versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

MULTIOBJECTIVE OPTIMIZATION OF AVAILABILITY AND COST IN REPAIRABLE SYSTEMS DESIGN VIA GENETIC ALGORITHMS AND DISCRETE EVENT SIMULATION

Isis Didier Lins
Enrique López Droguett*
Departamento de Engenharia de Produção
Universidade Federal de Pernambuco (UFPE)
isis.lins@gmail.com
ealopez@ufpe.br

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 11/2007; aceito em 12/2008 após 1 revisão
Received November 2007; accepted December 2008 after one revision

Abstract

This paper attempts to provide a more realistic approach to the characterization of system reliability when handling redundancy allocation problems: it considers repairable series-parallel systems comprised of components subjected to corrective maintenance actions with failure-repair cycles modeled by renewal processes. A multiobjective optimization approach is applied since increasing the number of redundancies not only enlarges system reliability but also its associated costs. Then a multiobjective genetic algorithm is coupled with discrete event simulation and its solutions present the compromise between system reliability and cost. Two examples are provided. In the first one, the proposed algorithm is validated by comparison with results obtained from a system devised as to allow for analytical solutions of the objective functions. The second case analyzes a repairable system subjected to perfect repairs. Results from both examples show that the proposed method can be a valuable tool for the decision maker when choosing the system design.

Keywords: availability, redundancy allocation, multiobjective optimization in system design.

Resumo

Esse artigo utiliza uma abordagem mais realista para a caracterização da confiabilidade de sistemas em problemas de alocação de redundâncias: são considerados sistemas série-paralelo formados por componentes sujeitos a ações de manutenção corretiva com ciclos de falha-reparo modelados por processos de renovação. É aplicada uma abordagem de otimização multiobjetivo, pois o aumento de redundâncias eleva a confiabilidade do sistema e também os seus custos. Assim, um algoritmo genético multiobjetivo é integrado com simulação discreta de eventos e suas soluções apresentam o compromisso entre confiabilidade e custo do sistema. Dois exemplos são fornecidos. No primeiro, o algoritmo proposto é validado através da comparação com resultados obtidos de um sistema criado de forma a permitir soluções analíticas das funções-objetivo. No segundo, analisa-se um sistema reparável sujeito a reparos perfeitos. Os resultados mostram que o método proposto pode ser uma ferramenta valiosa para o decisor no momento da escolha do projeto do sistema.

Palavras-chave: disponibilidade, alocação de redundâncias, otimização multiobjetivo em projeto de sistemas.

1. Introduction

A system is said to be *repairable* if, after a failure, it can be restored to operation by a maintenance procedure that does not consist of its entire substitution. When the time to repair is not negligible in relation to the operational time, system's reliability is measured by its availability. System availability can be defined as the probability of the system being operational in an arbitrary time instant (Rausand & Hoyland, 2004).

Customers expect that products (systems) they acquire perform according to the specifications informed by the manufacturer. System failures lead to costumer frustration and cost increase because of design modifications, changes in production processes, repairs and, consequently, warranty cost increase and reduction of sales (Droguett & Mosleh, 2006). Thus, the manufacturer goal is to design, develop and market systems in short periods of time at minimal cost and yet meet the requirements and customers' expectations. Reliability is therefore an inherent system characteristic that should be taken into consideration in the system development process.

As a result, the design process usually aims at high system availability and low associated costs. However, in general, the relation between availability and cost is direct, *i.e*., the higher the availability level, the higher the costs. Hence, the objectives of *designing a system with high availability* and *also with low associated costs* are normally conflicting.

To find the best solution of a problem with a single objective, one can make use of optimization methods. But, situations of practical interest frequently require the optimization of more than one objective – minimum costs, maximum reliability, maximum availability, minimum risk, among others – and a solution that simultaneously optimizes all the objectives is rarely within reach or it simply does not exist. In these cases, a multiobjective approach can be used to obtain a set of potential solutions (instead of one as in single objective problems) that are equally optimal from a multiobjective perspective, *i.e*., nondominated solutions (Coello *et al.*, 2002). Once this set has been found, the decision-maker can then choose any of the solutions based on his preferences and on the compromise among the considered objectives they present.

Genetic algorithms (GAs – Goldberg, 1989; Michalewicz, 1996) are probabilistic optimization techniques based on the natural evolution process that have been used to tackle a variety of single and multiobjective problems mainly when some features of the objective function, such as differentiability, continuity and convexity, are unknown or finding them is impractical. GAs have two important features for the multiobjective case: (i) they consider various potential solutions in a single run and (ii) the various objectives can be treated separately (Deb, 1999).

There are many works in literature regarding redundancy allocation problems and genetic algorithms. Cantoni *et al.* (2000) couples GA with Monte Carlo simulation in order to obtain an optimal plant design. They tackle a single objective redundancy allocation problem in which it is desired to obtain a combination of several repairable components that are subjected to imperfect repairs (see Doyen & Gaudoin (2004)) to be placed in a series-parallel layout in order to maximize system profit. Although the authors consider relevant aspects of system reliability, only a single objective is taken into account. Busacca *et al.* (2001) use a multiobjective GA to solve a redundancy allocation problem in a safety system and the objectives are to maximize the net profit drawn from system operation and system reliability during mission time. All components are supposed to have constant failure rates. Elegbede &

Adjallah (2003) aim to solve an availability allocation problem and describe a methodology based on GAs and weighting techniques to optimize the availability and the cost of a series-parallel repairable system. All components in each subsystem are identical and the authors consider an expression of the system asymptotic availability as the objective function regarding system availability. Goldbarg *et al.* (2005) apply GAs to determine the configuration of cogeneration systems with natural gas as the energy source and under the context of cost minimization. Chiang & Chen (2007) propose a multiobjective GA based on simulated annealing (Kirkpatrick *et al.*, 1983) to solve an availability allocation problem of a series-parallel repairable system but with the usually non-realistic consideration of exponentially distributed inter-arrival times.

In all the above mentioned works, the components reliability features are essentially the same: an underlying Exponential distribution (and hence a constant failure rate) governs the system failure process. Although Cantoni *et al.* (2000) go further in the system reliability aspect they also consider that components failure rates are constant from the time they return from a maintenance intervention to the time of the very next failure.

Taboada & Coit (2006) propose a multiobjective approach based on GAs to solve a redundancy allocation problem in a series-parallel system in which the objectives are to maximize system reliability, minimize system cost and system weight. The authors assume non-repairable components with time-independent reliability functions, *i.e.*, the effects of component degradation on the system reliability are not taken into consideration. Moreover, only acquisition costs are analyzed which, as a result of the previous assumption, are kept constant over the system mission time. Later Taboada *et al.* (2008) also consider a redundancy allocation problem with three objectives but, instead of maximizing system reliability as it is done in Taboada & Coit (2006), they use the universal moment generating function (Levitin, 2005) approach to evaluate the system availability.

Although the work of Marseguerra & Zio (2000) is not in the context of redundancy allocation, they combine GAs with Monte Carlo simulation with the aim of optimizing maintenance and repair policies of already defined systems, *i.e.*, they do not consider system design. They take into account a system gain function involving several costs and profit from plant operation as the single objective function to be evaluated by their binary-coded GA. Besides, in the context of spare parts allocation, Marseguerra *et al.* (2005) also integrate Monte Carlo simulation method and GAs for determining the optimal spare parts considering the objectives of maximizing net profit and minimizing total system volume. System availability is not taken into account as an objective itself, but it is included in net profit calculations. The authors use a ranking strategy to evaluate solutions and maintain the nondominated ones in a fixed length archive that is updated during the execution of the algorithm. In addition, all components are supposed to have exponentially distributed times between failures.

This paper attempts to overcome some of the limitations in the system reliability characterization and modeling in the context of redundancy allocation optimization. Indeed, a series-parallel repairable system is formed by components that are supposed to have failure-repair cycles modeled by renewal processes with Weibull distributed times between failures and Exponential repair times. Hence, the underlying stochastic process of the system's failure-repair process is not a renewal process and there is no analytical expression to the system availability. Moreover, for a given slot in a subsystem of the series-parallel system, multiple components capable of performing the same function but with different reliability and costs characteristics are considered. In order to obtain a more realistic

representation of the system dynamic behavior, the system availability is obtained via Monte Carlo simulation techniques, namely the discrete event simulation (DES) method (Banks *et al*., 2001). The flexibility provided by the DES permits the introduction of many real aspects in problem modeling and is especially useful in situations where an analytical treatment is prohibitive. Furthermore, a multiobjective GA is proposed to treat redundancy allocation problems involving repairable systems with two conflicting objectives: maximize system availability and minimize system cost. The proposed multiobjective GA algorithm is then coupled with the discrete event simulation in order to simultaneously optimize the system mean availability and the system overall cost. The latter objective corresponds not only to the acquisition costs related to individual components, but also it takes into consideration corrective maintenance costs, *i.e*., costs incurred in repairing and bringing back into operation a particular failed component. As a result, the decision maker can choose any of the obtained system designs based on its own preferences and also on the provided information about the system dynamic behavior during its lifetime cycle. Such information is a valuable aid in the strategic choice of the system design.

This paper is organized as follows. Section 2 introduces some aspects of renewal processes, alternating renewal processes, and availability concepts. Section 3 reviews traditional multiobjective methods and some relevant evolutionary multiobjective approaches. The proposed multiobjective GA is presented in Section 4. Section 5 discusses the basic ideas underpinning the coupling of the proposed multiobjective GA algorithm with the discrete event simulation algorithm for quantifying repairable systems availability. The subsequent sections discuss in detail two application examples. In Section 6 the proposed approach is validated by comparison with the results obtained from a system devised as to allow for analytical solutions of the objective functions. Then, in Section 7, the redundancy allocation problem of series-parallel repairable systems is discussed by means of a more realistic case in which it is desired to maximize system availability and minimize system cost. Section 8 presents some concluding remarks.

2. Preliminaries

2.1 Renewal Processes and Alternating Renewal Processes

According to Rigdon & Basu (2000), a counting process is a Renewal Process (RP) if the times between events X_1, X_2, … are independent and identically distributed. It is assumed that the times to repair are negligible if compared to the system operational time. In addition, repair actions are considered as perfect repairs, which mean that after a failure the system is returned into operation in an "as good as new" condition. When times to repair are not small if compared to the system operational time, they might be considered in the analysis of the system failure-repair process. In these cases, this process can be modeled using Alternating Renewal Processes (ARP).

Indeed, in an ARP, the times between failures X_1, X_2,… are independent and identically distributed. After a failure, the system becomes unavailable due to repair action. The times to repair D_1, D_2,… are independent and identically distributed. It is also assumed that $X_i + D_i$, $i = 1, 2, 3,\ldots$ are independent random variables. The binary state variable $X(t)$ indicates the system state in time t. Figure 1 illustrates the behavior of a system that has a failure-repair process modeled by an ARP.

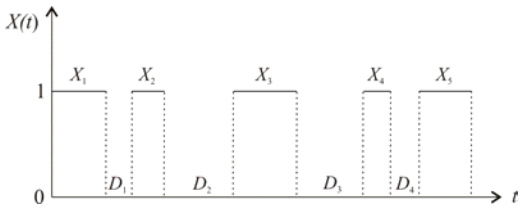

Figure 1 – Alternating renewal process.

When the system is configured in series and all of its components are modeled by Homogeneous Poisson Processes (HPPs), the superimposed process is also an HPP (Rausand & Hoyland, 2004). However, in general, an ARP is not a RP and the stochastic process that characterizes the failure-repair process of the system is unknown or it does not have an analytical solution. As a consequence, the reliability metrics of interest, such as the time to the *r*th failure, cannot be evaluated analytically. In these situations, the discrete event simulation can be used as an alternative.

2.2 Availability

The availability is a metric related to repairable systems. There are usually four availability metrics associated to repairable systems: instantaneous availability, limit availability, average availability and limit average availability (Rausand & Hoyland, 2004). In this paper, the following metrics are of interest:

- Instantaneous availability that is defined as the probability of the system being operational in an arbitrary instant of time. It also can be interpreted as the mean value of the binary random variable $X(t)$:

  $$A(t) = P[X(t) = 1] = E[X(t)],\ t \geq 0 \qquad (1)$$

- Average availability that indicates the expected time proportion of $(0, t]$ in which the system is operational:

  $$A_{av}(t) = \frac{1}{t}\int_0^t A(\tau)d\tau \qquad (2)$$

3. Multiobjective Optimization and Genetic Algorithms

Real situations frequently demand the simultaneous achievement of multiple objectives. Minimal costs, maximal reliability/availability, minimal risks are some of the objectives usually desired in reliability field. In addition, some of these objectives might be conflicting.

The general formulation of a multiobjective optimization problem is:

$$\begin{aligned} &\text{Maximize} && \mathbf{z} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \\ &\mathbf{x} \\ &\text{Subject to} && g_i(\mathbf{x}) = 0,\ i = 1, \ldots, p \\ & && h_i(\mathbf{x}) \leq 0,\ i = p+1, \ldots, m \end{aligned} \qquad (3)$$

where **z** is the vector formed by k objective functions, **x** is the n-dimensional vector of decision variables, p is the number of equality constraints $g_i(\mathbf{x})$, and $m - p$ is the number of inequality constraints $h_i(\mathbf{x})$.

The concepts of dominance and nondominance are essential to multiobjective optimization and they are introduced in next section.

3.1 Dominance and Nondominance

In single objective optimization, one unique solution is obtained – the optimal solution. However, when multiple conflicting objectives are considered, finding a single solution is very difficult to obtain or it might even not exist. Under this circumstance, it is desired to obtain a set of *nondominated* solutions. This set is also known as optimal Pareto set, in regard to the italian economist Vilfredo Pareto, who generalized the "optimal" concept in multiobjective context (Coello *et al.*, 2002).

A dominance/nondominance relation can be verified for each pair of solutions. A solution $\mathbf{x_1}$ dominates a solution $\mathbf{x_2}$, with $\mathbf{x_1} \neq \mathbf{x_2}$, if for all objectives, it has a performance at least as good as the performance of $\mathbf{x_2}$ and, at least for one of the objectives, its performance overcomes the performance of $\mathbf{x_2}$. In summary:

$$\mathbf{x_1} \succ \mathbf{x_2} \quad \Leftrightarrow \quad f_h(\mathbf{x_1}) \geq f_h(\mathbf{x_2}),\ \forall\, h \quad \text{and} \quad f_h(\mathbf{x_1}) > f_h(\mathbf{x_2}), \text{ for some } h, \tag{4}$$

where $\succ$ denotes dominance and $\mathbf{x_2}$ is a dominated solution for a maximization problem. If a minimization problem is considered, the signs $\geq$ and $>$ in (4) are replaced by $\leq$ and $<$ respectively. On the other hand, if one of the conditions in the right side of (4) is not satisfied, $\mathbf{x_1}$ is said to be nondominated in relation to $\mathbf{x_2}$ (and vice-versa). That is, for a quantity of objectives, $\mathbf{x_1}$ overcomes the performance of $\mathbf{x_2}$ and, for the remaining objectives, $\mathbf{x_2}$ overcomes the performance of $\mathbf{x_1}$. The nondominance relation is also observed when $\mathbf{x_1} = \mathbf{x_2}$.

The concepts of local and global optimality in single optimization are replaced by *local optimal Pareto set* and *global optimal Pareto set*, respectively, in the multiobjective case (Deb, 1999; Zitzler, 1999). To define these concepts, let **X** be the set of all **x** that satisfies the constraints in (3), *i.e.*, the feasible set:

- Local optimal Pareto set ($\underline{P}$): for all $\mathbf{x} \in \underline{P}$, there is no solution $\mathbf{x'} \in \mathbf{X}$ satisfying $\|\mathbf{x'} - \mathbf{x}\| < \varepsilon$ which dominates any member of $\underline{P}$ ($\|\cdot\|$ is the distance between two points and $\varepsilon > 0$). The solutions in $\underline{P}$ constitute the local optimal Pareto set.
- Global optimal Pareto set ($\overline{P}$): for all $\mathbf{x} \in \overline{P}$, there is no $\mathbf{x'} \in \mathbf{X}$ such that $\mathbf{x'} \succ \mathbf{x}$.

Deb (1999) emphasizes that a set of nondominated solutions is defined in the context of a sample of the feasible search space. On the other hand, a global optimal Pareto set is a set of nondominated solutions whose sample is equal to the entire feasible search space.

Applying the solutions of $\overline{P}$ in the objective functions, it is obtained the *global Pareto front* $F\overline{P}$ (see (5)). Following the same reasoning for the solutions of $\underline{P}$, the *local Pareto front* $F\underline{P}$ (see (6)) is found:

$$F\overline{P} = \{\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})),\ \forall\, \mathbf{x} \in \overline{P}\} \tag{5}$$

$$F\underline{P} = \{\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})),\ \forall\, \mathbf{x} \in \underline{P}\} \tag{6}$$

3.2 Traditional Multiobjective Methods

The traditional methods of multiobjective optimization are the Weighted Sum Method and the ε-Perturbation Method (Coello *et al.*, 2002; Deb, 1999). Both of them handle single objective optimization problems. The Weighted Sum Method uses weights to aggregate all the objective functions into a single objective-function. The weights do not represent the relative importance among objectives: they are only factors that may be altered to locate different points in the optimal Pareto set. In the ε-Perturbation Method, in turn, an objective chosen arbitrarily (or the one considered as the most important) is optimized and the others become constraints that must satisfy acceptable levels previously defined. Different Pareto optimal solutions are found by varying the acceptable levels.

One of the main drawbacks of the Weighted Sum Method is the fact that it cannot find solutions in non-convex regions of the Pareto front if the Pareto front is non-convex (Messac *et al.*, 2000). In the case of the ε-Perturbation Method, the choice of the acceptable levels demands a preliminary analysis since inappropriate values can result in empty feasible sets (see Deb (1999)). In addition, both traditional methods not only require multiple runs in order to obtain different optimal Pareto solutions (it is expected that they are indeed optimal Pareto solutions), but also a considerable knowledge of the problem is a must.

Alternatively, evolutionary algorithms such as GAs can be used to resolve multiobjective problems. These algorithms overcome some of the drawbacks in the abovementioned traditional methods. Evolutionary algorithms handle with many potential solutions simultaneously, which allows the achievement of different Pareto solution in a single execution of the algorithm. Also, they do not impose any requirement regarding the convexity of the Pareto front.

3.3 Genetic Algorithms

Genetic algorithms computationally simulate the natural evolution process. They are mainly directed toward the optimization of problems that have some features not tolerable by traditional mathematical programming methods as discussed above. In the following, the nomenclature and the steps of a single objective GA are briefly described.

Firstly, it is required a representation for the decision variables vectors (the potential solutions). A binary, an integer or a real representation may be used. Each coded decision variable is named *gene* and the set of *genes* forms the *genotype*. When the genotype is decoded into its original form, one obtains the *phenotype*. If either the integer or the real representation is chosen, then the genotype is equal to the phenotype. An *individual* is a potential solution which has a related genotype and phenotype. A set of individuals constitutes a *population*. The most common genetic operators are *selection*, *crossover* and *mutation*.

The selection of individuals is based on their *fitness*. The fitness for an individual is obtained by substituting its phenotype in the *fitness function* that frequently is, in numerical optimization problems, the same as the objective function (Coello *et al.*, 2002). The fittest individuals have more chances to be selected. When it is considered a constrained optimization problem, if the individual does not satisfy the constraints a *penalization* is incorporated to its fitness. In maximization problems, an individual's fitness is reduced by a certain value, whereas it is increased in case of minimization problems. In these situations,

the individual selection is anchored in its *penalized fitness*. The main purposes of the selection are to restrict the search to some regions of the search space and to improve the average quality of the population.

Some of the selected individuals are chosen to participate in the crossover according to an established crossover probability. In the crossover step, the individuals (*parents*, often taken in couples) exchange parts of their genotype to generate the *children*. Subsequently, the population and the children are subjected to mutation, that is, parts of the genotype of some individuals are modified in accordance with a predefined mutation probability. Both crossover and mutation aim to generate new solutions in the search space by the variation of the existing ones.

Most of the algorithms have a population composed by a fixed number of individuals. In these cases, some individuals must be discarded, since the population is increased after the crossover phase. The substitution of individuals is called *replacement*. Two well-known replacement techniques are *children replace parents* and *elitist replacement*. In the former, the parents are discarded and their positions are filled with their respective children. In the latter, after the mutation, the fitness of each individual (from population and children) is evaluated and then the individuals are ordered. The first N individuals form the new population, where N is the fixed size of the population.

The simulation of the evolution starts with a random generation of the initial population (or according to a predefined strategy). Afterward, the (penalized) fitness evaluation is performed. Then, selection, crossover, mutation and replacement are applied. In this way, a new population is created and an iteration of the algorithm, named *generation*, is ended. All the other steps (except the initial population generation) are repeated until a stop criterion is reached. An usual stop criterion is the number of generations.

3.4 Multiobjective Genetic Algorithms

In addition to the fact that more than one objective is taken into account, the main difference between the single objective GA and the multiobjective GA is the selection phase. In the multiobjective case, the concept of dominance is directly or indirectly incorporated in that step. The main multiobjective methodologies involving evolutionary algorithms are listed in Table 1.

Table 1 – Main methods of multiobjective optimization via evolutionary algorithms.

Method	Author(s) and Date
Vector Evaluated Genetic Algorithms (VEGA)	Schaffer (1985)
Multiobjective Genetic Algorithm (MOGA)	Fonseca & Fleming (1993)
Niched-Pareto Genetic Algorithm (NPGA)	Horn *et al.* (1994)
Nondominated Sorting Genetic Algorithm (NSGA)	Srinivas & Deb (1994)
Strength Pareto Evolutionary Algorithm (SPEA)	Zitzler & Thiele (1999)
Nondominated Sorting Genetic Algorithm II (NSGA-II)	Deb *et al.* (2002)
Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D)	Zhang & Li (2007)

In multiobjective GAs, it is often used an "elaborated fitness" that is capable of incorporating all features presented by an individual in all objectives. In MOGA and NSGA-II the "elaborated fitness" can be, for example, based on the number of solutions which dominates each individual such that nondominated individuals have a better fitness value, whereas the individual dominated by the largest number of solutions has the worst fitness value. In NSGA-II, in addition to the fitness based on dominance and nondominance, it is also employed a "distance fitness", which is calculated according to distances among individuals. Solutions with larger "distance fitness" are preferred in order to increase the exploration of the Pareto front.

In the context of reliability, Marseguerra *et al.* (2005) apply GAs combined to Monte Carlo simulation to define the number of spare parts in a safety system. They use a ranking strategy of the population based on the number of solutions that dominates each individual, a procedure similar to MOGA. During search, nondominated solutions are stored and updated in an archive with predefined size. In each generation, the current nondominated solutions are compared to the ones already stored in the archive where the following rules must be satisfied:

- If a new solution dominates individuals in the archive, these individuals are discarded and the new solution is added to the archive.
- If a new solution is dominated by individuals of the archive, it is not stored in the archive.
- If a new solution neither dominates nor is dominated by solutions in the archive:
    - If the archive is not full, the new solution is stored.
    - If the archive is full, the new solution replaces the most similar solution in archive (this similarity is evaluated by the euclidean distances between fitness values presented by the individuals).

Taboada & Coit (2006) developed an evolutionary algorithm based on NSGA-II to tackle redundancy allocation problems, namely MOEA-DAP (Multiple Objective Evolutionary Algorithm for Solving Design Allocation Problems). The algorithm uses two elaborated fitness metrics: the first one is anchored in the euclidean distance that an individual presents in relation to the others in the objective space, and the other considers the number of solutions that an individual dominates. These two metrics are combined to form the "aggregated fitness". At each generation, the dominated solutions are discarded such that the population is only composed by nondominated solutions. Furthermore, infeasible individuals cannot be generated, thus individual penalization is not required. It is also used an elitist strategy to construct a new population – a percentage of the best individuals goes further in next generation – and so individuals are ranked in accordance to the aggregated fitness they present. The next section discusses the proposed multiobjective GA.

4. Proposed Multiobjective Genetic Algorithm

This section presents the proposed multiobjective GA. As discussed earlier, the proposed method is based on some features presented by Marseguerra *et al.* (2005) and by Taboada & Coit (2006).

Similarly to Marseguerra *et al.* (2005), the proposed GA algorithm makes use of the strategy of storing nondominated individuals of each generation in an auxiliary population which is

updated at every generation. On the other hand, while Marseguerra *et al.* (2005) predefines the size of the nondominated individuals archive, the auxiliary population in the developed GA algorithm has not a fixed size. In this way, all nondominated solutions generated during the execution of the algorithm are stored in the auxiliary population. In addition, The proposed method eliminates dominated individuals at every generation, as in the method in Taboada & Coit (2006). Using this approach, the individuals are likely to be closer to the Pareto front. However, differently from Taboada & Coit (2006), the proposed algorithm allows for the generation of infeasible individuals during the evolution process. Hence, a penalization function is used to penalize the individuals that do not satisfy the problem constraints. By means of this approach, it is expected an increase of the population variability, which permits a more comprehensive exploration of the search space and consequently of the Pareto front.

It is important to note that the proposed algorithm does not make use of individuals ranking. Thus, there is no need to create fitness metrics as is the case, for example, in Fonseca & Fleming (1993), Deb *et al.* (2002), Marseguerra *et al.* (2005), Taboada & Coit (2006) and Taboada *et al.* (2008). In this way, each individual has a number of fitness values equal to the number of considered objectives. The next section describes the steps of the proposed multiobjective GA to handle redundancy allocation problems.

4.1 Steps of the Proposed Multiobjective Genetic Algorithm

Let N be the fixed size of population P, P_{aux} the auxiliary population that stores nondominated individuals at each iteration, and $\mathbf{x}$ the n-dimensional vector of decision variables (components quantities). The steps taken in the proposed algorithm are as follows (see Figure 6-a):

1) Individual Representation

The type of redundancy allocation problems considered in this paper has integer-valued decision variables. Therefore individual integer coding was chosen, since it is a more natural way of representation. For the sake of illustration of individual representation, consider a system formed by two subsystems S_1 and S_2 in series. Each one of these subsystems can have at least 1 and at most 4 components in parallel. Also, suppose that S_1 has 3 component options S_{11}, S_{12} and S_{13} (represented by the variables x_1, x_2 and x_3, respectively) and S_2 has 4 component options S_{21}, S_{22}, S_{23} and S_{24} (characterized by the variables x_4, x_5, x_6 and x_7, in this order). Figure 2 depicts the phenotype of an individual modeled according to these assumptions. The phenotype is a vector composed by 7 variables, where the first 3 correspond to S_1, and the remaining ones correspond to S_2. The value of each variable indicates the number of each type of component. This individual phenotype representation has also been used by Taboada & Coit (2006).

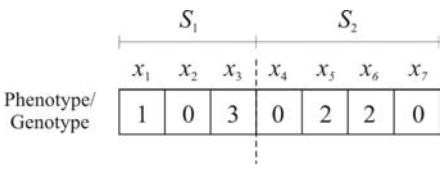

Figure 2 – Integer-coded individual.

The system represented by the individual of Figure 2 is illustrated in Figure 3.

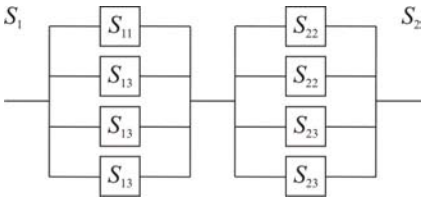

Figure 3 – System configuration of the individual in Figure 2.

2) Generation of the Initial Population

In order to create the initial population, N individuals are randomly constructed. Each individual decision variable j is generated according to a uniform distribution defined in $[l_j, u_j]$. If an individual $P[i]$ is composed by **x**, its random generation is described by the algorithm presented in Figure 4:

procedure INDRAND(l_1, u_1, l_2, u_2, … , l_n, u_n)
  for $j = 1 \ldots n$ do
    $x_j \leftarrow$ RAND($\{l_j, l_{j+1}, \ldots, u_j\}$) ▷ uniform random choice of an integer in $[l_j, u_j]$
return $x_1, x_2, \ldots, x_n$ ▷ returns a random individual

Figure 4 – Algorithm to randomly generate an individual

3) Calculating Fitness, Violation and Individual Penalization

Individual fitness values must be calculated for each objective function $f_1, f_2, \ldots, f_k$. Then, individuals' feasibility is verified by assessing the corresponding violation. The notation $v_i(\mathbf{x})$ represents the individual violation formed by **x** in relation to the ith constraint and is given by:

$$v_i(\mathbf{x}) = \max \{0, |g_i(\mathbf{x})| - \varepsilon \}, \quad \text{if } 1 \le i \le p \tag{7}$$

$$v_i(\mathbf{x}) = \max \{0, |h_i(\mathbf{x})|\}, \quad \text{if } p + 1 \le i \le m \tag{8}$$

where m is the total number of constraints, p is the number of equality constraints, $m - p$ is the number of inequality constraints, and ε is the equality constraint precision. The expression in equation (8) must be used in constraints with $\le$ or <. If a constraint has $\ge$ or > signs, it may be modified to a $\le$ or < constraint, before evaluating equation (8).

Violations are used in penalty calculations. A dynamic penalization function is adopted (Joines & Houck, 1994):

$$pen(\mathbf{x}) = (c \cdot gen)^{\alpha} \sum_{i=1}^{m} v_i(\mathbf{x})^{\beta} \tag{9}$$

where $c = 0.5$, $\alpha = \beta = 2$, m is the number of constraints, and gen is the current generation. This penalty function is said to be dynamic because it depends on the number of the current generation. At the start of the evolution process, it is suitable to allow larger violations, as infeasible individuals can generate feasible ones in subsequent generations and they also

permit a greater exploration of the search space (Martorell *et al.*, 2000). However, as the algorithm proceeds, it is desirable to increase the penalties for infeasible individuals in order to obtain the maximum number as possible of feasible individuals at the end of the optimization process.

The penalized fitness values of an individual $P[i]$ (considering a maximization problem) are defined as follows:

$$f_j^{pen}(\mathbf{x}) = f_j(\mathbf{x}) - pen(\mathbf{x}), \qquad j = 1, 2, \ldots, k, \qquad \text{if } P[i] \text{ is infeasible} \tag{10}$$

$$f_j^{pen}(\mathbf{x}) = f_j(\mathbf{x}) \qquad j = 1, 2, \ldots, k, \qquad \text{if } P[i] \text{ is feasible} \tag{11}$$

4) Selection

Firstly, in the selection step, the relation of dominance among individuals in the *current population* is evaluated according to their penalized fitness values. The dominated solutions are eliminated from P. On the other hand, the nondominated individuals continue in P and are candidate solutions that may be stored in P_{aux}. Afterward, the P_{aux} updating takes place in accordance with the following rules:

- If $P[i]$ is a candidate solution and is dominated by already stored individuals in P_{aux}, $P[i]$ is discarded.
- If $P[i]$ is a candidate solution and dominates individuals in P_{aux}, then such dominated solutions are eliminated from P_{aux} and a copy of $P[i]$ is inserted into P_{aux}.
- If $P[i]$ is a candidate solution and neither dominates nor is dominated by solutions in P_{aux}, a copy of $P[i]$ is stored in P_{aux}.

Since the dominated individuals are eliminated from P, its size is reduced to N_r ($N_r \leq N$). In order to maintain the population with N individuals, $N - N_r$ solutions are randomly selected from P_{aux} and inserted into P. It is important to emphasize that P_{aux} has not a predefined size and that it may contain infeasible solutions.

5) Crossover

After selection, a random number is generated for each individual in P. If this number is less than the crossover probability p_c for a certain $P[i]$, this individual will participate in the crossover.

In this paper, it is used a crossover procedure for integer variables based on an adaptation of the BLX_α crossover for real variables (see Herrera *et al.* (1998)). Indeed, for each pair of individuals (parents), m positions are defined in their phenotypes by the generation of random numbers in $[1, n]$, where n is the total number of variables (genotype/ phenotype length). For exemplifying, suppose that $Parent^1 = (x_1^{P1}, ..., x_j^{P1}, ..., x_n^{P1})$, that $Parent^2 = (x_1^{P2}, ..., x_j^{P2}, ..., x_n^{P2})$ and that only the jth variable is subjected to crossover ($m = 1$). Two random numbers r_1 and r_2 are generated in the interval $I = [x_j^{\min} - \alpha \cdot (x_j^{\max} - x_j^{\min}), x_j^{\max} + \alpha \cdot (x_j^{\max} - x_j^{\min})]$, where $x_j^{\min} = \min(x_j^{P1}, x_j^{P2})$ and $x_j^{\max} = \max(x_j^{P1}, x_j^{P2})$. Subsequently, the jth variable of $Child^1$ and of $Child^2$ are set respectively as $x_j^{C1} = \lfloor r_1 \rfloor$ and $x_j^{C2} = \lfloor r_2 \rfloor$, where $\lfloor \cdot \rfloor$ means the round-down integer. If any

of the values x_j^{C1} or x_j^{C2} exceeds one of the original *j*th variable bounds l_j or u_j, then such value is set equal to the bound it has just surpassed. For example, if $x_j^{C2} < l_j$, then $x_j^{C2} = l_j$.

Note that when $\alpha = 0$, then $I = [x_j^{\min}, x_j^{\max}]$, that is, the region where children are generated is limited by their parents variable values. If $\alpha < 0$, then *I* becomes smaller and the algorithm tends to have a population with low diversity levels and can also converge prematurely. On the other hand, if $\alpha > 0$, children can have variable values beyond the interval made up by their parents and the search space can have a more comprehensive exploration. For real-coded GAs, a recommended value of α is 0.5. However, in the proposed integer-coded multiobjective GA, it is used $\alpha = 1$.

6) Replacement

The adopted replacement strategy is children replace parents. According to this approach, after crossover, the parents can be immediately replaced by their corresponding children. In this case, the mutation is applied in *P* and after the replacement step. If an elitist replacement were to be assumed, the mutation must be employed in *P* and in the resulting individuals of the crossover, thus, before the replacement step.

7) Mutation

The mutation step consists in changing the value of a variable in an individual phenotype. For each position of an individual phenotype is generated a random number. If this random number is less than or equal to the mutation probability p_m, the content of the corresponding position *j*, is changed by a random number generated in $[l_j, u_j]$. The mutation is applied in all population.

The algorithm is repeated N_{gen} times, where N_{gen} is the number of generations, which also corresponds to the stop criterion. At the end, a last update in P_{aux} takes place and the proposed algorithm returns the nondominated individuals of P_{aux} that are feasible solutions. The proposed multiobjective GA is shown in Figure 5 where the selection phase is further detailed.

procedure MULTIOBJECTIVEGA (N, N_{gen}, l_1, u_1, … , l_n, u_n, f_1, … , f_k, g_1, … , g_p, h_{p+1}, … , h_m, p_c, p_m)
  ▷ generate initial population
  for $i = 1 \ldots N$ **do**
    $P \leftarrow P \cdot \langle$ INDRAND(l_1, u_1, … , l_n, u_n) $\rangle$
  end for
  for $g = 1 \ldots N_{gen}$ **do**
    ▷ initialize vectors of nondominated and dominated individual indexes of P in relation to f_1^{pen},…, f_k^{pen}, respectively J_B and J_R
    $J_B \leftarrow \langle\ \rangle$, $J_R \leftarrow \langle\ \rangle$
    for $i = 1 \ldots N$ **do**
      if $P[i]$ is nondominated in P in relation to f_1^{pen},…, f_k^{pen} **then**
        $J_B \leftarrow J_B \cdot \langle i \rangle$

```
            ▷ update P_aux
            if P[i] is dominated by solutions already stored in P_aux then
                ▷ P[i] does not enter in P_aux
            else
                if P[i] dominates solutions in P_aux then
                    ▷ eliminate dominated individuals from P_aux
                P_aux ← P_aux . ⟨P[i]⟩
        else J_R ← J_R . ⟨i⟩
    end for
    ▷ perform crossover and replace parents by children
    ▷ perform mutation
  end for
  ▷ update P_aux
  return feasible individuals from P_aux that are nondominated in relation to f_1, ..., f_k
end procedure
```

Figure 5 – Proposed multiobjective GA pseudocode.

Although the proposed multiobjective GA is quite straightforward, the integer representation associated with the auxiliary population updating at every generation improves the quality of the obtained solutions. Indeed, this was observed due to the fact that essentially the same algorithm was implemented with a binary representation and without the auxiliary population updating and the final results were considerable worse than the ones obtained by the multiobjective GA discussed in this paper.

5. Coupling the Discrete Event Simulation with the Multiobjective Genetic Algorithm

This paper couples the multiobjective genetic algorithm proposed in the previous section with Monte Carlo simulation for a more realistic representation of the dynamic behavior of a system. This technique is costly in terms of computation effort, but its application may be justified by a problem modeling closer to real world situations and/or by the possibility of handling non-analytical problems. In particular, discrete event simulation is used to analyze the behavior of a system, which consists in generating random discrete events during simulation time with the aim of creating a "typical" scenario for a system as to allow for the evaluation of some of its features that are of interest for the calculation of the objective functions of an individual $P[i]$ defined in Section 4. Both algorithms were implemented in C++ by the authors.

As depicted in Figure 6, the coupling takes place at the fitness evaluation step. For a given generation in the genetic algorithm evolution process, the availability objective function is estimated for every individual in the candidate solutions vector $P[i]$, $i = 1,\ldots, N$, via the discrete event simulation algorithm. For a predetermined mission time, t_n, the ith individual (a candidate system) undergoes stochastic transitions between its possible states which, for the problem of interest, correspond to available and unavailable (under repair) states. The system availability estimation is based on its components, *i.e.*, the system evolves through states of availability and unavailability depending on the components states: available or unavailable due to corrective maintenance.

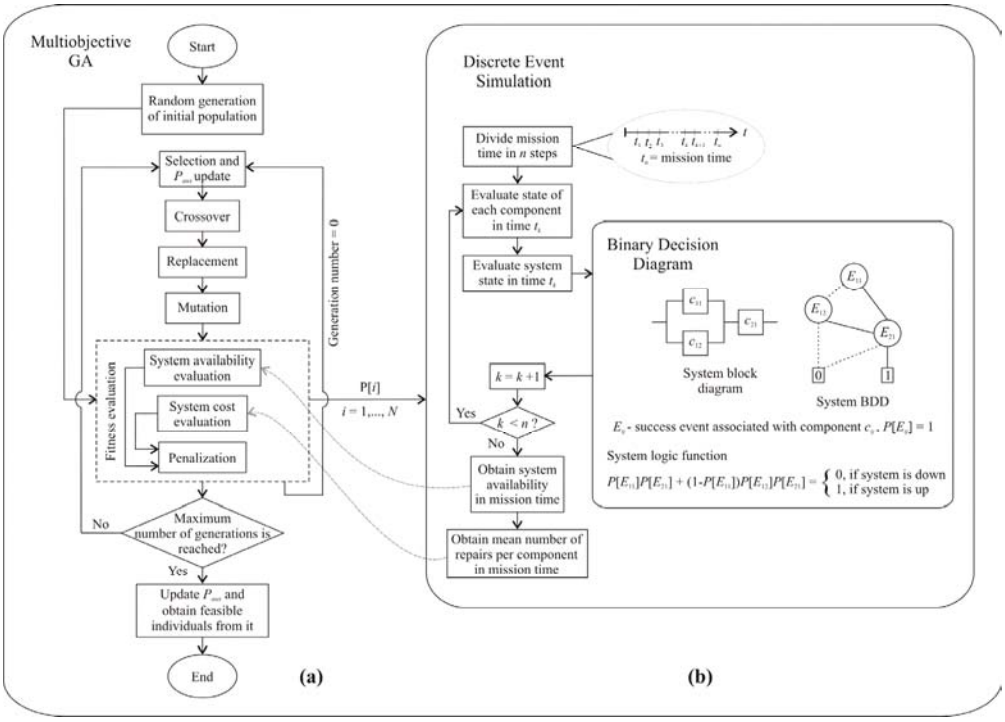

Figure 6 – Proposed multiobjective GA+DES.

Indeed, the availability quantification for the *i*th individual starts by subdividing the mission time t_n into n time steps. For a given iteration, say t_k, with $k = 1,\ldots, n$, it is evaluated the current state for each component, as shown in Figure 6-b. In order to determine the system state from its components states at t_k, one needs to construct the relationship characterizing the system logic as a function of the components. In the proposed approach, this is achieved by means of Binary Decision Diagrams – BDD (Rauzy, 2001). Thus, in order to keep track of the simultaneous evolutions of all components of the *i*th individual as well as of the system itself during the mission time, the observed realizations of the following random variables are recorded: number of failures/repairs of each component, number of times the system fails, the time intervals for which each component is under repair (corrective maintenance), and the time intervals during which the system is in the available state. The DES is replicated several times such that independent realizations of these random variables are obtained and then used to estimate the point availability, $A(t)$, for the *i*th system over the mission time. Then, one obtains estimates for the two objective functions, namely the system average availability $\overline{A}$, and the system total cost, where both metrics are estimated over the mission time. These results are then fed back to the GA algorithm, as illustrated in Figure 6.

6. Example Application 1: Validating the Proposed Multiobjective Genetic Algorithm

In this first numerical example, the proposed approach is validated by comparison with results obtained from a system devised as to allow for analytical solutions of the objective functions. In fact, it is considered a series-parallel non-repairable system adapted from

Taboada & Coit (2006), where one wants to maximize system reliability and minimize system cost.

The non-repairable system is composed by 3 subsystems S_1, S_2 and S_3. Components reliabilities are supposed to be constants during mission time and only the acquisition costs are considered in the system cost calculation. Each subsystem must have at least 1 component and at most 8 components. S_1 and S_3 have each one 5 component options, and S_2 has 4 component options. The general mathematical formulation is presented as follows:

$$\text{Maximize} \quad \prod_{i=1}^{s}\left[1-\prod_{j=1}^{m_i}(1-r_{ij})^{x_{ij}}\right] \tag{12}$$

$$\text{Minimize} \quad \sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij} \tag{13}$$

$$\text{Subject to} \quad \sum_{j=1}^{m_i} x_{ij} \le n_i^{\max}, \qquad i = 1, 2, \ldots, s$$

$$\sum_{j=1}^{m_i} x_{ij} \ge 1, \qquad i = 1, 2, \ldots, s$$

$$x_{ij} \in Z^{+},$$

where s is the number of subsystems, x_{ij} is the quantity of the jth component in subsystem i, m_i is the total number of components in ith subsystem, $n_i^{\max}$ is the maximum number of components in subsystem i, r_{ij} and c_{ij} are, respectively, the reliability and the acquisition cost of the jth component of ith subsystem. In the present example, $s = 3$, $n_i^{\max} = 8$, $i = 1, 2, 3$ and components features are shown in Table 2. Note that (12) is nonlinear and that (13) is linear.

Table 2 – Components reliabilities and costs (adapted from Taboada & Coit (2006)).

	S_1		S_2		S_3	
Component	r_{ij}	c_{ij}	r_{ij}	c_{ij}	r_{ij}	c_{ij}
1	0.94	900	0.97	1200	0.96	1000
2	0.91	600	0.86	300	0.89	600
3	0.89	600	0.70	200	0.72	400
4	0.75	300	0.66	200	0.71	300
5	0.72	200	–	–	0.67	200

In this example, the solution consisted in testing all the 816,975,224 possible alternatives of the search space, and then obtaining the nondominated solutions. The exact global Pareto front is formed by 221 different solutions. The example was also resolved by the proposed method and, in order to evaluate the algorithm behavior, it was replicated 30 times. The parameters for the multiobjective GA are exposed in Table 3.

Table 3 – Multiobjective GA parameters for example 1.

Parameter	**Value**
Population size	100
Number of generations	200
Probability of crossover	0.95
Number of variables for crossover	7
BLX parameter	1
Probability of mutation	0.01

The execution of the algorithm resulted in 30 Pareto fronts that were compared to the real front by a distance metric. Firstly, for each point in an obtained front, it was calculated the minimum euclidean distance from it to one of the 221 points in the real front. Then, with the purpose of finding a mean distance representing an entire front ($\overline{d_i}$, i = 1,…,30), say the ith front, all the minimum distances were summed up and divided by ns_i (number of obtained solutions in the ith front, i = 1,…,30). Next, it was calculated the following weighted mean:

$$D = \frac{\sum_i \overline{d_i} \cdot ns_i}{\sum_i ns_i}, \qquad i = 1,\ldots,30 \tag{14}$$

that attempts to summarize the convergence of the obtained front in one single number. However, note that in this procedure there is loss of information since the solution is the entire front and not only one single point. Table 4 presents $\overline{d_i}$, the variance of the minimum distances (var_i) and ns_i for each one of the 30 Pareto fronts. The metric D was $1.30 \cdot 10^{-3}$.

Table 4 – Mean and variance of minimum distances for 30 obtained Pareto fronts.

i	$\overline{d_i}$	var_i	ns_i	i	$\overline{d_i}$	var_i	ns_i
1	$1.43 \cdot 10^{-3}$	$1.11 \cdot 10^{-4}$	80	16	$1.35 \cdot 10^{-3}$	$2.93 \cdot 10^{-5}$	72
2	$2.29 \cdot 10^{-3}$	$1.62 \cdot 10^{-4}$	93	17	$1.73 \cdot 10^{-6}$	$5.27 \cdot 10^{-11}$	76
3	$2.66 \cdot 10^{-4}$	$3.49 \cdot 10^{-6}$	83	18	$4.77 \cdot 10^{-4}$	$9.32 \cdot 10^{-6}$	78
4	$3.71 \cdot 10^{-6}$	$7.38 \cdot 10^{-10}$	74	19	$8.19 \cdot 10^{-6}$	$1.82 \cdot 10^{-10}$	71
5	$3.50 \cdot 10^{-4}$	$7.63 \cdot 10^{-6}$	90	20	$1.64 \cdot 10^{-3}$	$1.00 \cdot 10^{-4}$	90
6	$1.52 \cdot 10^{-5}$	$1.17 \cdot 10^{-8}$	89	21	$2.18 \cdot 10^{-6}$	$1.81 \cdot 10^{-11}$	64
7	$1.04 \cdot 10^{-5}$	$1.21 \cdot 10^{-9}$	75	22	$2.52 \cdot 10^{-3}$	$9.19 \cdot 10^{-5}$	87
8	$1.29 \cdot 10^{-3}$	$7.63 \cdot 10^{-5}$	92	23	$2.24 \cdot 10^{-6}$	$2.34 \cdot 10^{-11}$	67
9	$1.58 \cdot 10^{-6}$	$1.22 \cdot 10^{-11}$	76	24	$1.18 \cdot 10^{-2}$	$1.08 \cdot 10^{-3}$	89
10	$8.15 \cdot 10^{-4}$	$4.65 \cdot 10^{-5}$	70	25	$7.56 \cdot 10^{-4}$	$1.55 \cdot 10^{-5}$	94
11	$3.21 \cdot 10^{-4}$	$7.23 \cdot 10^{-6}$	71	26	$1.99 \cdot 10^{-3}$	$1.31 \cdot 10^{-4}$	84
12	$9.60 \cdot 10^{-6}$	$1.89 \cdot 10^{-10}$	80	27	$3.13 \cdot 10^{-6}$	$5.27 \cdot 10^{-11}$	72
13	$9.38 \cdot 10^{-4}$	$1.66 \cdot 10^{-5}$	81	28	$1.36 \cdot 10^{-5}$	$7.24 \cdot 10^{-9}$	89
14	$2.15 \cdot 10^{-6}$	$1.42 \cdot 10^{-11}$	76	29	$1.29 \cdot 10^{-6}$	$9.31 \cdot 10^{-12}$	66
15	$1.57 \cdot 10^{-3}$	$1.31 \cdot 10^{-4}$	75	30	$6.04 \cdot 10^{-3}$	$3.00 \cdot 10^{-4}$	90

In addition, it can be observed from Table 4 that $\overline{d_{24}}$ and $\overline{d_{29}}$ are, respectively, the highest and the lowest mean distance. Figure 7 shows the true Pareto front and the obtained fronts 24 and 29. In particular for the front 29, 49 of its 66 solutions were from the true Pareto front. Note that the proposed multiobjective GA is able to find solutions on and near the real Pareto front. The system designs represented by the selected solutions in Figure 7 are pictured in Figure 8.

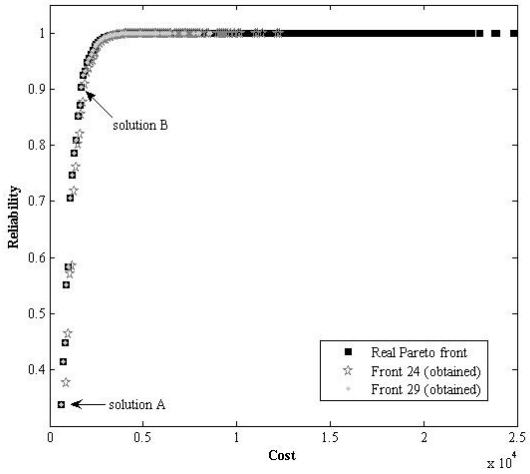

Figure 7 – Real Pareto front and estimated Pareto fronts.

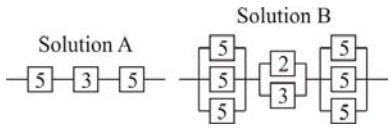

Figure 8 – Selected solutions from Pareto front in Figure 7.

7. Example Application 2: The Case of Repairable Systems

Consider now a repairable series-parallel system comprised of components that undergo operating/failure/repair cycles. Moreover, the components times between failures are assumed to follow Weibull distributions, whereas the times to repair are exponentially distributed. Under these more realistic assumptions, the availability of each candidate system is estimated via discrete event simulation, and the integrated multiobjective GA+DES discussed in the Section 5 is used to tackle this problem.

The system is supposed to have four subsystems in series S_1, S_2, S_3 and S_4. Each one of them must have at least 1 component. Table 5 shows the maximum number and the quantity of alternative components for each subsystem. In this example, the search space is constituted of 726,750 possible combinations among components.

An example of phenotype for an individual, which is a vector formed by 14 decision variables, is pictured in Figure 9.

Table 5 – Subsystems features.

	S_1	S_2	S_3	S_4
Maximum number of components	3	2	5	3
Number of alternative components	4	3	4	3

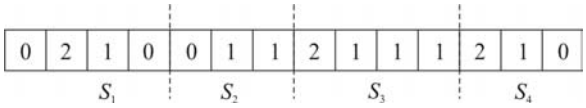

Figure 9 – Example of an individual phenotype (example 2).

The two considered objectives are the system mean availability maximization and the system cost minimization. However, the corrective maintenance cost is also taken into account along with the acquisition cost in the estimation of the total system cost. The acquisition cost is defined as follows:

$$C_A = \sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}^{a} x_{ij}\,, \tag{15}$$

where c_{ij}^{a} is the acquisition cost of one unit of the *j*th component of subsystem *i*. The corrective maintenance cost is:

$$C_{CM} = \sum_{i=1}^{s}\sum_{j=1}^{m_i}\sum_{k=1}^{x_{ij}} c_{ij}^{cm} n_{ijk}\,, \tag{16}$$

where n_{ijk} is the mean number of repairs of the *k*th unit of the *j*th component type of the *i*th subsystem during simulation time and c_{ij}^{cm} is the repair cost of a component of type *j* of *i*th subsystem. Therefore, the system total cost is defined as follows:

$$C_S = \sum_{i=1}^{s}\sum_{j=1}^{m_i}\left[c_{ij}^{a} x_{ij} + \sum_{k=1}^{x_{ij}} c_{ij}^{cm} n_{ijk} \right] \tag{17}$$

Table 6 presents the Weibull distributions (α is the scale parameter and β is the shape parameter) for the components times between failures, and the Exponential distributions (with parameter λ) for the repair times. Table 6 also lists the acquisition and corrective maintenance costs of every possible component in each subsystem. The multiobjective GA parameters are listed in Table 7.

After running the coupled multiobjective GA+DES algorithm for a mission time of 100 time units, 55 nondominated solutions were obtained (Figure 10). Systems corresponding to the solutions indicated in the Pareto front in Figure 10 are illustrated in Figure 11. Solutions A and E (extreme solutions) have, respectively, mean availabilities equal to 0.5625 and 0.9860, with associated costs of 3,262.87 and 10,862.25 monetary units.

Table 6 – Components features.

	Component	$f_X(x)$	$f_D(d)$	c_{ij}^a	c_{ij}^{cm}
S_1	1	Weibull(25; 1.5)	Exp(0.20)	730	55
	2	Weibull(27; 1.7)	Exp(0.30)	650	45
	3	Weibull(40; 1.4)	Exp(0.20)	1000	85
	4	Weibull(35; 1.2)	Exp(0.50)	850	78
S_2	1	Weibull(25; 1.4)	Exp(0.25)	700	50
	2	Weibull(28; 1.3)	Exp(0.20)	760	65
	3	Weibull(24; 1.2)	Exp(0.20)	650	70
S_3	1	Weibull(20; 1.2)	Exp(0.30)	670	60
	2	Weibull(40; 1.8)	Exp(0.20)	1100	90
	3	Weibull(32; 1.6)	Exp(0.50)	1215	95
	4	Weibull(29; 1.3)	Exp(0.20)	950	90
S_4	1	Weibull(33; 1.6)	Exp(0.20)	920	87
	2	Weibull(31; 1.3)	Exp(0.20)	870	76
	3	Weibull(19; 1.2)	Exp(0.40)	400	35

Table 7 – Multiobjective GA parameters for example application 2.

Parameter	**Value**
Population size	50
Number of generations	100
Probability of crossover	0.95
Number of variables for crossover	7
BLX parameter	1
Probability of mutation	0.01

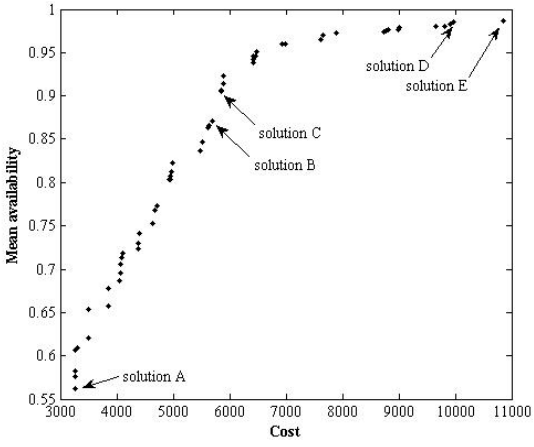

Figure 10 – Results for example application 2.

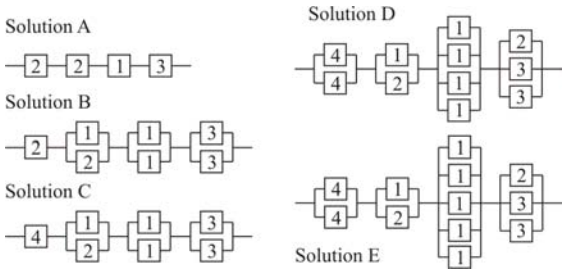

Figure 11 – Selected solutions from the obtained Pareto front (example application 2).

7.1 Return of Investment Analysis

Since all nondominated solutions are equivalent according to the multiobjective approach, the performance of these solutions can be investigated by means of a return of investment analysis. This analysis consists in verifying the gain in availability in relation to the required investment in system design:

$$ROI = (A_i - A_j) / (C_i - C_j) \text{ , } i \neq j \quad (18)$$

where i and j are solutions in the Pareto front, and A_i, A_j, C_i and C_j are their respective availabilities and costs. It is desired a large value in the numerator of equation (18) and a small difference $C_i - C_j$ in order to obtain a large *ROI*.

Consider, for instance, solutions B, C, D and E (Figure 10). Table 8 shows their corresponding availability, cost and *ROI*. From Table 8, note that for an increase in system availability of 0.0366, it is required 164.67 monetary units, whereas for improving system availability from 0.9852 to 0.9860, it is necessary to invest 897.80 monetary units on system design. Therefore, the gain in availability is sometimes very small in relation to the required investment. In addition, the *ROI* in the first case is about 249 times the *ROI* in the second case.

Table 8 – Return of investment of some solutions of example 2.

Solution	Availability	Cost	*ROI*
B	0.8705	5,688.75	$2.22 \cdot 10^{-4}$
C	0.9071	5,853.42	
D	0.9852	9,964.45	$8.91 \cdot 10^{-7}$
E	0.9860	10,862.25	

8. Concluding Remarks

This paper proposed a multiobjective genetic algorithm for the treatment of redundancy allocation problems involving repairable systems under the conflicting objectives of maximizing system mean availability and minimizing system cost (acquisition and corrective maintenance costs). The work attempted to overcome some of the limitations in the system reliability modeling. In fact, it was considered a series-parallel topology comprised by components supposed to have failure-repair cycles modeled by renewal processes with Weibull distributed times between failures and Exponential repair times. Under these

conditions, the resulting system's failure-repair cycles are not characterized by a renewal process and therefore no analytical solution to system availability is provided. Moreover, it was also considered multiple alternative components for a given slot in a subsystem with different reliability and cost characteristics. For these more realistic assumptions regarding the system dynamic behavior, system availability was estimated through DES. The integration between the multiobjective GA with DES takes place at the fitness evaluation step, *i.e.*, the availability is estimated via DES for every individual in a given generation of the evolution process in the GA algorithm. The DES permits the incorporation of real world features in problem modeling, but also demands a considerable computation effort. In this way, its application is indicated when the analysis of the system dynamic behavior is mandatory and/or when there is no analytical characterization of the problem under consideration.

The proposed approach was illustrated by means of two application examples. In the first one, a simple problem consisting of a non-repairable system with time independent failure probabilities was considered in order to validate the multiobjective GA. The results showed that the proposed algorithm, even being quite straightforward, was able to find solutions near and on the true Pareto front. According to the explained distance metric, even the furthest encountered front is very close to the real one. The second example consisted in a more realistic situation regarding the system dynamic behavior. The coupled multiobjective GA with discrete event simulation was used to obtain the Pareto front corresponding to the maximization of system mean availability and minimization of system cost. The results from both application examples indicate that the proposed method can be a valuable tool for the decision maker particularly in situations where the true Pareto front is unknown. In this way, one can obtain *good* solutions even if those solutions are not the optimal ones.

However, the assumption of perfect repair might be unrealistic for some problems of practical interest. In this context, the integrated multiobjective GA+DES can be extended to deal with systems and components subjected to imperfect repairs. Therefore, the redundancy allocation problem can be addressed for situations, for example, where components failure-repair processes are modeled via generalized renewal processes (Moura *et al.*, 2007). Moreover, other maintenance policies such as preventive maintenance and inspection with resource restrictions such as spare parts and maintenance crew availability can be also taken into account by extending the proposed method. Besides, additional improvements in the multiobjective GA (such as the coupling with local search algorithms) can enhance its performance and these are current research topics by the authors.

References

(1) Banks, J.; Carson, J.S.; Nelson, B.L. & Nicol, D.M. (2001). *Discrete event system simulation*. 3ed. Prentice Hall, Upper Saddle River.

(2) Busacca, P.G.; Marseguerra, M. & Zio, E. (2001). Multiobjective optimization by genetic algorithms: application to safety systems. *Reliability Engineering & System Safety*, **72**, 59-74.

(3) Cantoni, M.; Marseguerra, M. & Zio, E. (2000). Genetic algorithms and Monte Carlo simulation for optimal plant design. *Reliability Engineering & System Safety*, **68**, 29-38.

(4) Chiang, C.-H. & Chen, L.-H. (2007). Availability allocation and multiobjective optimization for parallel-series systems. *European Journal of Operational Research*, **180**, 1231-1244.

(5) Coello, C.A.C.; Veldhuizen, D.A.V. & Lamont, G.B. (2002). *Evolutionary algorithms for solving multiobjective problems*. Kluwer Academic, New York.

(6) Deb, K. (1999). Evolutionary algorithms for multicriterion optimization in engineering design. **In**: *Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN'99)*.

(7) Deb, K.; Pratrap, A.; Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182-197.

(8) Doyen, L. & Gaudoin, O. (2004). Classes of imperfect repair models based on reduction of failure intensity or virtual age. *Reliability Engineering & System Safety*, **84**, 45-56.

(9) Droguett, E.L. & Mosleh, A. (2006). Análise Bayesiana da confiabilidade de produtos em desenvolvimento. *Gestão & Produção*, **13**, 57-69.

(10) Elegbede, C. & Adjallah, K. (2003). Availability allocation to repairable systems with genetic algorithms: a multiobjective formulation. *Reliability Engineering & System Safety*, **82**, 319-330.

(11) Fonseca, C.M. & Fleming, P.J. (1993). Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. **In**: *Proceedings of the Fifth International Conference on Genetic Algorithms*.

(12) Goldbarg, M.C.; Goldbarg, E.F.G. & Medeiros Neto, F.D. de (2005). Algoritmos evolucionários na determinação da configuração de custo mínimo de sistemas de co-geração de energia com base no gás natural. *Pesquisa Operacional*, **25**, 231-259.

(13) Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading.

(14) Herrera, F.; Lozano, M. & Verdegay, J.L. (1998). Tackling real-coded genetic algorithms: operators and tools for behavioral analysis. *Artificial Intelligence Review*, **12**, 265-319.

(15) Horn, J.; Nafpliotis, N. & Goldberg, D.E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. **In**: *Proceedings of the First IEEE Conference on Evolutionary Computation*.

(16) Joines, J.A. & Houck, C.R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. **In**: *Proceedings of the First IEEE International Conference on Evolutionary Computation.*

(17) Kirkpatrick, S.; Gelatt Jr., C.D. & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, **220**, 671-680.

(18) Levitin, G. (2005). *The universal generating function in reliability analysis and optimization*. Springer, London.

(19) Marseguerra, M. & Zio, E. (2000). Optimizing maintenance and repair policies via a combination of genetic algorithms and Monte Carlo simulation. *Reliability Engineering & System Safety*, **68**, 69-83.

(20) Marseguerra, M.; Zio, E. & Martorell, S. (2006). Basics of genetic algorithms optimization for RAMS applications. *Reliability Engineering & System Safety*, **91**, 977-991.

(21) Marseguerra, M.; Zio, E. & Podofillini, L. (2005). Multiobjective spare part allocation by means of genetic algorithms and Monte Carlo simulation. *Reliability Engineering & System Safety*, **87**, 325-335.

(22) Martorell, S.; Carlos, S.; Sánchez, A. & Serradell, V. (2000). Constrained optimization of test intervals using steady state genetic algorithm. *Reliability Engineering & System Safety*, **67**, 215-232.

(23) Messac, A.; Sundararaj, G.J.; Tappeta, R.V. & Renaud, J.E. (2000). Ability of objective functions to generate points on nonconvex Pareto frontiers. *AIAA Journal*, **38**, 1084-1091.

(24) Michalewicz, Z. (1996). *Genetic algorithms + data structures = evolution programs*. 3ed. Springer, Berlin.

(25) Moura, M.J.C.; Rocha, S.P.V.; Droguett, E.A.L. & Jacinto, C.M.C. (2007). Bayesian assessment of maintenance efficiency via generalized renewal process. In portuguese. *Pesquisa Operacional*, **27**, 569-589.

(26) Rausand, M. & Hoyland, A. (2004). *System reliability theory: models and statistical methods*. 2ed. John Wiley & Sons, New York.

(27) Rauzy, A. (2001). Mathematical foundations of minimal cutsets. *IEEE Transactions on Reliability*, **50**, 389-396.

(28) Rigdon, S.E. & Basu, A.P. (2000). *Statistical methods for the reliability of repairable systems*. John Wiley & Sons, New York.

(29) Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. **In**: *Proceedings of the First International Conference on Genetic Algorithms*.

(30) Srinivas, N.K. & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Journal of Evolutionary Computation*, **2**, 221-248.

(31) Taboada, H.; Baheranwala, F.; Coit, D. & Wattanapongsakorn, N. (2007). Practical solutions for multi-objective optimization: an application to system reliability design problems. *Reliability Engineering & System Safety*, **92**, 314-322.

(32) Taboada, H. & Coit, D.W. (2006). MOEA-DAP: a new multiple objective evolutionary algorithm for solving design allocation problems. Under review. *Reliability Engineering & System Safety*.

(33) Taboada, H.; Espiritu, J. & Coit, D.W. (2008). MOMS-GA: a multiobjective multi-state genetic algorithm for system reliability optimization design problems. *IEEE Transactions on Reliability*, **57**, 182-191.

(34) Zhang, Q. & Li, H. (2007). MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, **11**, 712-731.

(35) Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods and applications. PhD Thesis, Swiss Federal Institute of Technology, Zurich.

(36) Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257-271.