a way that generated solutions that do not satisfy the given availability constraint are rejected and replaced by new acceptable ones in order to get a high qualitative start population. The constituting individuals or chromosomes have been created using a weighted pseudo random number generator which generates numbers between zero and the total number of components allowed in each subsystem, which have been set to ten. The probability of occurrence of 0 has been varied between 0.7 and 0.9 during the random chromosome generation process depending on the length of the chromosome corresponding to the analyzed problem. This limitation to the number of components allowed within a subsystem in addition to the rejection of non-feasible solutions. Furthermore a weighted generation of chromosome would limit the area that will be explored within the search space and should accelerate the search procedure towards convergence.

After evaluating and ranking populations (cost and availability estimation) chromosomes are selected to mate, recombine and finally mutate in order to build new offsprings that complete the next population of size Pop_{size}. This genetic procedure repeats until the predefined maximal number of generations N_{rep} is reached (Termination criterion). After completing each population through crossover and

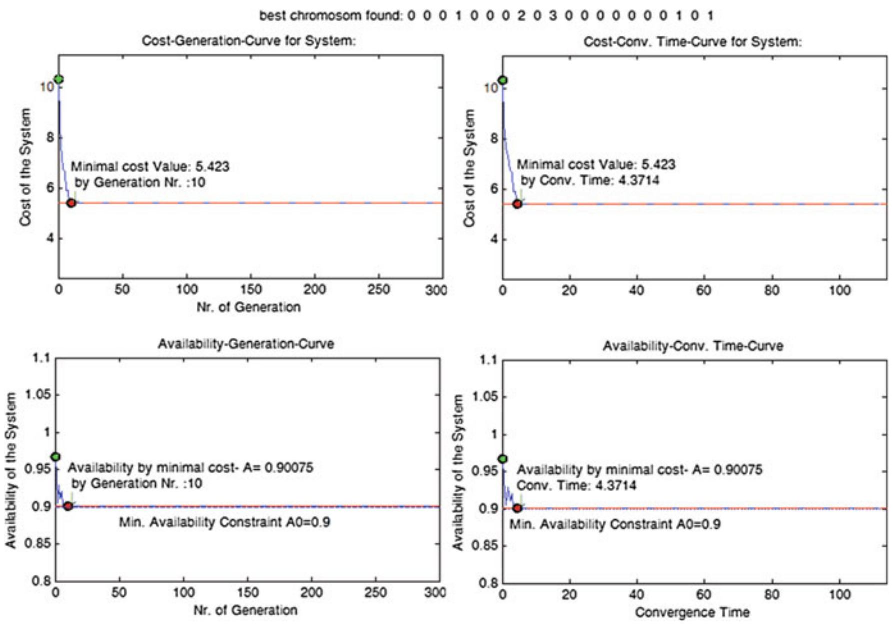

Fig. 16.8 Results of the genetic algorithm run of the heterogeneous case (Levitin—model containing four subsystems, availability constraint $A_0 = 0.900$ and 300 generations). The cost-generation and availability-generation curves dependencies are depicted on the left-hand side. The cost-time and availability-time dependencies are depicted on the right-hand side

mutation the population will be checked for multiplicity before evaluation by term of fitness function and new chromosomes would be generated to replace the chromosomes that appear more than once and have therefore been removed. This procedure of inserting new chromosomes to the population is compared to the act of inserting new genetic materials and may lead to new search areas that have not been explored or searched before and may accelerate the convergence speed. Figures 16.8 and 16.9 show the results of one run of the GA over the Lev4-(4/6)-3 model (heterogeneous case) data by predefined availability constraints of $A_0 = 0.900$ and $A_0 = 0.960$, and Fig. 16.10 shows the run of the GA over the Ouz6_(4/11)_4 by an availability constraint of $A_0 = 0.99$. The different plots show the evolution progress outgoing from the random initial population up to the predefined maximum number of generations. The best result (Cost—upper plots and Availability—lower plots), received after each genetic cycle, is depicted.

The time needed to find the best solution (convergence time) depends on the quality of the start population and on how the selected fittest chromosomes evolve throughout crossover and mutation.

On the left-hand side of Figs. 16.8, 16.9 and 16.10 (heterogeneous case) the best solution found (Top: cost value, bottom: availability value for found cost) during each generation is plotted against generation number whereas the same plots are

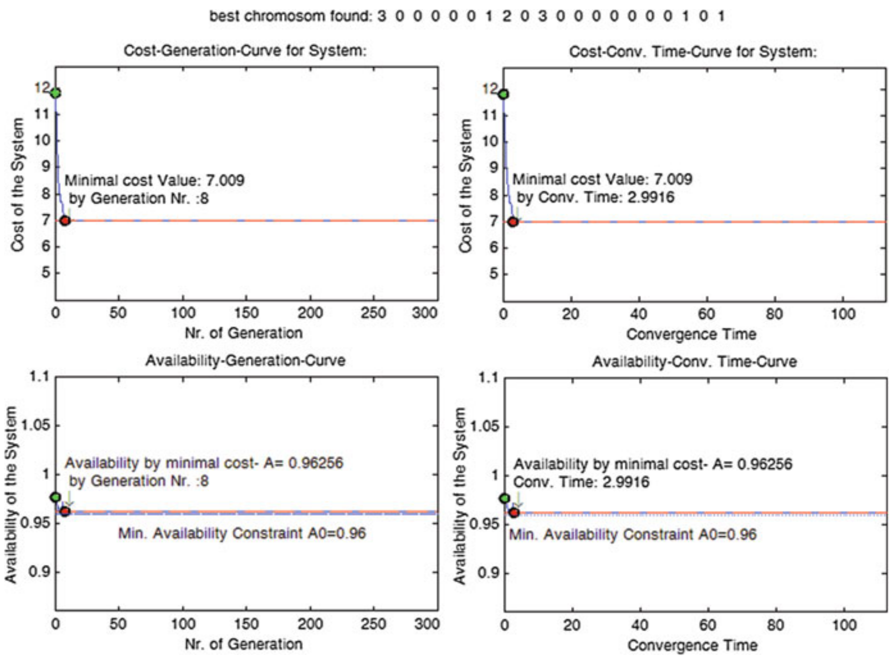

Fig. 16.9 Results of the genetic algorithm run of the heterogeneous case (Levitin—model containing four subsystems, availability constraint $A_0 = 0.960$ and 300 generations). The cost-generation and availability-generation curves dependencies are depicted on the left-hand side. The cost-time and availability-time dependencies are depicted on the right-hand side

represented on the right hand side against algorithm processing time. On the head of each plot the best chromosome or system design corresponding to the optimal (minimal) found cost subject to the given availability constraint is represented. In the context of the plots the generation number and convergence time are reported for which the best result has been identified.

Figures 16.11, 16.12 and 16.13 represent the homogeneous case of the heterogeneous problems analyzed successively in Figs. 16.8, 16.9, and 16.10. These figures have been included in order to show that through mixing of components lower or better system costs (Lev4-(4/6)-3—Model) can be reached in comparison to the homogeneous case subject to the same availability constraint. In the analyzed Ouz6_(4/11)_4 model no better cost results have been achieved but at least the same results as in the homogeneous case have been reached. One additional reason is to show that with the GA approach analyzed in this paper it was also possible to get the same results got with the hybridized GA + TS algorithm implemented in [5, 7]. This fact shows the effectiveness and accuracy of the GA approach discussed in this work since with the GA implemented in [6, 7] different results have been achieved.

The best test results got within 15 successive runs of the genetic algorithm over the different models mentioned previously are shown in Table 16.1. Computing and convergence time for best achieved results are also included in the same table and

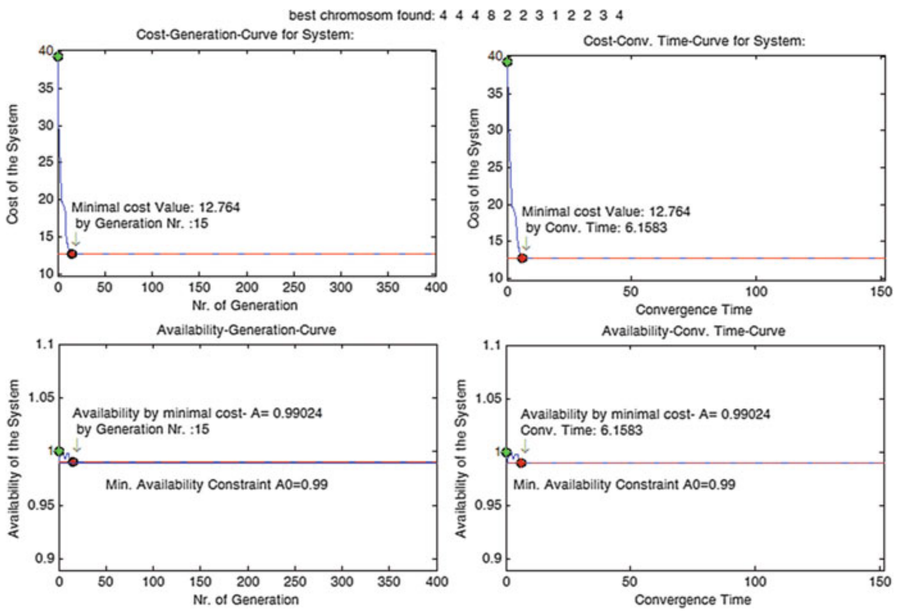

Fig. 16.10 Results of the genetic algorithm run of the heterogeneous case (Ouzineb—model containing six subsystems, availability constraint $A_0 = 0.990$ and 300 generations). The cost-generation and availability-generation curves dependencies are depicted on the left-hand side. The cost-time and availability-time dependencies are depicted on the right-hand side

serve to show how well the genetic algorithm customized for the heterogeneous case is performing in term of convergence speed, which can also be seen in the results depicted in Figs. 16.8, 16.9 and 16.10.

16.8 Conclusion and Future Works

Based on the facts and experimental results shown in the previous section and resumed in Table 16.1 for the different analyzed models, it can be recognized and concluded that the GA algorithm for heterogeneous series-parallel multi-states systems implemented in this work was performing in a great and efficient manner in term of convergence speed towards optimal results being expected and represented by Ouzineb in [5–7] and obtained by the hybrid GA + TS metaheuristic approach,

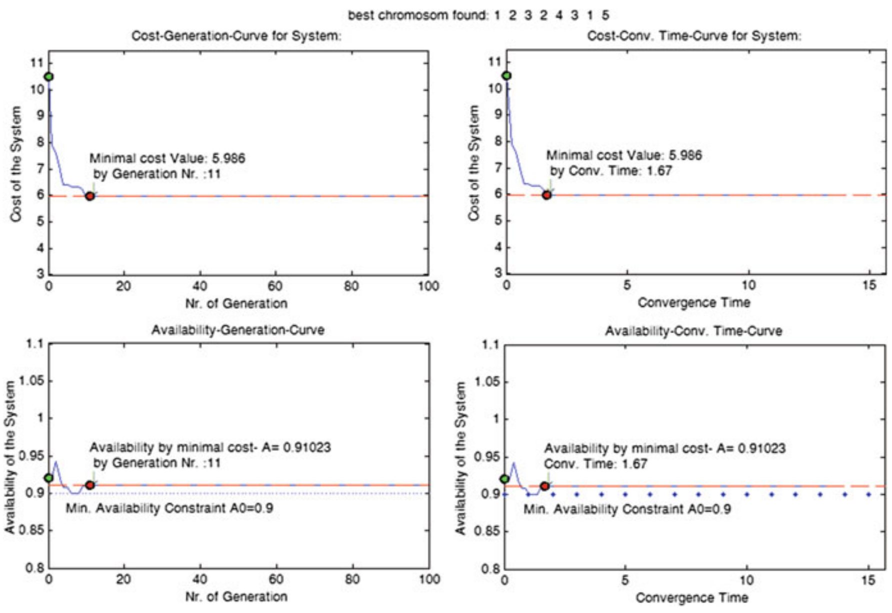

Fig. 16.11 Results of the genetic algorithm run of the homogeneous case for the same upper system (Levitin—model containing four subsystems) and subject to the same availability constraint $A_0 = 0.900$. As one can see in the heterogeneous case a better cost factor can be reached (5.423) than the homogeneous case (5.986) due to the fact that components have been mixed

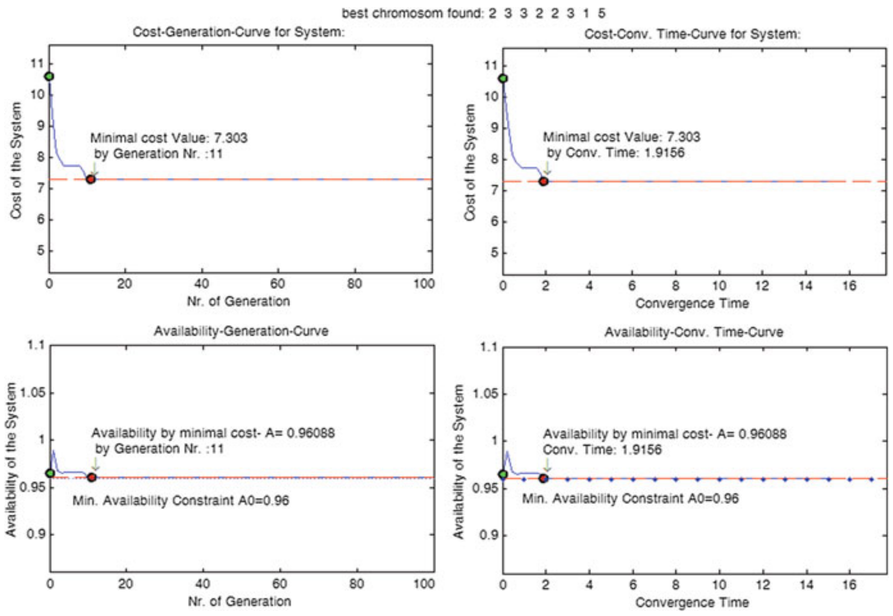

Fig. 16.12 Results of the genetic algorithm run of the homogeneous case for the same upper system (Levitin—model containing four subsystems) and subject to the same availability constraint A0 = 0.960. As one can see in the heterogeneous case a better cost factor can be reached (7.009) than the homogeneous case (7.303) due to the fact that components have been mixed

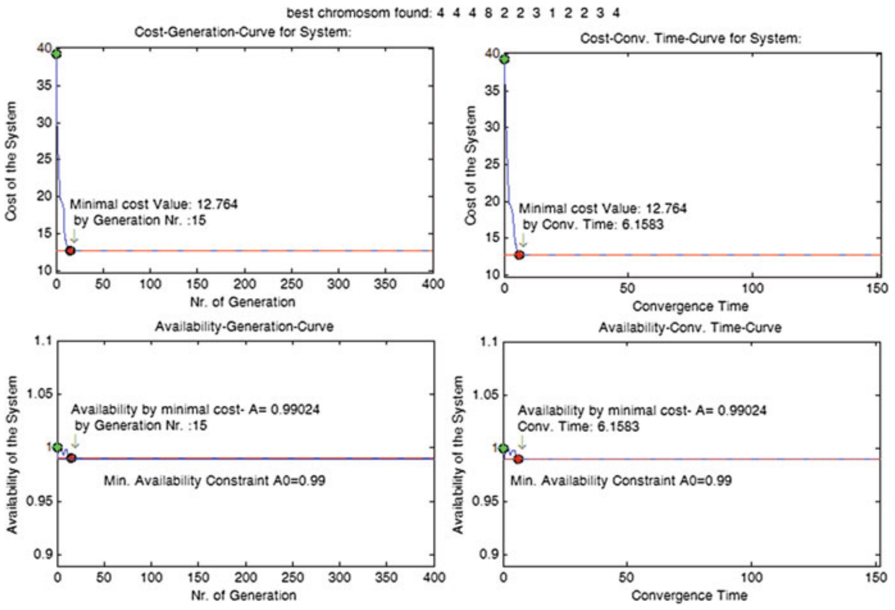

Fig. 16.13 Results of the genetic algorithm run of the homogeneous case for the same upper system (Ouzineb—model containing six subsystems) and subject to the same availability constraint A0 = 0.990. As one can see in both cases the same cost factor has been reached (12.764) but definitely for less computation time in the homogeneous case

in addition to the high accuracy in determining or finding the optimal solution (minimal system cost). And since genetic searching seems like searching for a small fish in a big ocean one small disadvantage or drawback is the standard one known in (heuristic) genetic approaches and that is resumed in the fact that the best optimal solution is not guaranteed or ensured in each run due to the limitation of the maximum number of iterations that may result, that some regions of the search or solution space that may include the optimal solution remains unexplored or out of reach.

The genetic approach implemented in this paper represents a very effective means in solving single objective constrained redundancy design problems like the complex heterogeneous one discussed in this work.

One of our future intentions is to tune genetic algorithms with local search algorithms targeting to increase the level of search accuracy. This kind of tuning is referred to in the literature as hybridization of genetic global searching algorithms.

Table 16.1 Computation results of the homogeneous and the heterogeneous case using the GA

Model name	Availability constraint A_0	Av. value $A(X,J)$ (hom.)	Av. value $A(X)$ (het.)	Cost $C(X,J)$ (*mln* \$) (hom.)	Cost $C(J(X))$ (*mln* \$) (het.)
lev4-(4/6)-3	0.900	0.9102	0.90075	5.986	5.423
	0.960	0.9609	0.96256	7.303	7.009
	0.990	0.9917	0.99148	8.328	8.180
lev5-(4/9)-4	0.975	0.9774	0.97615	16.450	12.855
	0.980	0.9808	0.98009	16.520	14.770
	0.990	0.9937	0.99211	17.050	15.870
ouz6-(4/11)-4	0.975	0.9790	0.9790	11.241	11.241
	0.980	0.9802	0.9802	11.369	11.369
	0.990	0.9902	0.9902	12.764	12.764

Model name	Best found chromosome homogeneous case [X,J]	Best found chromosome heterogeneous case J(X)	Conv. time (het.)
lev4-(4/6)-3	[1 2 3 2 4 3 1 5]	[4(1) 3(2) 1(3) 3(1),5(1)]	2.1992
	[2 3 3 2 2 3 1 5]	[1(3) 2(1),3(2) 1(3) 3(1),5(1)]	1.6700
	[3 3 3 5 1 3 1 2]	[1(3) 3(3) 1(3) 3(1),4(2)]	2.4501
lev5-(4/9)-4	[2 2 3 3 1 2 3 2 7 2]	[4(2),6(1) 5(6) 1(1),4(1) 7(3) 4(3)]	16.7701
	[2 6 3 3 1 2 5 2 7 2]	[4(2),6(1) 3(2) 2(1),3(2) 7(3) 3(2),4(1)]	14.0462
	[2 2 3 3 3 2 3 2 7 4]	[4(2),6(1) 3(2) 2(2),3(1) 7(3) 4(3)]	11.1849
ouz6-(4/11)-4	[4 4 5 7 2 1 3 1 2 2 3 4]	[3(4) 1(4) 2(5) 2(7) 3(2) 4(1)]	24.3273
	[4 5 5 8 2 1 3 1 2 2 3 4]	[3(4) 1(5) 2(5) 2(8) 3(2) 4(1)]	17.4778
	[4 4 4 8 2 2 3 1 2 2 3 4]	[3(4) 1(4) 2(4) 2(8) 3(2) 4(1)]	16.8972

It is important to mention at this point that the results got with the GA match either for homogeneous or for the non-homogeneous case the one got by Ouzineb using the hybrid GA + TS (Tabu Search) [5–7]. The best values of convergence time of the heterogeneous GA within 15 runs have been also included

References

1. Kuo, W., Rajendra Prasad, V., Tillman, F.A., Hwang, C.-L.: Optimal Reliability Design, Fundamentals and Applications. Cambridge University Press, Cambridge (2001)
2. Levitin, G., Lisnianski, A., Haim, H.B., Elmakis, D.: Genetic Algorithm and Universal Generating Function Technique for Solving Problems of Power System Reliability Optimization. The Israel Electric Corporation Ltd., Planning Development & Technology Division (2000)
3. Levitin, G., Lisnianski, A., Haim, H.B.: Redundancy optimization for series-parallel multi state systems. IEEE Trans. Reliab. **47**(2) (1998)
4. Lisnianski, A., Livitin, G., Haim, H.B., Elmakis, D.: Power system optimization subject to reliability constraints. Electr. Power Syst. Res. **39**, 145–152 (1996)
5. Ouzineb, M.: Heuristiques éfficaces pour l'optimisation de la performance des systèmes séries-parallèles. Département d'informatique et de recherche opérationnelle Faculté des arts et des sciences, Université de Montréal, 2009
6. Ouzineb, M., Nourelfath, M., Gendreau, M.: Tabu search for the redundancy allocation problem of homogenous series-parallel multi-state systems. Reliab. Eng. Syst. Saf. **93**, 1257–1272 (2008)
7. Ouzineb, M., Nourelfath, M., Gendreau, M.: A heuristic method for non-homogeneous redundancy optimization of series-parallel multi-state systems. J. Heuristics **17**(1), 1–22 (2009)
8. Yalaoui, A., Chu, C., Châtelet, E.: Reliability allocation problem in a series–parallel system. Reliab. Eng. Syst. Saf. **90**, 55–61 (2005)
9. Li, C.-y., Chen, X., Yi, X.-s., Tao, J.-y.: Heterogeneous redundancy optimization for multi-state series–parallel systems subject to common cause failures. Reliab. Eng. Syst. Saf. **95**, 202–207 (2010)
10. Chaaban, W., Schwarz, M., Börcsök, J.: Budgetary and redundancy optimisation of homogeneous series-parallel systems subject to availability constraints using Matlab implemented genetic computing. In: 24th IET Irish, Signals and Systems Conference (ISSC 2013)
11. Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
12. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading (1989)
13. Tian, Z., Zuo, M.J., Huang, H.: Reliability-redundancy allocation for multi-state series-parallel systems. IEEE Trans. Reliab. **57**(2), 303–310 (2008)
14. Affenzeller, M., Winkler, S., Wagner, S., Beham, A.: Genetic Algorithms and Genetic Programming, Modern Concepts and Applications. CRC Press, Boca Raton (2009)
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd revised and extended edition. Springer, Berlin (2011)
16. Tillman, F.A., Hwang, C.-L., Kuo, W.: Optimization techniques for system reliability with redundancy—a review. IEEE Trans. Reliab. **R-26**(3), 148–155 (1977)
17. Levitin, G.: The Universal Generating Function in Reliability Analysis and Optimization. Springer, London (2005)
18. Chaaban, W., Schwarz, M., Börcsök, J.: Cost and redundancy optimization of homogeneous series-parallel multi-state systems subject to availability constraints using a Matlab implemented genetic algorithm. In: Recent Advances in Circuits, Systems and Automatic Control, WSEAS 2013, Budapest, Hungary, 2013