Availability Allocation to Repairable Systems with Genetic Algorithms: A Multi-Objective Formulation

Charles Elegbede, Kondo Hloindo Adjallah

Availability Allocation to Repairable Systems with Genetic Algorithms: A Multi-Objective Formulation

Charles Elegbede* & Kondo Adjallah

Laboratoire d'Optimisation des Systèmes Industriels (LOSI)
Université de Technologie de Troyes (UTT)
12, rue Marie Curie, BP 2060 10010 Troyes (France)

(*) Corresponding address: EADS SPACE Transportation
BP 11 - 33165 Saint Medard en Jalles (France)
charles.elegbede@launchers.eads.net
kondo.adjallah@utt.fr

ABSTRACT: This paper describes a methodology based on genetic algorithms (GA) and experiments plan to optimize the availability and the cost of reparable parallel-series systems. It is a NP-hard problem of multi-objective combinatorial optimization, modeled with continuous and discrete variables. By using the weighting technique, the problem is transformed into a single-objective optimization problem whose constraints are then relaxed by the exterior penalty technique. We then propose a search of solution through GA, whose parameters are adjusted using experiments plan technique. A numerical example is used to assess the method.

Key words: availability allocation, multi-objective optimization, genetic algorithms.

This work has been done when (*) was in LOSI laboratory of Université de Technologie de Troyes (France).

NOTATIONS

A_{obj} : Availability target

C_{max} : maximum cost admissible for the system

s : number of subsystems

i : index of a subsystem

λ_i : failure rate of components in subsystems i

$\boldsymbol{\lambda}$: $\boldsymbol{\lambda} = (\lambda_1, \lambda_2,...,\lambda_s)$

μ_i : repair rate of components in subsystems i

$\boldsymbol{\mu}$: $\boldsymbol{\mu} = (\mu_1, \mu_2,...,\mu_s)$

k_i : number of components in subsystem i

$\boldsymbol{k}$: $\boldsymbol{k} = (k_1, k_2,..., k_s)$

$\boldsymbol{l_k}$, $\boldsymbol{u_k}$: vector of minimum and maximum number of components allowable in each subsystem

$\boldsymbol{l_\lambda}$, $\boldsymbol{u_\lambda}$: vector of lower and upper bound of failure rate of components in each subsystem

$\boldsymbol{l_\mu}$, $\boldsymbol{u_\mu}$: vector of lower and upper bound of repair rate of components in each subsystem

a_i, b_i, p_i, q_i : coefficients of the cost function

$\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{p}$, $\boldsymbol{q}$: vector of a_i, b_i, p_i, q_i

$C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k})$: cost of the system

$A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k})$: asymptotic availability of the system

ω_1, ω_2 : weighting factors respectively for the cost and the availability

δ_1, δ_2 : relaxation coefficients respectively for the cost and the availability

Θ_i : sensitivity factors (i = 1,...,4)

$\lfloor x \rfloor$: integer part of x such that $\lfloor x \rfloor \leq x$

Ngene : number of generation

Tpop : populations size

p_m : mutation probability

p_c : crossover probability

s.t . : subject to.

1. INTRODUCTION

Let us recall first of all that if the reliability of a system decreases toward 0 when its operating time tends to infinity (i.e. becomes very large), then its availability decreases to a value called asymptotic availability [1]. The main requirements for complex systems operating are usually specified in terms of cost and availability and/or of reliability, or equivalently in terms of mean operating time and/or of mean down time under cost constraint. These requirements have then to be taken into consideration in the system design stage in order to determine the appropriate reliability and availability of each of its components. The requirements, generally expressed by the owner of the system, correspond to a global dependability objective, which must be apportioned to each component of the system by the designer [3]. The result can then be used to conceive an appropriate logistic to support the system (determination of optimal stock of spare parts, development of an appropriate maintenance plan, etc.).

The problem of availability allocation can be formulated as a multi-objective optimization problem: minimize the cost and maximize the availability of the global system. As constraints relating to both objectives, there are the system asymptotic availability must be smaller than the target availability A_{obj} on the one hand, and the cost of the system must be lower than a given maximum cost $C_{\max}$ on the other hand.

The decision variables of this problem are the number of components k_i in each subsystem, the failure rate λ_i and the repair rate μ_i of each component (i = 1, ..., s), which are also used to compute the cost and the availability of the system [4]. The objective and constraint functions of

the problem are non-linear, and in addition they depend on continuous and discrete variables (respectively the failure and repair rates, and the number of components). The considered optimization problem was proven NP-hard [29]. Therefore, an heuristic or a meta-heuristic such as genetic algorithms can provide solutions close to the optimal solution with good efficiencies [31]. Indeed, optimization based on GA is known to be an efficient approach to solve classical problems of multi-objectives optimization under constraints ([5], [6], [30], [35] , [35]). Several papers relating to systems reliability optimization using GA was published these last few years ([7], [8], [9], [10], [10], [10]). For all those reasons, we chose genetic algorithms to solve the optimization problem stated in section 2.

In our approach, without loss of generality, we will consider only the case of parallel-series systems. Using appropriate objective functions and fitness functions, this approach can be generalized to other categories of system. More details on the use of GA for reliability optimization can be found in the state-of-the-art of Gen & Kim [11] or the one proposed by Kuo & Prasad [31]. To solve this problem, weights are used to transform the multi-objective functions into a single objective function. Then, we relax the constraints using an exterior penalty approach [12] to lead finally to a problem without constraint. The new objective function is then used as adaptation function (fitness function) of the GA for which we design an experiment plan to tune the parameters. Lastly, the methodology is applied to solve the availability allocation problem under cost constraint, for a repairable parallel-series system.

Section 2 of the paper states the multi-objective optimization problem. Section 3 presents the method used to relax the constraints, and gives details on the cost function and the new formulation of the problem. Then, details on the implemented GA are presented in section 4. Next, we propose an experiment plan approach to tune the parameters of the algorithm in section

5. It enables to assess the sensitivity of each parameter on the GA effectiveness. Finally, section 6 summarizes the main results obtained with the approach proposed to solve the specific optimization problem.

2. STATEMENT AND FORMULATION OF THE PROBLEM

2.1 General model of the problem

Let us recall that a parallel-series system is a system made of parallel subsystems put in serial (figure 1). A parallel subsystem works when at least one of its components works and a series system fails when at least one of its components fails. We suppose in all the remainder of the paper that the subsystems are parallel, and that in each subsystem all the components are identical. Here, the "identical" term means that the components have the same reliability and availability. The number of components in each subsystem, the availability of the system and the system cost will be denoted respectively by k_i, A_S *and* C_S. Assume now that during the design stage of a parallel-series system, the availability objective A_{obj} has to be apportioned to each component, with the aim of maximizing the availability and of minimizing simultaneously the total cost of the system. We then have a multi-objective optimization problem that can be formulated as follows:

$$\left.\begin{array}{l} \underset{\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}}{Maximize}\ A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k})\ \ and\ \ \underset{\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}}{Minimize}\ C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \\ \underline{s.t.}\ A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \geq A_{obj}\ ;\ C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \leq C_{\max} \\ \ell_{\lambda i} \leq \lambda_i \leq u_{\lambda i}\ ;\quad \ell_{\mu i} \leq \mu_i \leq u_{\mu i}\ and\ \ell_{ki} \leq k_i \leq u_{ki} \end{array}\right\} (i = 1,...,s) \qquad (1)$$

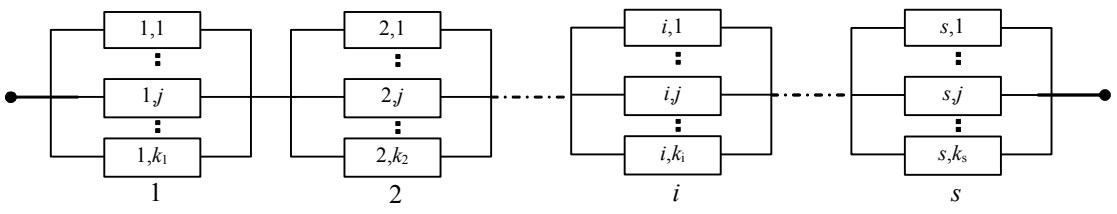

Figure 1: General structure of a parallel-series system.

For the parallel-series systems, the asymptotic availability and the cost are given by:

$$A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}) = \prod_{i=1}^{s}\left(1-\left(\frac{\lambda_i}{\lambda_i+\mu_i}\right)^{k_i}\right) \tag{2}$$

$$C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}) = \sum_{i=1}^{s} k_i\left(a_i\lambda_i^{p_i}+b_i\mu_i^{q_i}\right) \tag{3}$$

where a_i, b_i, p_i, and q_i are real numbers such that : $a_i > 0$, $b_i > 0$, $p_i < 0$, $q_i > 0$ ($i = 1,...,s$).

The decision variables of the problem modeled by equation (1) are λ_i, μ_i, k_i ($i = 1,...,s$). Since maximizing the availability A_S is equivalent to minimizing the unavailability (1-A_S), the problem can be solved through a vector minimization. In addition, the following observations about the cost model could help to formulate the problem.

2.2 The cost model

Equation (3) describes the cost model of the system. The cost of the system is obtained by the sum of the costs of all the components, which are supposed to be identical for a given subsystem. It then remains to choose the suitable models for the cost of components. The approach used in this paper consists in providing, for a given component, a cost model that depends on the main features enabling to describe the technical efficiency of the system. This technical efficiency is expressed in term of availability which depends on components failure and repair rates. Thus, in the formulation of a given component cost, there are two main contributions:

- the failure rate λ_i contribution : $a_i\lambda_i^{p_i}$, where $a_i > 0$, $p_i < 0$, ($i = 1,...,s$) are real numbers, which means that the more the component failure rate is low the more the component will be expensive.

- the repair rate μ_i contribution : $b_i \mu_i^{q_i}$, where $b_i > 0$, $q_i > 0$, $(i = 1,...,s)$ are real numbers, which means that the lower the repair rate of the component is, the less expensive the component will be.

In practice, the coefficients a_i , b_i, p_i , q_i $(i = 1,...,s)$ may be obtained from a maintenance data base. If one assumes that $(\boldsymbol{\lambda}^{(j)}, \boldsymbol{\mu}^{(j)}, \mathbf{k}^{(j)}, C_S^{(j)})$ $(j=1,...,h)$ are known for a set of h configurations of the system to design (where $C_S^{(j)}$ is the cost of the jth configuration), the coefficients may be obtained by minimizing the quantity:

$$W(\mathbf{a},\mathbf{b},\mathbf{p},\mathbf{q}) = \sum_{j=1}^{h} \left(C_S\left(\mathbf{a},\mathbf{b},\mathbf{p},\mathbf{q} \middle| \boldsymbol{\lambda}^{(j)}, \boldsymbol{\mu}^{(j)}, \mathbf{k}^{(j)}\right) - C_S^{(j)} \right)^2$$

ie. by using the least square regression:

$$\left.\begin{aligned} \frac{\partial W(\boldsymbol{a},\boldsymbol{b},\boldsymbol{p},\boldsymbol{q})}{\partial a_i} &= 0 \\ \frac{\partial W(\boldsymbol{a},\boldsymbol{b},\boldsymbol{p},\boldsymbol{q})}{\partial b_i} &= 0 \\ \frac{\partial W(\boldsymbol{a},\boldsymbol{b},\boldsymbol{p},\boldsymbol{q})}{\partial p_i} &= 0 \\ \frac{\partial W(\boldsymbol{a},\boldsymbol{b},\boldsymbol{p},\boldsymbol{q})}{\partial q_i} &= 0 \end{aligned}\right\} \quad (i = 1, \ldots, s) \qquad (4)$$

This set of 4s independent equations and 4s unknown have a solution [12]. There are various ways in engineering to model the cost of a component. For instance, Misra et *al.* [32] considered in their approach that the cost of a component may be modeled as a continuous function of the reliability. Recently, Giuggioli et *al.* [24] also used continuous functions to express the contribution of the reliability of the components to the cost of the system. In this cost model, they took into account the profit from the system operation, purchase and installation costs, the repair costs, and the penalty related to the downtime due to miss contract delivery of service. The interest of their model is to highlight its bonds with the intention of the designer. For example, the downtime of a component may not cause a penalty if one considers that in case of redundancy the service continues to be delivered. On the contrary, Levitin et *al.* (1998) [25] consider that it is more realistic to suppose that there is no analytic function that expresses the dependence of cost on its nominal performance. They consider that for the nominal performance, the supplier offers reliability and price characterizing each system or component version. Many

others continuous functions are presented in [2]. But let us say, in general, that the cost model depends on the studied problem. Hence, only an analysis of the problem can lead to a relevant formulation of the cost model.

To our opinion, a solution could be found to the difficulties pointed out by Levitin et *al.* by building a cost function that depends on each version of component involved in the system. However, it is clear that there is no general solution to the problem and a tradeoff should be sought.

In our paper, we choose to model the cost function by equation (3) taking only into account the parameters (λ_i and μ_i), which enables to calculate the performance of the system expressed in terms of asymptotic availability.

3. CONSTRAINTS RELAXATION AND NEW FORMULATION OF THE PROBLEM

3.1 Multi-objective optimization

In single-objective optimization problem, the optimum corresponds to the point that maximizes or minimizes the objective function in the feasible domain ([12], [13], [14]). But in multi-objective optimization, the notion of optimal solution is replaced by the notion of Pareto optimal solution, *i.e.* a non-dominated solution ([14], [33], [34], [35]).

Assume that a vector $\boldsymbol{Z}(\boldsymbol{x})$ has to be maximized *i.e.* $\boldsymbol{Z}(\boldsymbol{x}) = (Z_1(\boldsymbol{x}),\ldots,Z_p(\boldsymbol{x}))$, where $\boldsymbol{x} \in \boldsymbol{IR}^n$, with m constraints $g_i(\boldsymbol{x})$:

$$\text{Maximize } \boldsymbol{Z}(\boldsymbol{x}) \text{ s.t. } \quad g_i(\boldsymbol{x}) \leq 0 \quad (i = 1, \ldots m) \tag{5}$$

a) Definition of a feasible solution

A solution $\boldsymbol{x}^*$ is a feasible solution of the optimization problem (5) , if and only if it satisfies all the constraints of the problem. The set of feasible solutions is called “feasible domain”.

b) *Definition* [36]

A solution $\boldsymbol{x}^*$ in the feasible domain is Pareto-optimal or non dominated solution, if there is no solution $\boldsymbol{x}^{**}$ in the feasible domain such that $Z_j(\boldsymbol{x}^{**}) \geq Z_j(\boldsymbol{x}^*)$, $j = 1,\ldots, p$, and $Z_l(\boldsymbol{x}^{**}) > Z_l(\boldsymbol{x}^*)$ for at least one $l \in [1, p]$.

A feasible solution $\boldsymbol{x}^*$ is weakly Pareto-Optimal if there is no feasible solution $\boldsymbol{x}^{**}$ such that $Z_j(\boldsymbol{x}^{**}) > Z_j(\boldsymbol{x}^*)\, j \in [1, p]$.

c) Generation of a Pareto optimal solution

There are several approaches to solve a multi-objective optimization problem to obtain a Pareto optimal solution ([14], [33], [34], [35], [36]). The existing methods can be classified in two categories:

- the first one gathers the methods which are based on the preference of a decision maker. It consists of approaches based on the use of weightings, the constraint-based approach, or distances based approach [36]. The output of these methods is a single non-dominated solution, corresponding to the preference expressed by a decision maker for example. To obtain a set of Pareto-optimal solution, it is sufficient to consider a set of preferences, and to run the algorithm for each expressed preference.

- the second one gathers methods using genetic algorithms, the main one are the Vector Evaluation Genetic Algorithms developed by Shaffer [36] and the Non dominated Sorting Genetic Algorithms (NSGA) developed by Srinivas et *al.* [34]. The output of these two methods is a set of Pareto-Optimal solutions.

Let us present briefly the first category of methods.

Weighting-based approach **[14][30][34]**

Consider $\omega_j > 0, j = 1,\dots, p$. The weighting approach consists in solving the following single-objective optimization problem:

$$Maximize \sum_{j=1}^{p} \omega_j Z_j(\boldsymbol{x}) \quad \underline{s.t.} \quad g_i(\boldsymbol{x}) \leq 0 \quad (i = 1, \dots m) \tag{6}$$

Consequently, the solution of (6) is also a solution of (5). The coefficients ω_j are arbitrarily chosen for this method, and depend on the decision maker preference.

Constraints-based approach **[30] [34]**

To solve equation (5) by the constraints-based approach, one needs to choose one component of the vector to be optimized as objective and some appropriate coefficients h_j (according to the decision maker preference) and solve the following system (7), whose solution is also solution of (5):

$$\left.\begin{array}{l} \textbf{Maximize } Z_k(\boldsymbol{x}) \\ s.t. \underset{i=1,\cdots,m}{g_i(x) \leq 0}; \quad \underset{j=1,\cdots,p}{Z_j(x) \geq h_j} \end{array}\right\} \tag{7}$$

Approach based on distance function **[30][34]**

This method consists of minimizing the objective chosen in the vector to optimize, compared to the specified target vector $\tilde{z}$ of the problem. The multi-objective problem scalarization is:

$$Minimise \left[\sum_{i=1}^{p} \left(Z_i(\boldsymbol{x}) - \tilde{z}_i \right)^r \right]^{1/r} \tag{8}$$

Where $r \in \mathbf{N}\text{-}\{0\}$. The solution depends on the specified vector $\tilde{z}$. The case $r = 2$ corresponds to the classical Euclidian metric.

Other specific methods used for genetic algorithms

In the literature, there are some specific multi-objectives optimization methods based on genetic algorithms. The Vectors Optimization Genetic Algorithm (VEGA) for instance modifies the classical genetic algorithms by performing independent selection cycles according to each objective [36], and provides a set of Pareto optimal solutions.

Equally, the Non-dominated Sorting Genetic Algorithms (NSGA) approach [34] provides a set of Pareto optimal solutions.

3.2 Multi-objective optimization and constraints relaxation

In this paper, we choose to use the weighting-based approach, more easy to implement.

a) Weighting coefficients

To a given set of weights (ω_1, ω_2) corresponds a Pareto optimal solution. Hence, if one wants to propose several choices to the decision maker, it will suffice to run the algorithm described in this paper the number of time it is necessary to obtain a set of Pareto optimal solutions. Indeed, the choice of weighting coefficients vector expresses a preference on the objective to be optimized. This approach may be time consumer with respect to the VEGA and NSGA [34] to perform the Pareto optimal search. But when the decision maker is sure on his preference (*i.e.* it does not need any other alternative), the simple weighting approach may be sufficient with a set of coefficients.

By choosing the appropriate values for $\omega_1 > 0$ and $\omega_2 > 0$ according to the preference of the decision maker, the optimization problem could be stated as follows:

$$\left.\begin{array}{l}\underset{\lambda,\mu,k}{\text{Minimize}}\ \omega_1\left(1-As\left(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}\right)\right)+\omega_2 Cs\left(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}\right) \\ \underline{s.t.}\ As\left(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}\right)\geq A_{obj};\ Cs\left(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}\right)\leq C_{\max} \\ \ell_{\lambda i}\leq\lambda_i\leq u_{\lambda i};\ \ell_{\mu i}\leq\mu_i\leq u_{\mu i}\ \text{and}\ \ell_{ki}\leq k_i\leq u_{ki}\end{array}\right\}\ i=1,\ldots,s \qquad (9)$$

where $\ell_{\lambda i}$, $\ell_{\mu i}$ and ℓ_{ki} respectively denote the lower bound of the failure rate, of the repair rate and of the number of components in a subsystem, and $u_{\lambda i}$, $u_{\mu i}$ and u_{ki} respectively denote the upper bound of the same variables.

On the basis of equation (9), the problem will be transformed into a single-objective optimization problem that can be solved by constraints relaxation.

b) Penalty coefficients for constraints relaxation

The exterior penalty methods are generally used to relax the constraints of a problem [7][26][27], and enable to solve an equivalent optimization problem without constraints. The choice of penalty coefficients takes into account the preference of the decision maker with respect to the constraints violation. Penalty coefficients are chosen according to practical implication of constraints violation. For example, the penalty on the cost will be large if the cost violation is less acceptable than the availability violation.

Coit et *al.* 1996 [38] used adaptive coefficients to relax the constraints which are difficult to satisfy. They introduced the notion of "non feasible threshold solutions penalty".

In order to compare a solution with the same evaluation function, the penalty coefficients are chosen before the beginning of the simulation and are kept constant throughout the simulation. Thus, from a given generation *i* to *i*+1 (*i* = 1 to *Ngene*), the evaluation function which depends on weighting coefficients and penalty coefficients that will remain the same during all simulation enable to re-formulate the problem as follows:

$$\left.\begin{array}{l} \underset{\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}}{Minimize}\, F(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \\ \underline{s.t.}\quad \ell_{\lambda i} \le \lambda_i \le u_{\lambda i};\quad \ell_{\mu i} \le \mu_i \le u_{\mu i};\quad \ell_{ki} \le k_i \le u_{ki} \end{array}\right\}\ i = 1, \cdots, s \qquad (10)$$

where δ_1 and δ_2 are the penalty coefficients, and where the function $F(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{k})$ is defined as:

$$\begin{aligned} F(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) &= \omega_1\left(1 - A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k})\right) + \omega_2 C_S\left((\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k})\right) \\ &+ \delta_1 \max\left(A_{obj} - A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}); 0\right) + \delta_2 \max\left(C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) - C_{\max}; 0\right) \end{aligned} \qquad (11)$$

A solution of (10) is also solution of (9) [14]. Thus, the search will use equation (10) which corresponds to a single-objective optimization problem.

c) The evaluation function used in the genetic algorithm

The new objective function $F(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{k})$ is bounded by a value Q such that for all λ_i, μ_i, k_i (i = 1, …, s), $Q - F(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}) > 0$. To prove this, it is sufficient to prove that the cost has an upper bound over the search space. Let us observe that if $u_k = \sup_{i=1,..,s}(u_{ki})$, then the following relation holds:

$$C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \le u_k \sum_{i=1}^{s}\left(a_i \lambda_i^{p_i} + b_i \mu_i^{q_i}\right) \qquad (12)$$

In addition, according to Heine theorem [16], as the function $\lambda_i \mapsto a_i \lambda_i^{p_i}$ is continuous on a closed and bounded interval $[\ell_{\lambda i}, u_{\lambda i}]$, it is bounded and reaches its upper and lower bounds on the interval. Here, we just need the fact that the function is upper-bounded. Let $M_i \in \boldsymbol{IR}^+$ be the upper bound, then the relation $\sum_{i=1}^{s} a_i \lambda_i^{p_i} \le s \cdot \sup_{i=1,\ldots,s}(M_i)$ holds. According to the continuity and the monotony of the function $\lambda_i \mapsto a_i \lambda_i^{p_i}$, the upper bound can be calculated as $M_i = a_i (\ell_{\lambda i})^{p_i}$.

Similarly, it can be proved that there exits a value $P_i \in \boldsymbol{IR}^+$, $P_i = b_i (\ell_{\mu i})^{q_i}$, such that $\sum_{i=1}^{s} b_i \mu_i^{q_i} \le s \cdot \sup_{i=1,\ldots,s}(P_i)$.

Therefore, one can conclude that $C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \le s \cdot u_k \left(\sup_{i=1,\ldots,s}(M_i) + \sup_{i=1,\ldots,s}(P_i) \right)$.

Let us denote: $$Q_1 = s \cdot u_k \left(\sup_{i=1,\ldots,s}(M_i) + \sup_{i=1,\ldots,s}(P_i) \right) \quad (13)$$

Given that $0 < A_{obj} < 1$, $0 < A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) < 1$, $C_{\max} > 0$, and $\max\left(A_{obj} - A_S(\lambda_i,\mu_i,k_i);0\right) < 1$,

then $\max\left(C_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) - C_{\max};0\right) < Q_1$, because $C_{\max} > 0$ and $\max\left(A_{obj} - A_S(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k});0\right) < 1$.

Taking into account these properties, we can determine an upper bound of the objective function in equation (11) as: $F(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \le \omega_1 + \omega_2 Q_1 + \delta_1 Q_1 + \delta_2$.

Hence, for any positive real ε, there exists a value Q such that $Q - F(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) > 0$, defined by

$Q = (1+\varepsilon)\cdot\left(\omega_1 + \delta_2 + (\omega_2 + \delta_1) Q_1\right)$. We chose $\varepsilon = 0$ in this paper.

Let $H(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k}) = Q - F(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{k})$, then the optimization problem can be stated as follows:

$$\left.\begin{array}{c} \underset{\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}}{Maximize} \quad H(\boldsymbol{\lambda},\boldsymbol{\mu},\mathbf{k}) \\ \underline{s.t.} \quad \ell_{\lambda i} \le \lambda_i \le u_{\lambda i}; \quad \ell_{\mu i} \le \mu_i \le u_{\mu i}; \quad \ell_{ki} \le k_i \le u_{ki} \\ i = 1,\cdots,s \end{array}\right\} \quad (14)$$

By construction, the solutions of (14) satisfy the constraints related to the search domain [14]. Note that H is the scaling form of the objective function also called evaluation function because through this function, the decision maker preferences are taken into account from two points of view: 1) to obtain a single objective optimization problem, 2) to relax the constraints of the problem.

The quality of the results of GA depends strongly on its adaptation function. The function H defined in this section is the adaptation function used in the proposed GA. H is also called fitness function or evaluation function.

4. PRINCIPLE OF GENETIC ALGORITHMS

Genetic algorithms are introduced by John Holland at Michigan University in the late sixties [6]. Genetic algorithms belong to evolutionary algorithms and are devoted to simulate animals reproduction by using the Darwinian theory [17] which stipulates that only individuals that are strong may survive and will be able to reproduce. Then naturally the weak individuals will be eliminated.

There are many papers that give details on this family of algorithm. The reader may refer to [6][18][26] for more details on the topic. A set of recent PhD dissertations [30], [33] on the topic shows that genetic algorithms are very efficient to solve NP-hard problems.

The general schema of genetic algorithms may be summed up as follows (figure 2). First of all, an initial population (generation) of potential solutions (individuals) is generated randomly. One makes also evolve this population by recombination of individuals by the mean of crossover operator and/or mutation operator. A selection procedure based on a fitness function (also called evaluation function or adaptation function) enables to choose the individuals candidate for reproduction. The reproduction consists in recombining two individuals by the crossover operator, possibly followed by a mutation of the offspring. Therefore, from the initial population a new generation is obtained. From this new generation, a second new generation is produced by the same process and so on. The stop criterion is generally based on the number of generations.

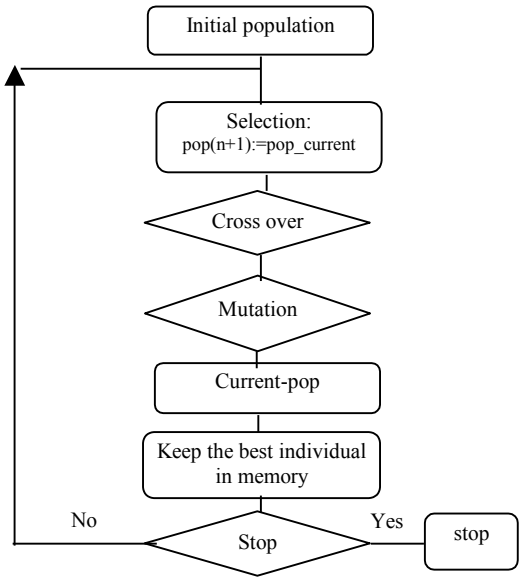

Figure 2: Flowchart representing the GA's principle.

In the remainder of this section, the details of the items of the genetic algorithm we have implemented are presented with appropriate comments.

4.1 Encoding of the solution

A solution of equation (1) represents a chromosome which can be coded as a matrix of dimension 3×*s*. The first row of the matrix contains the failure rates $\lambda_i \in [\ell_{\lambda i}, u_{\lambda i}]$ of the components of the system. The second row contains the repair rates $\mu_i \in [\ell_{\mu i}, u_{\mu i}]$ and the third row contains the necessary number $k_i \in [\ell_{ki}, u_{ki}]$ of components in each subsystem such that the availability of the system reaches the target A_{obj}. This encoding appears to be simpler, in comparison to the binary encoding where the computing times are large without significant improvement in the search process [5].

The encoding proposed is phenotypic because the genes of the chromosomes are directly chosen in the variables space. One could have used a binary string to encode the chromosome of

solutions; in this case the gene would be a string of bits that would be decoded to obtain the solution in the variables space. This latter approach is a genotypic one.

4.2 Generating the initial population

The generation of an initial population is necessary to start solving the optimization problem with a GA. This process is based on the construction of initial solutions or individuals chromosomes by sampling randomly values of λ_i, μ_i and k_i in their appropriated interval. The size of any population is given and remains the same at each generation. The main difficulty in the initial population is that the individuals may not satisfy all or part of the constraints of the problem. A GA must improve the individuals of the populations throughout the successive generations, so as to tend towards the optimal solution.

4.3 Selection of the individuals

The selection principle is based on Darwin theory [17], which postulates that in a natural system, only strong individuals will survive. Selecting individuals who are prone to reproduction improves the new successive generations of individuals. Thus, the fitness of each individual of the population is evaluated in order to select the best individual who is in fact the individual that has the highest fitness. The mainly known selection processes are the roulette technique and the tournament technique. The first one may be consulted in [5] [18]. We shall present only the tournament technique used in this paper.

The tournament technique

The tournament technique consists in choosing M individuals at random ($M < n$), according to uniform law in (0,1) in a population and in retaining the m best individuals among M ($m < M$),

who will be inserted in the new generation that is being formed ([5], [20]) for the crossover. Generally, one chooses $m = 1$ ([5], [20]).

In this paper, we use the tournament technique for selecting the individuals in the current population. The fitness of the individuals is measured by means of the adaptation function H defined in section 3.2, which corresponds to the relaxed form of the penalized objective function.

4.4 Crossover operators

There are several crossover operators in the literature [18]. The crossover must enable to obtain offspring whose chromosomes represent children who have kept the good properties of the parents. But it is not always possible to satisfy the latter crossover properties, and one defines crossover just to obtain new individuals that will be evaluated by the fitness function. Crossover operators must be adapted to the studied problem. The most current operators are the arithmetic crossover [5] [18]) and the crossover with one or several crossing points [20]. We shall present only the one point crossover, which is used in our paper.

Crossover with one point

Let $[f_1 f_2 \ldots f_n]$ and $[t_1 t_2 \ldots t_n]$ be two strings encoding two potential solutions of a problem. Let us choose randomly a value k in the set $\{1,\ldots,n\}$ following an uniform law. The crossover with one crossing point consists in creating two offspring as follows: $f_1 f_2 \ldots f_k t_{k+1} \ldots t_n$ and $t_1 t_2 \ldots t_k f_{k+1} \ldots f_n$. This crossover is the one used in our approach and simple to implement. It will be applied to each row of the chromosomes at the same position randomly chosen in each chromosome.

4.5 Mutation operators

This operator is very relevant, for it enables to insert new genes in the chromosomes [6][18][27]. This can be seen as local reparation on the chromosome and thus brings new genes into the chromosomes of the individuals in order to increase diversity in the population. The mutation occurrence probability should be very low if one wants to ensure stability in the neighborhood of the current solution found by the algorithm. In our approach, the mutation consists in reducing the failure and the repair rates of 5% on 20% of the genes of the chromosomes. In addition, the number of components is increased by 1 in the corresponding genes. The values 5% and 20% are chosen because they appeared to be appropriate, but they could also be adjusted for instance by an experiment plan. But according to the importance of these parameters in the algorithm, we consider that it is not necessary to implement such a technique.

4.6 Immigration technique

In absence of sufficient diversity in the population, the algorithm can quickly converge towards a weak solution. This behavior is due to the heuristic character of the GA. To avoid such situation, we decide to change all or part of individuals of the current population when the best individual remains the same after a given number of successive generations. This process is called immigration [40]. In our case, it consists in removing randomly 50% of the individuals from the current population and in replacing them by new individuals generated randomly in the search space. This is performed when during $\lfloor Ngene/4 \rfloor$ successive generations the algorithm has not improved the quality of the best individual, *i.e*. a new best individual is not obtained.

4.7 Tuning of the parameters of the algorithm

The described GA needs the tuning of several parameters. The main parameters are the size of populations, the number of generations, the crossover probability, the mutation probability, and the stop criterion. The last one consists in stopping the algorithm after a given number of successive generations without any improvement of the best solution. But tuning the other parameters, remains a difficult task. To overcome this difficulty, we propose to study the sensitivity of each parameter on the results and the algorithm efficiencies, by using the experiment plan described in section 5. This approach gives two indications for the parameters algorithm adjustment:

- the sensitivity of each parameter on the solution,
- the optimal set of parameters that will be used for the algorithm.

In 1986, Grefenstette [39] proposed to find the set of optimal parameters of genetic algorithms by means of efficiency criteria based on the objective function to be optimized (for unconstrained problems). He determined the set of optimal parameters by means of a meta GA to optimize the parameters of the GA to be designed. In our concerns, we not only take into account the objective to be optimized, but also the fact that the random process (seed) may have an effect on the simulation.

a) Population size

Usually, the number of individuals in the population which stands for the population size (*Pop*) depends on the problem to be solved. In the case where the initial population consists of feasible solutions, generally difficult to generate, we suggest to consider populations of small size. Indeed, in an optimization problem, find a feasible solution may be as difficult as to find the

optimal solution. Therefore, it is suggested to consider small size of populations, according to the computing time necessary to obtain such a solution, when the initial population is made up of feasible solutions. In the contrary case where the initial population is not made up of feasible solutions, it seems better to consider large size of populations. For our concern, we have generated non-feasible solutions and have relaxed the constraints.

b) Number of generations

The current population is called generation in GA. The number *Ngene* of generations must be fixed at the beginning of the algorithm by taking into account its convergence. This number may be large or not according to the problem under consideration. It is also one of the most used stop criteria. Let us observe that it is the computing time which will give a limitation to *Ngene*. Anyways, it is better to have largest size possible for *Ngene*. A good tradeoff must be found between the computing time and *Ngene*.

c) Crossover probability

The crossover probability (p_c) is used to decide which couples of individuals constituting the current population to be crossed, according to the random variable u sampled in the interval (0, 1): if $u < p_c$, then the couple of individuals corresponding to this pulling is crossed. The value of p_c is generally chosen bigger than 0.5, and is often considered equal to 0.6 [6]. In order to increase the possibility of obtaining good individuals among the populations generated during the successive improvements, we choose these two values for the crossover probability: 0.5 and 0.8 for the experiment plan.

d) Mutation probability

While referring to the natural systems, the mutation is a rare phenomenon, and occurs with a small probability. Thus, we adopted the usual values of the probability of mutation p_m [6]: p_m is generally chosen in the interval (0.03, 0.05).

4.8 General structure of the algorithm

1. Encode the solution;
2. Define the parameters of the algorithm: *Pop, pm, pc, Ngene;* m_0 := 0, d = ⌊Ngene/4⌋ /**d* immigration parameter */
3. Do $n := 0$
4. Create randomly the initial population $P(0)$
5. Select *Pop* individuals to be crossed, that constitute the population *PCO* of individuals to be crossed. /* *PCO* is an intermediate population*/
6. For $i = 1{:}2{:}Pop{-}1$;
   u_1 := rand(0,1);
   if $u_1 < pc$;
   cross $PCO(i)$ and $PCO(i{+}1)$; /* CH_1, CH_2 are returned */
   u_2:= rand(0,1);
   if $u_2 < p_c$
   CH_1 := mutate(CH_1); /* CH_1, CH_2 are the children */
   CH_2 := mutate(CH_2); /*mutate is the mutation operator */
   endif
   else
   $CH_1 := PCO(i)$; $CH_2 := PCO(i{+}1)$;
   endif;
   $PCT(i) := CH_1$; /* *PCT* current population */
   $PCT(i{+}1) := CH_2$;
   Endfor
7. $BEST(n)$:= best individual of *PCT* /**BEST* stores the best individuals */
8. If $BEST(n)$ is better than $BEST(m_0)$
   $m_0 = n$;
   endif
9. If $n{-}m_0 > d$;
   do immigration in *PCT*;
   endif
10. If $n < Ngene$
    $n := n{+}1$;

```
        goto 5;
    else
        stop;
    endif;
```

5. EXPERIMENT PLAN AND APPLICATION

Generally, researchers use arbitrary parameters values in the algorithms they implemented. For example, some authors consider that a crossover probability value may range from 0.6 to 0.8 and a mutation probability may range from 0.0 to 0.05. The size of population and number of generation are parameters that authors used to choose arbitrary without any particular investigation. It seems important to find the set of parameters that would provide the best results in terms of search efficiency. This set of (optimal) parameters may be found by using experiments plan to tune the parameters of the GA before running it with the set of optimal parameters. In this section, we present first of all the principle of experiments plan. Then, we present the application of this technique to help the tuning of the parameters of the GA. For that, we present the main elements from which the tuning is performed: the coding of the concerned parameters, the experimental matrix, the sensitivity measurement function of the efficiency of the algorithm and the result in term of choice of the parameters. The selected parameters are those used in the experiment that gives the best value of adaptation function. Finally, some comments related to the results are given.

5.1 The experiment plan

The tuning of a complex process controlled by several parameters is a very hard time consuming task in practice. Achieving an optimal tuning of the parameters of the process may be ensured through an experiment plan ([19], [20]), which is an optimization technique easy to implement. This technique, which principle is the following, is based on effectiveness evaluation

and sensitivity analysis of the process compared to the parameters. It aims at finding the right parameters through a small number of experiments. Let k be the number of parameters whose variation can affect the effectiveness of the process, *i.e.* k is the dimension of the vector of parameters also called factors. To reduce the number of experiments during the tuning of the parameters, let us consider discrete values for each parameter and let m be the possible number of discrete values for each parameter. Hence, there are m^k possible combinations of the values of, parameters enabling to evaluate the effectiveness of the process. This process can then be evaluated through m^k experiments. The evaluations will be based, in this experiment plan, on the input and output data of the process and on given criteria functions. Let us note that an experiment represents the running of the algorithm with a combination of the values of the parameters. For example, according to the experiment plan used, the experiment #1 will consist in running the algorithm with parameters (Pop, p_m, p_c, $Ngene$) = (100, 0.02, 0.5, 200).

5.2 Tuning the parameters of the GA

The process under consideration here is the GA, and we define two levels (m = 2) for each of its factors or parameters (-1 for a low value, +1 for a high value), as in table 1. Four parameters (k = 4) are concerned $\{p_c, p_m, Pop, Ngene\}$. The optimal combination of parameters will be investigated in a factorial plan of dimension 2^4, *i.e.* 16 experiments will be performed to determine the best values combination of the parameters. The experiment plan can be modeled by a matrix $M(i,j)$ in which the index of the rows corresponds to the experiment index, and index of the column corresponds to the index of the parameters. Let ψ be a parameter, $\psi \in (p_c, p_m, Pop\ Ngene)$. The matrix M defined as follows is summarized in table 2.

$$M(i,j) = \begin{cases} -1 & \text{if } \psi \text{ has low value} \\ +1 & \text{if } \psi \text{ has high value} \end{cases}$$

With this model, the evaluation of the sensitivity of the algorithm will be based on the following measurement functions with respect to each parameter:

$$\Theta_j = \left| \frac{1}{m_k} \sum_{i=1}^{m_k} M(i,j) H_i \right|, \quad j = 1, \ldots, k$$

The aim is to identify the best experiment, which is defined by the one that gives statistically the best results. The criteria that measure the best experiment will be the score S_i of each experiment after a given number of runs of the algorithm with the experiment plan. To obtain the score of an experiment, it must be run during a given number of times with a change of the seed of the random generator. The score of each experiment corresponds to the number of times it finds the best solution. Thus, the sensitivity analysis will be done by the use of the score S_i. The *ad hoc* sensitivity analysis is then given by:

$$\Theta_j = \left| \frac{1}{m^k} \sum_{i=1}^{m^k} M(i,j) S_i \right|, \quad j = 1, \ldots, k$$

The input data of the optimization problem are the following:

s= 5; A_{obj} = 0.999	***Cmax***= 600 units ***;*** $\boldsymbol{u_k}$= 9x(1 1 1 1 1)
ω_1 = 0.75; ω_2 =0.25	δ_1 = 100; δ_2 =10000
p =-0.8x(0.4, 0.2, 1, 0.8, 1.2)	***q*** = 0.85 x (0.4, 0.2, 1, 0.8, 1.2)
a = 0.01 x (4, 2, 5, 8, 12);	***b*** = 0.1 x (0.4, 0.2, 1, 0.8, 1.2)
$\boldsymbol{l_\lambda}$=10^{-3}x(0.4, 0.4, 0.5, 0.3, 0.5)	$\boldsymbol{u_\lambda}$= 2.10^{-3} x (1, 1, 1, 1, 1)
$\boldsymbol{l_\mu}$=0.5 x (0.4, 0.7, 0.9, 0.8, 0.7)	$\boldsymbol{u_\mu}$= 0.85 x (0.4, 0.7, 0.9, 0.8, 0.7)

The efficiency criterion according to experiment of index *i* is denoted H_i, which corresponds to the adaptation function defined in section 3.2. In this paper, the relevant criterion for the efficiency of the algorithm is the score S_i defined above. The sensitivity measurement increases with the influence of the corresponding parameter.

Table 1 describes the encoding of the parameters and Table 2 shows the matrix of the experiment plan for the efficiency evaluation of the GA.

Table 1: Factors of the experiment plan.

Parameters	-1	+1
$F_1 = P_c$	0.5	0.8
$F_2 = p_m$	0.02	0.05
$F_3 = Pop$	60	90
$F_4 = Ngene$	100	200

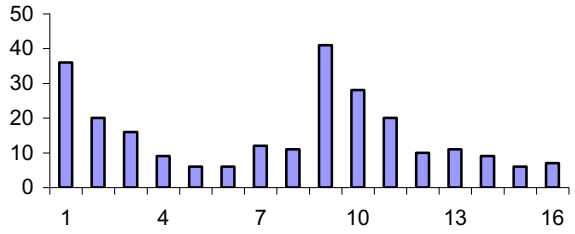

Figure 3: Experiments results.

Table 2: Experiments matrix and statistics on 250 runs.

N	F_1	F_2	F_3	F_4	*250 runs*
1	-1	-1	-1	-1	38
2	+1	-1	-1	-1	20
3	-1	+1	-1	-1	16
4	+1	+1	-1	-1	9
5	-1	-1	+1	-1	6
6	+1	-1	+1	-1	6
7	-1	+1	+1	-1	12
8	+1	+1	+1	-1	11
9	**-1**	**-1**	**-1**	**+1**	**41***
10	+1	-1	-1	+1	28
11	-1	+1	-1	+1	20
12	+1	+1	-1	+1	10
13	-1	-1	+1	+1	11
14	+1	-1	+1	+1	9
15	-1	+1	+1	+1	6
16	+1	+1	+1	+1	7

Table 3: sensitivity measurement for 250 runs.

Θ_j	Θ_1	Θ_2	Θ_3	Θ_4
250 runs	3,125	4,25	7,125	0,875

5.3 Results and comments

The adaptation function defined in section 3.2 is considered as an efficiency measurement function for the individuals of the GA. After 250 runs of the GA, the experiment #9 gives the best results according to the experiment plan (see Table 2). Indeed, this experiment has obtained the best score (number of maximum values of the adaptation function). The parameters of the experiment #9 are selected for implementing the GA to find the solution of the optimization problem. This helps to choose the parameters of the GA which enable to reach an optimal

solution. In addition, the sensitivity measurement of this experiment plan enables to identify the most sensitive parameters of the algorithm: *Pop* and p_m (factors 3 and 2). In addition, it is interesting to notice that the factor 1 (P_c) is also important (see Table 3).

5.4 Optimal search with GA

The stability of the algorithm with respect to the parameters and even to the structure of the cost function constitutes a problem of thorough investigation. But, with regard to our optimization problem, the method which is suggested for tuning the parameter of the algorithm remains statistically acceptable. By using the retained parameters in the case studied and referring to the experiment plan technique, experiment # 9 corresponding to factors level (-1, -1, -1, +1) gives the solution depicted in table 4 for our problem. For a sufficiently exhaustive search of solution and to ensure that the solution is close to the optimal solution, it is essential to increase the size of the population (300 for example) on the one hand. It can be also necessary to increase the number of factors considered in the implementation of the experiment plan on the other hand. But that orientation may be expensive in computing times and it thus becomes important to find very quickly a good compromise.

The system considered consists of 5 parallel subsystems in serial (cf. figure 1). The successive improvements of the best individual at each generation, during the search of the optimal solution with the GA are plotted in figure 4. The curve in this figure shows the evolution of the best fitness or the adaptation function from the first generation to the current generation:

$$H(i) = \max_{l=1,\ldots i,} \left(H_l\right), \text{ for } i = 1,\ldots, Ngene\,.$$

Table 4: Results obtained for the best solution

i	k_i	$\lambda_i(10^{-3}h^{-1})$	$\mu_i\,(h^{-1})$
1	3	0.0721	0.2281
2	3	0.1258	0.4349
3	1	0.1991	0.6148
4	3	0.1758	0.4713
5	1	0.1952	0.5872
As = 0.9993		C_S = 546.43 units	

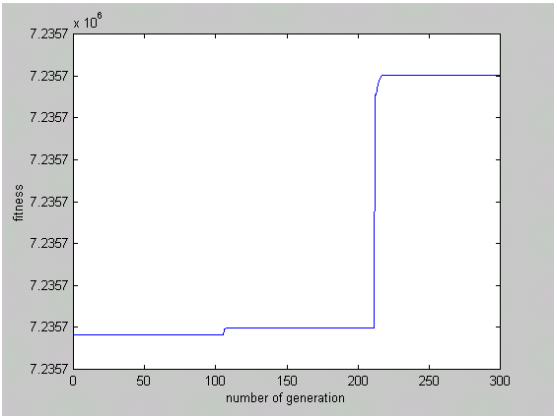

Figure 4: Max Fitness from the first generation

Table 4 summarizes the optimization results, with the failure and repair rate and the number of components allocated to each subsystem. One notes that the cost of the system is also less than C_{Max}.

6. CONCLUSION

In this paper, a GA-based approach is proposed to solve a multi-objective optimization problem. It simultaneously aims at maximizing the availability and minimizing cost of repairable parallel-series systems. In a first step, we used the weighting technique to transform the problem into a problem of single objective optimization. Next we relaxed the constraints by mean of the technique of exact penalty to reformulate the problem and to find a solution. The objective function obtained is then used for adaptation in a GA. In a second step, we developed the GA for which we have tried to provide guidance to adjust the parameters and control the fitness. For that purpose, we designed an experiment plan that shows its relevance for the GA implementation. The use of experiment plans could also be suitable to solve the optimization problems with simulated annealing, where there are several parameters to adjust. The methodology can be structured in two steps as follows: 1) working out of an experiment plan for the tuning of the parameter of the algorithm; 2) selection of the best parameters and implementation of the GA for

the search of an optimal solution. Numerical experiments enable to illustrate the effectiveness of the approach. Even if GA do not give the optimal solution, it remains a practical way to search a solution complying at best with the problem constraints.

Acknowledgements: The authors indebted to anonymous referees for their invaluable comments and suggestions.

REFERENCES

[1] Pagès, A., & M.,Gondran, (1980). *Fiabilité des systèmes*. Ed. Eyrolles, Paris.

[2] Tillman F, C.-L. Hwang, W. Kuo (1985). Optimization of systems reliability. *Ed.Marcel Dekker*.

[3] Elegbede C., F. Yalaoui, K. H. Adjallah, C. Chu, (1999). Allocation de fiabilité par les algorithmes génétiques et le recuit simulé. Congrès International Pluridisciplinaire, Qualité et Sûreté de Fonctionnement. Paris, France, 25-26 Mars, pp. 475-483.

[4] Villemeur, A., (1988). *Sûreté de fonctionnement des systèmes industriels*. Ed. Eyrolles, Paris.

[5] Michalewicz, Z., (1996). Genetics algorithms + data structures = Evolution programs. Ed. Springer-Verlag. New York.

[6] Goldberg, D., (1994). *Les algorithmes génétiques*. Ed. Adisson–Wesley, Paris.

[7] Coit, D., & E. Smith, (1996). Reliability optimization of series parallel systems using genetic algorithm. IEEE Transactions on Reliability, R45(2) pp. 254-260.

[8] Campbell, J., & L., Painton (1995) Genetic algorithm in optimization of system reliability. IEEE Transactions on Reliability, R44(2) pp. 172-178.

[9] Elegbede, C. & K. Adjallah, (1999). Reliability allocation to components following Weibull law using genetics algorithms. ESREL'99, European Safety and reliability Conference, Munich, Germany, 13-17 September, pp. 999-1004**.**

[10] Elegbede, C. A. O. (2000). *Contribution aux méthodes d'allocation d'exigences de fiabilité aux composants de systemes*. Ph D dissertation, Université de Technologie de Compiègne.

[11] Gen, M. & J. R., Kim, (1999). GA-based reliability design: state-of-the-art survey. Proc. of Computers & Industrial Engineering, 37(1-2), pp. 151-155.

[12] Minoux, M., (1983). *Programmation mathématique.* Vol. 1, Ed. Dunod, Paris.

[13] Culioli, J.-C., (1994). *Introduction à l'optimization*. Ed. Ellipses, Paris.

[14] Cohon J. L., (1978). Multi-objective programming and planning. Academic Press. Californie.

[15] Dhingra, A., (1992) Optimal apportionment of reliability in Series systems under multiple objectives. IEEE transactions on reliability, vol. 41, n° 4, 1992 December.

[16] Valiron, G., (1948). *Théorie des fonctions*. Ed. Masson et Cie, Paris.

[17] Darwin, C., (1992). *L'origine des espèces*. Ed. GF-Flamarion, Paris.

[18] Bäck, T. (1996). *Evolutionnary algorithms in theory and practice*. Oxford University Press.

[19] Goupy, J. (1998). *La méthode des plans d'expériences.* Ed. Dunod, Paris.

[20] G. Sado & M-C. Sado, G. (1991). *Les plans d'expériences*. Ed. Afnor, Paris.

[21] Dey, A. & R. Mukerjee (1999). *Fractional, Factorial Plans.* Ed. Willey, in Series of Probability and Statistics.

[22] Myers, H. R. & DC Montgomery, (1995). *Response surface methodology: process and product optimization using designed experiments*. Ed. John Willey & Sons, Inc. New York.

[23] Yang, J-E., M.-J Hwang., T-Y Sung, Y. Jin (2000). Application of genetic algorithm for reliability allocation in nuclear power plants. *Reliability Engineering and System Safety* 65 pp-229-238

[24] Giuggioli Busacca, P., M. Marsenguera and E. Zio (2001). Multiobjective optimization by genetic algorithms : application to safety systems. *Reliability Engineering and System Safety* 72 pp-59-74

[25] Levitin G., A Lisnianski, H. Ben-Haim, D. Elmakis (1998). Redundancy optimization for series-parallel multi-state systems. *IEEE transactions on reliability, Vol. 47, NO. 2,* pp 165-172.

[26] Schoenauer M. and Z. Michalewicz. Evolutionary Computation. *Control and Cybernetics* 26 (3) pp 307-338.

[27] Michalewicz, Z., D. Daguspta, R. G. Le Riche, and M. Schoenauer (1996). Evolutionnary Algorithms for constrained Engineering Problems. Computer engineering and industrial journal. Vol. 30 (4), pp. 851-870.

[28] Lin C.-Y. and P. Hajela (1992). Genetic algorithms in optimization problems with discrete and integer design variables. *Eng. Opt.,* vol. pp 309-327.

[29] Chern, M.-S (1992). On the computational complexity of reliability redundancy allocation series system. *Operations Research Letters* 11, pp 309-315.

[30] Coello, C. A. C (1996). An empirical study of evolutionary techniques for multiobjective optimization in engineering design. *An abstract submitted on the 4th day of 1996 partial fulfillment of the requirements for the degree of Doctor of philosophy*.

[31] Kuo, W and R. Prasad (2000). An annotated overview of system-reliability optimization. *IEEE transactions on reliability*. Vol. 49 NO. 2, pp 176-187.

[32] Misra K. B. and M. D. Ljubojecvic (1973). Optimal reliability design of a system: a new look. *IEEE transactions on reliability*. Vol. R-22, No 5, pp 255-258.

[33] Eckart Z (1999). Evolutionary algorithms for Multiobjective Optimization Methods and applications. *PhD dissertation. Shaker Verlag Aachen*.

[34] Srinivas N., and K. Deb (1995). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation* 2 (3), pp 221-248.

[35] Eckart Z (1999). Multiobjective Algorithms: A comparative case study and the strength Pareto Approach. *IEEE transactions on evolutionary computation*, vol. 3 No. 4, pp. 257-271.

[36] Fonseca C., and P. J. Fleming(1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation* 3 (1), pp 1-16.

[37] Spear W (1998). Role of mutation and recombination in evolutionary algorithms. PhD dissertation. George Mason University, Fairfax, Virginia.

[38] Coit D.W. A. Smith (1996). Penalty guided genetic search for reliability design optimization. Computers and Eng. Vol.?? No 30, pp 895-904.

[39] Grefenstette, J. J. (1986). Optimization of control parameters for genetics algorithms. Transactions on systems, man, and cybernetics. Vol. 1, pp 122-128.

[40] Lin C.-Y., and P. Hajela, (1992). Genetic algorithms optimization problems with discrete and integer design variables. Eng. Opt.. Vol. 19, pp. 309-327.

AUTHORS:

Charles ELEGBEDE is PhD of Compiègne University of Technology (UTC) in Systems dependability since January 2000, his PhD topic is on reliability allocation on complexes systems. He is graduated in Civil Engineering in 1996 by the **C/U/S/T** of Clermont-Ferrand II University (France). He received the same year a DEA in Material Science, Mechanical and Civil engineering Structures Reliability. He was member of the Laboratory of Industrial Systems Optimization (LOSI) of Troyes University of Technology. His research activities are reliability analysis of dynamics systems and fuzzy modeling of systems reliability. He is member of IMdR-SDF (National Institute of Dependability of France). He rejoined **E**uropean Aeronautical **D**efense and **S**pace Company in **S**pace **T**ransportation division (**EADS-ST**) since July 2000, in the department of Analysis and Justification in reliability office.

Kondo Hloindo ADJALLAH obtained his M.Sc. in metrology, automatic control and electrical engineering of Université of Nancy 1 (France) in 1989, followed by PhD in automatic control and electrical engineering of the Institut National Polytechnique de Lorraine (France) in 1993. His PhD related to the diagnosis of the dynamic systems and was led at the Centre de Recherche automatically of Nancy. From 1993 to 1994, he was assistant professor at Université de Nancy 1. He is now an associated professor and obtained the full professor habilitation (HDR) since 2003 at the Université de Technologie de Troyes (UTT) where he jointly conducts researcher and lecturer activities. He is currently responsible of the engineer training program in maintenance and diagnosis within the department of Industrial System Engineering at the UTT.

Member of the laboratory of Modeling and Systems Safety (LM2S) of 1994 to 2000 at the UTT, he was at the origin of the project of the RAMS requirements allocation to the systems during design. He joined the laboratory of Industrial Systems Optimization of the (LOSI) since

April 2000 for interested in the problems of operations research and optimization relating to the systems' availability. His applications relate to the modeling of the maintenance of the distributed systems, and the planning and management of the diagnosis and maintenance activities.

He is member of the French National Association of Maintenance Engineers, member of the French Academic Working Group on “Modeling and Optimization of the Distributed & Collaborative Maintenance” affiliated to CNRS, and also member of the European Safety and Reliability Data Association (ESReDA).