2011-01-01

# An Evolutionary Approach Based On Viral Replication For Solving Combinatorial Optimization Problems

Claudia Evangelina Valles Sosa
*University of Texas at El Paso*, cevalles@miners.utep.edu

# AN EVOLUTIONARY APPROACH BASED ON VIRAL REPLICATION FOR SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS

CLAUDIA EVANGELINA VALLES SOSA

Department of Industrial, Manufacturing &System Engineering

APPROVED:

Heidi A. Taboada Jimenez , Ph.D., Chair

Jose F. Espiritu Nolasco Ph.D.

Connie Gomez, Ph.D.

Benjamin C. Flores, Ph.D.
Acting Dean of the Graduate School

# Dedication

To my son, Julio Cesar.

# AN EVOLUTIONARY APPROACH BASED ON VIRAL REPLICATION FOR SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS

by

CLAUDIA EVANGELINA VALLES SOSA, BS IN IE

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Industrial, Manufacturing, & System Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

August 2011

# Abstract

A new multi-objective evolutionary algorithm that mimics the way viruses replicate is presented. The purpose of viral replication is to allow the reproduction and survival of its kind. The algorithm is able to produce a large number of solutions to be evaluated by imitating the way viral replication and the infection process work. The well-known multi-objective redundancy allocation problem is used to show the performance of this new viral-based evolutionary approach. The algorithm considers the maximization of system reliability, the minimization of system cost, and the minimization of system weight to be optimized simultaneously. The solution to the multi-objective optimization problem is a set of Pareto-optimal solutions.

# Table of Contents

# List of Tables

# List of Figures

# List of Illustrations

# Chapter 1: Introduction

A complex reliability design problem is the well-known Redundancy Allocation Problem (RAP). Realistic formulations of the RAP are characterized by large combinatorial search space with multiple or single constraints and objectives. The Redundancy Allocation Problem basically refers to the process of adding redundant components to improve the reliability of the system meeting time constraints of cost, weight, volume, or any other. This problem has previously been solved using many different mathematical and metaheuristic optimization approaches.

This thesis presents a new approach to solve the multi-objective redundancy allocation problem with an innovative optimization method called the Viral System Algorithm (VSA). The idea of solving the RAP as a multi-objective optimization problem is because almost all the optimization problems in real life involve more than one objective to be optimized and normally those objectives are in conflict to each other. To the best of our knowledge, this is the first research that involves the application of the VS to solve a multi-objective combinatorial optimization problem.

The Viral System is a recently developed bio-inspired optimization approach created in the Seville University, by Cortes et al. (2008). VSA is a new approach to solve optimization problems, which makes use of a biological analogy inspired by the performance of viruses. The replication mechanism, as well as the hosts' infection processes, is used to generate a meta-heuristic that helps obtain an optimal solution. To test the performance of the VSA we used the multiple-objective redundancy allocation problem from the paper presented by Taboaba & Coit (2007) that involves the maximization of reliability, the minimization of cost, and weight. The solution to this multi-objective problem is presented in a set of Pareto-optimal solutions and compared with the Pareto-optimal set obtained using NSGA-II from the selected paper.

A different approach, which is more correlated with the VSA, is the Artificial Immune System (AIS). This algorithm is based in the principles and processes of the vertebrate immune system. The AIS became one of the foundations of the VSA because; is also a bio-inspired algorithm: dealing with immune system behavior. These algorithms are based in the response of the organism to a pathogen. AIS procedures are inspired by the response of the body when infected by viruses or bacteria. In contrast VSA procedures follow the process of a virus to reach the objective to successfully multiply and kill the organism in which it lies. As a result, many of their procedures are similar but each has a different focus in the algorithm's evolution and diversity. AIS monitors the success of the immune system defeating a pathogen and VSA monitors the progression and success of a virus over a host.

The research objective of this thesis is the development of a new algorithm to solve combinatorial optimization problems, inspired by the virus process of infecting bacterium. To reach our objective we need to model the Viral System Algorithm and adapt it to solve multi-objective optimization problems. This was accomplished by coding the VSA algorithm in Matlab to solve the problem used to check the performance of the algorithm proposed.

The content of the thesis contains the following: introduction (current section), chapter 2 which describes the multiple-objective optimization principles and methods which is divided in mathematical and metaheuristic approaches. The most common mathematical approaches are presented: goal programming, weighted sum method, lexicographic method, and multi-attribute utility theory. Following the metaheuristic approaches such as ant colony optimization, particle swarm optimization, artificial immune system algorithm and genetic algorithm are presented. Chapter 3 explains the Redundancy Allocation Problem (RAP) in detail and the optimization approaches that had been developed to solve it as single and multi-objective problem. In Chapter 4 we present the Viral System Algorithm (VSA). First, we explained how a virus works in nature; namely the replication process and creation of antibodies. Then the principles and components of the Viral System algorithm are presented.

Finally the steps of the Viral System Algorithm are explained. In Chapter 5, the Viral System Algorithm's steps are applied to a small Multi-Objective Redundancy Allocation Problem. In Chapter 6 the Viral System Algorithm was tested using the well-known Redundancy allocation problem, considering multi-objective. Finally, Chapter 7 presents the conclusions and some important points that can be considering as part of future research.

# Chapter 2: Multi-Objective Optimization

Multi-objective optimization (MOO) can be defined as the process of optimizing, systematically and simultaneously, a collection of objective functions. In the real world many search and optimization engines are formulated as non-linear programming problems having multiple objectives. In the past, due to the lack of suitable solution techniques, MOO problems were solved as a single-objective optimization problem. Such compromise created a conflict between the single solution point, achieved for one of the objectives, and the others that deteriorated as they did not get the main attention as they were not properly addressed (Marler & Aurora 2009).

There exist a number of fundamental differences between the working principles of single- and multi-objective optimization techniques. In single objective optimization problem, the task is to find one solution which optimizes the sole objective function. Therefore, if the problem is extended to solve for multiple objectives, finding a solution that optimizes each objective function may not possible.

Multi-objective optimization can create a set of solutions that attempt to solve multiple objective functions simultaneously. One of its main characteristics is that it gives rise to a set of trade-off optimal solutions (*Pareto-optimal* solutions), instead of a single optimum solution. As a result, multi-objective problems (MOP) need to find many optimal points to form a set of solutions called the Pareto Front. MOP usually requires repetitive applications of an algorithm to find multiple Pareto-optimal solutions, in some occasions such applications do not necessarily guarantee finding a certain Pareto-optimal solution (Ded 2001).

The general multi-objective optimization (MOO) problem can be formulated as follows:

$$\text{Minimize: } F(x) = [F_1(x), F_2(x), \dots., F_k(x)]^T \qquad (1)$$

$$\text{Subject to: } g_j(x) \leq 0; j = 1,2, \dots \dots, m$$

where $k$ is the number of objective functions and $m$ is the number of inequality constraints. $x \in E^n$ is a vector of design variables, and $F(x) \in E^k$ is a vector of objective functions $F_i(x)$ :

$E^n \rightarrow$ E1. The feasible design space is defined as X= {x $g_j$ (x) ≤0, j=1,2,…..,m}. The feasible criterion space is defined as Z={F(x) x € X}.

The relative importance of these objectives is not generally known until the system's best capabilities are determined and the tradeoffs between the objectives are fully understood; this depends greatly on the problem being analyzed. As the number of objectives increases, tradeoffs are likely to become more complex and less easily quantified. It usually evolves into NP-hard (non-deterministic polynomial-time hard) type of problems (Sawaragi et al. 1985).

**2.1 Mathematical Approaches**

**2.1.1Goal programming**

Goal programming can be thought as an extension or generalization of linear programming to handle multiple normally conflicting objective measures. Each goal has a value to be achieved and the target of the goal programming is to minimize the deviations from the goal. This can be a vector or a weighted sum dependent on the goal programming variant used. As the target becomes satisfactory by the decision makers, a sufficiently good enough philosophy is assumed.

The optimization problem is formulated as follows:

$$\begin{aligned} &\underset{\mathbf{x}\in\mathbf{X},\mathbf{d}^-,\mathbf{d}^+}{\text{Minimize}} \sum_{i=1}^{k}(d_i^+, d_i^-) \\ &\text{subject to} \quad F_j(x) + d_j^+ - d_j^- = b_j, \quad j = 1,2,.....,k \\ &\qquad d_j^+, d_j^- \geq, 0\, j = 1,2,....., k, \\ &\qquad d_j^+, d_j^- = 0,\ j = 1,2,....., k \end{aligned} \tag{2}$$

where $b_j$ are the goals of each objective function $F_j$(x), $d_j$ is the deviation from the goal $b_j$ for the *j*th objective, which is split into positive and negative parts. $d_i^+$ and $d_i^-$ represent underachievement and overachievement.

Goal programming (GP) was proposed by Charnes et al. (1955). The formulation used orders of unwanted deviations in priority level. Some variations to GP are archimedean goal programming, multigoal programming and goal attainment method. In the archimedean goal programming each deviation have a different weight depending in the perspective goal (Charnes & Cooper 1977). In multigoal programming, functions of $|d_j|$ are minimized as independent objective functions in a vector optimization problem (Zeleny 1982).

Gembicki (1974) proposed the goal attainment method, which entails using a modification of the weighted mix-max approach, but it has the inability to yield Pareto optimal solutions. As a result, Ogryczak (1994) created the reference goal programming. It entails using a modification of the weighted min-max method with an aspiration point rather than the utopia point. Major strengths of the goal programming are: its simplicity and ease of use, capability of handling large numbers of variables, constraints and objectives. In the other hand, weaknesses of goal programming are the ability to produce solutions that are not Pareto efficient.

### 2.1.2 Weighted sum method

Weighted sum model (WSM) or multi-criteria decision analysis / multi-criteria decision making method (MCDM) is used to evaluate a number of alternatives in terms of a number of decision criteria (Fishburn 1967). The way WSM works is by evaluating a set of alternatives where each alternative's total importance is calculated and then compared to the others to choose the one with the max weighted sum method - score, in the case of objective function as maximization. Expression (3) shows the formula used to calculate the total importance of alternative $A_i$, denoted as $A_i^{\text{WSM-score}}$. The weight of importance of the *j-th* criterion, $w_j$ and the performance value of alternative $A_i$, $a_{ij}$ are multiplied and then added the decision criteria:

$$A_I^{wsm-score} = \sum_{j=1}^{n} w_j a_{ij}, \ for \quad i = 1,2,3,...., m. \qquad (3)$$

where *m* represents the alternatives and *n* the decision criteria.

Or, commonly represented:

$$U = \sum_{i=1}^{k} w_i F_i(x). \qquad (4)$$

An example of how weighted sum model works is by representing the following matrix:

| | $\mathbf{C_1}$ | $\mathbf{C_2}$ | $\mathbf{C_3}$ |
|---|---|---|---|
| **Alts.** | 0.4 | 0.35 | 0.25 |
| $\mathbf{A_1}$ | 15 | 20 | 10 |
| $\mathbf{A_2}$ | 20 | 30 | 20 |
| $\mathbf{A_3}$ | 25 | 15 | 30 |

Where $C_x$ are the criteria with their respective weights, and $A_i$ are the alternatives. Taking in consideration the above matrix the previous formula is applied to see which alternative is better in case of maximization case:

$$A1^{\text{WSM-score}} = 15x.4+20x.35+10x.25=15.5$$
$$A2^{\text{WSM-score}} = 20x.4+30x.35+20x.25=23.5$$
$$A3^{\text{WSM-score}} = 25x.4+15x.35+30x.25=22.75$$

As result, the best alternative is A2 due to maximization objective. The result in ranking order is A2>A3>A1.

Many authors concerns are on difficulties with weighted sum method. First, a priori selection of weights does not necessarily guarantee that the final solution will be acceptable; one may have to resolve the problem with new weights (Messac 1996). The second difficulty is that it is impossible to obtain points on on-convex portions of the Pareto optimal set in the criterion space ( Das &Dennis 1997; Messac et al. 2000). Finally, the third difficulty is that varying the weights consistently and

continuously may not necessarily result in an even distribution of Pareto optimal points and accurate complete representation of the Pareto optimal set (Das & Dennis 1997).

### 2.1.3Lexicographic method

In the lexicographic method proposed by Fishburn (1974), the objective functions are arranged in order of importance and solve one by one. The following optimization problems are solved one at a time:

$$\begin{aligned} &\textit{Minimize} \quad F_i(x) \\ &\quad x \in X \\ &\textit{subject to} \quad F_j(x) \le F_j(x_j^*), \quad j = 1,2,....,i-1, i>1, \\ &i = 1,2,....,k. \end{aligned} \tag{5}$$

Where, *i* represents a function's position, and $F_j(x^*_j)$ is the optimum of the *j*th objective function, found in the *j*th iteration. The first iteration (j=1), $F_j(x^*_j)$ is not necessarily the same as the independent minimum of $F_j(x)$, because new constraints have been introduced (Stadler 1988).

Some variations where proposed to the lexicographic method: Osyczka (1984) proposed substituting the constraints in eq. (5) by the ones showed in eq. (6), which represents a constraint relaxation. This constraint's modification result, in generation of different Pareto optimal points. Another variation to the constraints formulation was proposed by Waltz (1967), $F_j(x) < F_j(x_j^*) + \delta_j$. With the variation of the constraints, the expansion of the feasible region is present, reducing the sensitivity of the final solution to the initial objective-function raking process.

$$F_J(x) \le (1 + \frac{\delta_j}{100}) F_J(x_j^*), \quad j = 1,2,....,i, \quad i>1. \tag{6}$$

Zikyna (2003) proposed the follow step for the lexicographic optimization algorithm:

**Step 0.** Choose a positive-definite parametrization matrix *P* of size (*m*-*k*+1) and an initial $\gamma^0 k$.

Take *j* = 0.

**Step 1.** Find the generalized solution of the system

$f_k(x) - \gamma^j{}_k \le 0$;

$f_{k+1}(x) - \gamma_{k+1} \leq 0 \ldots, f_l(x) - \gamma_i \leq 0,$

$f_{l+1}(x) \leq 0, \ldots, f_m(x) \leq 0.$

Let $y^*$ be the estimate of the generalized solution.

**Step 2.** If $(p^i; y^*) \neq 0$ for some $i \in \{k+1, \ldots, m\}$, take $\gamma^{j+1}{}_k = \gamma^j{}_k + (p^k; y^*)$, $j := j + 1$, and go to step 1.

**Step 3.** If $(p_i; y^*) = 0$, $i \in \{k+1 \ldots\ldots, m\}$ take $\gamma_k = \gamma^j{}_k + (p^k; y^*)$ and the $k$ th iteration terminates.

### 2.1.4 Multiattribute Utility Theory

Multiattribute utility theory (MAUT) is an extension of the utility theory applied to multi-objective problems. The Utility theory is the base of the Multi-objective mathematical programming and Goal programming methods because their philosophy is to find an efficient solution that maximizes the decision maker's utility. Another method that follows this philosophy is the ADELAIS (Aide a la Decision pour systems Linearies multicriteres par Aide a la Structuration des preferences) created by Siskos & Despotis (1989).

MAUT has the objective to model and represent the decision's maker's preferential system with a utility/value function $U$ (g), where g is the vector of evaluation criteria

$$g = (g_1, g_2, \ldots, g_n).$$

The most commonly used form of utility function is:

$$U(g) = p_1 u_1(g_1) + p_2 u_2(g_2) + \ldots + p_n u_n(g_n) \qquad (7)$$

where, $u_i$ is the marginal utility function and $u_i(g_i)$ is the utility/value of the alternatives for each individual criterion $g_i$, $p_n$ is the trade-off that the decision maker is willing to take on a criterion in order to gain one unit criterion $g_i$. Therefore as $p_n$ are the weights of the criteria, their sum is equal to one $\sum_{i=1}^{n} p_i = 1$.

## 2.2 Metaheuristic Approaches

### 2.2.1 Swarm Intelligence

#### 2.2.1.1Ant colony optimization

Ant colony optimization algorithm (ACO) is defined as a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs, this system was proposed by Marco Dorigo (1992). The objective is to search for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food. While exploring ants wander randomly, upon finding food they return to their colony laying down a pheromone trail. If other ants find such a path, they are likely not to keep traveling at random, but they instead follow the trail, returning and reinforcing it if they eventually find food. Over time, the pheromones trail starts to evaporate. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones (Deneubourg 1990).

After some experiments about the behavior of ants with a choice between two unequal lights paths, observed saw that ants tended to use the shortest route (Mullen 2009). The model that explains this behavior is as follows:

1. An ant runs more or less at random around the colony;
2. If it discovers a food source, it returns more or less directly to the nest, leaving in its path a trail of pheromone;
3. These pheromones are attractive to nearby ants and they are likely to follow the trail;
4. Returning to the colony, these ants will strengthen the route;
5. If there are two routes to reach the same food source then, in a given amount of time, the shorter one will be traveled by more ants than the long route;
6. The short route will be increasingly enhanced, and therefore become more attractive;
7. The long route will eventually disappear because pheromones are volatile;
8. All the ants have determined and therefore "chosen" the shortest route.

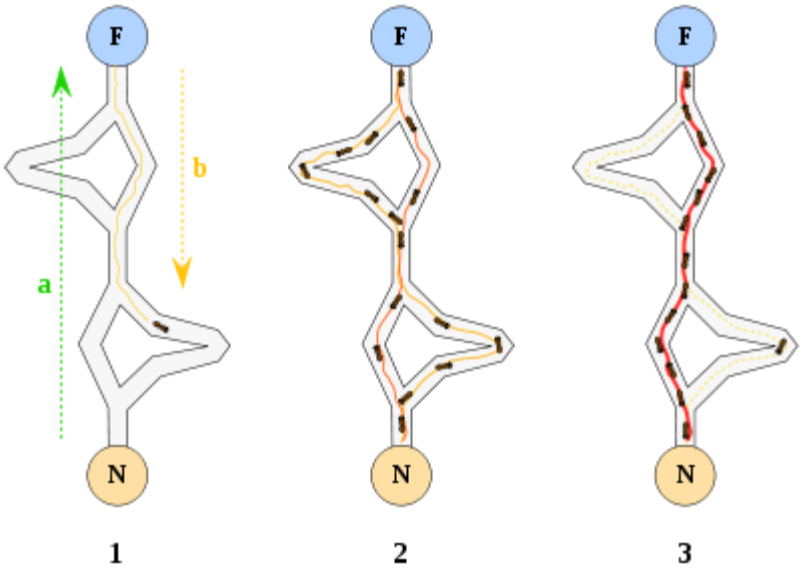


Illustration 1: Ant colony concept.

Ant system algorithm was first applied to the Travelling Salesman Problem (TSP) by Dorigo (1996), this problem involve finding the shortest length tour visiting towns only once. Each ant in the system had the following characteristics:

- An ant decides which town to go to using a transition rule that is a function of the distance to the town and the amount of pheromone present along the connecting path;
- Transitions to already visited towns are added to a tabu list and are not allowed;
- Once a tour is complete, the ant lays a pheromone trail along each path visited in the tour;
- At each interaction the pheromone intensity trails are updated with the following formula;

$$\tau_{ij}(t+n) = (1-\rho)\cdot\tau_{ij}(t) + \sum_{k=1}^{N}\Delta\tau_{ij}^{k} \tag{8}$$

where p is the evaporation rate and is the quantity of pheromone laid on path (I,j) by the $k_{th}$ and between time t and t+n, and is given by

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_K & if \quad ant \quad k \quad used \quad edge \quad (i,j) \quad in \quad its \quad tour \\ 0 & otherwise \end{cases} \tag{9}$$

Here, Q is quantity of trail laid by ants and $L_k$ is the tour length of the kth ant.

The probability of the kth ant making the transition from town I to town j is given by

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left|\tau_{ij}(t)\right|^{\alpha} \cdot \left|\eta_{ij}\right|^{\beta}}{\sum_{k \in allowed_k} \left|\tau_{ik}(t)\right|^{\alpha} \cdot \left|\eta_{ik}\right|^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where $allowed_k = \{N - tabu_k\}$ and where β and α are parameters that control the relative importance of trail versus visibility (Mullen 2009).

A pseudo-code of the Ant Colony Optimization (ACO) algorithm is:

```
procedure ACO_MetaHeuristic
while(not_termination)
generateSolutions()
daemonActions()
  pheromoneUpdate()
end while
 end procedure
```

Some of the improvements made to the ACO Algorithms are: elitist ant system, Max-Min ant system (MMAS) (Stutzle et al. 2000), Ant Colony System (Dorigo & Gambardella 1997), Rank-based ant system (ASrank) (Bullnheimer et al. 1996), Continuous orthogonal ant colony (COAC). In the Elitist ant system, developed by Dorigo, the global best solution deposits pheromone on every iteration, along with all other ants. The Max-Min Ant System (MMAS) Algorithm created by Stutzle (2000) differs in two main ways: only the best ant updates of the pheromone trials and the pheromone update function is bound; the Maximum and Minimum pheromone amounts [$\tau_{max}$, $\tau_{min}$] are added. The Ant Colony System (ACS), Dorigo & Gambardella (1997), differs mainly by its pheromone update function, where ACS employs a local pheromone update. The rank-based Ant System (ASrank), proposed by Bullnheimer, et al. (1996), incorporates the concept of ranking the pheromone update procedure; solutions are ranked according to their fitness. The amount of pheromone deposited is then weighted for each solution, such that solutions with better fitness deposit more pheromone than the solutions with worse fitness. Finally, in Continuous orthogonal ant colony (COAC) the pheromone deposit mechanism is to enable ants to search for solutions collaboratively and effectively. This is done by using an

orthogonal design method: ants in the feasible domain can explore their chosen regions rapidly and efficiently, with enhanced global search capability and accuracy.

Ant algorithms had been applied to many different problems. In which they are particularly well suited to NP-hard combinatorial optimization problems. After the Ant System (AS) was applied to the Traveling Salesman Problem (TSP), AS was also used to solve other NP-hard problems. Some examples are the Quadratic Assignment problem (QAP) (Dorigo, 1996; Maniezzo et al., 1994) and the Job-Shop Scheduling (JSP) problem (Colorni et al., 1994; Dorigo, 1996), in which AS showed its robustness. Other applications are the vehicle routing problem (Bullnheimer et al, 1999; Gambardella & Taillard, 1999), graph colouring (Costa & Hertz, 1997) and set covering (Lessing et al, 2004). The most recent trends include continuous optimization (Socha, 2004; Socha & Blum, 2007) and parallel processing implementations (Manfrin et al., 2006; Talbi et al., 1999).

#### 2.2.1.2 Particle swarm optimization

Particle swarm optimization (PSO) is a computational method that optimizes problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO was developed by Kennedy and Eberhart (1995) and used for optimization of continuous nonlinear functions. This optimization method can be used on problems that are partially irregular, noisy, change over time and with other difficulties as well.

The particle swarm concept originated as a simulation of a simplified social system with the original intent of graphically simulate the graceful but predictable choreography of a bird flock. In PSO a number of simple entities (particles) are placed in the search space of some problem or function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been

moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function (Poli et al. 2007).

The (original) process for implementing the global version of PSO is as following (Eberhart 2001):

1) Initialize a population of particles with random positions and velocities on *d* dimensions in the problem space.

2) For each particle, evaluate the desired optimization fitness function in *d* variables.

3) Compare particle's fitness evaluation with particle's *pbest*. If current value is better than *pbest*, then set *pbest* value equal to the current value and the *pbest* location equal to the current location in *d*-dimensional space.

4) Compare fitness evaluation with the population's overall previous best. If current value is better than *gbest*, then reset *gbest* to the current particle's array index and value.

5) Change the velocity and position of the particle according to the following equations:

$$v_{id} = v_{id} + c_l * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id})$$
$$x_{id} = x_{id} + v_{id} \quad (11)$$

6) Loop to step 2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

As we see, Particle Swarm Optimization algorithm has a small number of parameters that need to be fixed such as: size of the population, acceleration coefficients, and velocities. Some modifications proposed to improve performance of the PSO algorithm are: Inertia Weight (Shi & Eberhart 1998), Constriction coefficients (Clerc 1999), and Fully informed particle swarm (Mendes et al. 2002). In the PSO the addition of the Inertia Weight is to multiply a factor of $v_{id}$ as is show in (3) and (4), and it was proposed by Shi & Eberhart (1998). The application of the inertia weight showed an improved performance in a number of applications, it provided a balance between global and local exploration and exploitation, and results in less iteration.

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id})$$
$$x_{id} = x_{id} + v_{id} \quad (12)$$

Constriction Factor was proposed by Clerc (1999), he indicates that use of a constriction factor may be necessary to insure convergence of the particle swarm algorithm. As in the eq. (13) K is a function of $c_1$ and $c_2$.

$$v_{id} = K * \left[ v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id}) \right]$$
$$K = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|}, where \varphi = c_1 + c_2, \varphi > 4 \quad (13)$$

The improvement in fully informed particle swarm (FIPS), Mendes (2002), is that the particle is affected by all its neighbors. This does not occur in the standard version of PSO; which is only affected by self and best neighbor.

The main applications in which PSO is applied are: image and video analysis applications, design and restructuring of electricity networks and load dispatching, electronics and electromagnetic, power generation and power systems, scheduling, design applications, design and optimization of communication networks, biological, medical, and pharmaceutical applications, clustering, classification and data mining, fuzzy and neuro-fuzzy systems and control, signal processing, neural networks, combinatorial optimization problems, robotics, prediction and forecasting, modeling, detection or diagnosis of faults and the recovery, sensors and sensor networks, applications in computer graphics and visualization, design or optimization of engines and electrical motors, applications in metallurgy, music generation and games, security and military applications, and finance and economics.

**2.2.1.3 Artificial immune system algorithm**

Artificial Immune System (AIS) is inspired by the human immune system. AIS has been defined as "Adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving". The main properties of the Artificial Immune Systems (AIS) are: self-organization, learning and memory, adaptation, recognition, robustness and scalability. (Castro & Timmis 2002)

As before was established the Artificial Immune Systems (AIS) is an algorithm bio-inspired by the principles and processes of the vertebrate immune system. Therefore, immune system is a system of biological structures and processes within an organism that protects against disease by identifying and killing pathogens and tumor cells. There are two arms of the immune system: innate and adaptive. Innate immune system defends the host from infection by other organisms, in a non-specific manner. Specialized cells of the innate immune system have evolved as to recognize and bind to common molecular patterns found only in microorganisms. The main important functions of the innate immune systems are: recruiting immune cells to sites of infection, the production of chemical factors, activation of the complement cascade to indentify bacteria, activate cells and to promote clearance of dead cells, the identification and removal of foreign substances present in organs, tissues, the blood and lymph with the use of specialized white blood cells, activation of the adaptive immune system through a process known as antigen presentation. Adaptive immune system allows the body to launch an attack against any invader that the innate system cannot remove as well as the immunological memory, where each pathogen is remembered by a signature antigen. Adaptive response is antigenic specific and it requires the presence of a non-self compatible antigen that may be present in the body. Should a pathogen infect the body more than once, the memory cells are used to quickly eliminate the foreign invader. The main functions of the immunological memory system are: the recognition of specific non-self antigens during the process of antigen presentation, generation of responses that are tailored to maximally eliminate specific pathogens, development of immunological memory, in which each pathogen is remembered by a signature antibody.

The techniques algorithms are inspired by immunological theories: clonal selection, immune networks, negative selection, danger theory and dendritic cell. Clonal selection algorithms are most commonly applied to optimization and pattern recognition domains. This theory is used to explain the basic response of the adaptive immune system to an antigenic stimulus. It is believed that only cells

capable of recognizing an antigen will proliferate, while those that do not recognize an antigen no longer reproduce and eventually undergo apoptosis. Negative selection algorithm is used for classification and pattern recognition problem domains where the problem space is modeled in the complement of available knowledge. Negative selection theory inspired by the positive and negative selection processes that occur during the maturation of T cells in the thymus called T cell tolerance. Immune network algorithms have been used in clustering, data visualization, control, and optimization domains, and share properties with artificial neural networks. This is based in the idea that the immune system is capable of achieving immunological memory via the existence of a mutually reinforcing network of B cells. The danger theory algorithm is based in the idea that antigen presenting cells (APCs) are activated by danger/alarm signals from injured cell, such as those exposed to pathogens, toxins, mechanical damage, and so forth .The dendritic cell algorithm (DCA) is inspired by innate immune, especially in the function of dendritic cells. DCA have been used in optimization, and classification.

The first paper which proposes the implementation of the immune system theory as a network model was "The immune system, adaptation, and machine learning" by Farmer et al. (1986). It describes the parallels between a network of immune cells and the Holland classifier system (Holland 1975). As result, inspired by Famer's publication Ishida (1990) proposed the first immune network algorithm. In addition, Castro & Zube (2000) proposed the clonal selection algorithm (CSA), called after CLONALG, which is based on clonal selection and affinity maturation principles. This paper compares the CLONALG against genetic algorithm (GA), showing that clonal selection algorithm can reach a diverse set of local optima solutions. Using this algorithm to binary character recognition, multimodal optimization, and the Traveling Salesman Problem (TSP), it shows capability of performing learning and maintenance of high quality memory. The areas in which Artificial Immune Systems (AIS) were applied are: computer security, numerical function optimization, learning, bio-informatics, image processing, robotics, adaptive control systems, virus detection and web mining. The earlier work

implementing the Artificial Immune System (AIS) in the area of computer security proposed a simple self/non-self discrimination algorithm (Forrest et al. 1994), follow by many others work implementing AIS in this area (Dasgupta 2006). In the recent years AIS was implemented to some applications as: data mining, networking and computer security, optimization, automation and design, anomaly detection, bioinformatics, text processing, pattern recognition, clustering and classification.

### 2.2.2 Evolutionary algorithms

#### 2.2.2.1 Genetic algorithm

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. The techniques used are inheritance, mutation, selection and crossover.

The process of GA is as follows (Tamaki et al. 1996):

1) Individual solutions are generated randomly to form an initial population. The size of it depends on the nature of the problem. Evaluate the fitness of every individual in the population.
2) Selected a proportion of the existing population to breed a new generation. An individual solution is selected through a fitness-based process where best solutions are more likely to be selected. Another method used is a random sample of the population.
3) Generate a second generation population applying crossover and mutation to the select solutions. The child is going to have the same characteristic of the parents.
4) This generational process is repeated until a termination condition has been reached.

Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above.

A simple pseudo-code of the Genetic Algorithm is:

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in that population
3. Repeat on this generation until termination: (time limit, sufficient fitness achieved, etc.)
    1. Select the best-fit individuals for reproduction
    2. Breed new individuals through crossover and mutation operations to give birth to offspring
    3. Evaluate the individual fitness of new individuals
    4. Replace least-fit population with new individuals

Improvements of the GA include: population-based but non-pareto approaches, pareto-based approaches, and maintenance of diversity. In Population-Based but Non-Pareto Approaches, the selection/reproduction is performed by treating the non-commensurable objective functions separately. The most important approach is the Vector Evaluated Genetic Algorithm (VEGA) created by Schaffer (1985), in which sub-populations of the next generation are reproduced from the current population according to each of the objectives separately. Fourman (1985) proposed to perform a selection by comparing pairs of individuals with respect to one objective. Kursawe (1991) develop a method which proposed a multi-objective evolution strategies, in which a population of the next generation are formed by repeating a selection operation using one objective selected randomly according to pre-determined probabilities. The Pareto-Based approaches proposed that the selection/reproduction is performed by referring not only to objective values themselves but also to the dominance property of them (Tamaki et al. 1996). Golberg (1989) proposed a method of ranking and fitness assignment based on the Pareto-optimality of an individual. The maintain of diversity is required in solving MOP and in most GA's, the genetic diversity of the population is lost due to their stochastic selection processes, resulting in fitness sharing methods that have been additionally used. Kita (1995) proposed a combination of thermodynamical genetic algorithm (TDGA) and Pareto-based ranking method.

Some of the applications in which Genetic Algorithm has been applied are State Assignment Problem (SAP) (especially Traveling Salesman Problem TSP), Economics, scheduling and Computer-Aided Design.

# Chapter 3: Redundancy Allocation Problem

The optimization of the system reliability is very important in real-world applications and various kinds of systems have been studied in the literature for decades. Two approaches can be enforced to increase the system's reliability: one is to increase the reliability of the elements constituted in the system and another is the use of redundant elements in various subsystems in the system. The first approach is not effective because, the system is not entirely accurate, although elements increase their reliability. The second approach is more effective, because it assigns a redundancy level for each subsystem and the system reliability increase. The most effective approach is found to be a combination of both because it consists of selecting the optimal combination of elements and redundancy levels; this method is more accurate and feasible. To solve this type of problem, several researchers have developed different methodologies. The techniques used to solve the Redundancy Allocation problem are dividing the problems in three categories: approximate techniques, exact techniques, and heuristic/meta-heuristic techniques.

The Redundancy Allocation Problem (RAP) refers to the process of adding redundant components to improve the reliability of the system while meeting the set constraints of cost, weight, volume, or any other; currently, there are many tools used in the literature for solving such problems. There are many variations of the RAP that usually involves the selection of components and levels of redundancy, this to maximize the reliability of the series-parallel system. The reason of adding parallel redundant components to a series system is to prevent a complete system failure. This is because in a series system all components must be operational before the system can work correctly; the total reliability is then given by the multiplication of the reliability of its *n* components.

$$R_{Total}=R_1*R_2*R_3* \ldots..*R_n \quad (14)$$

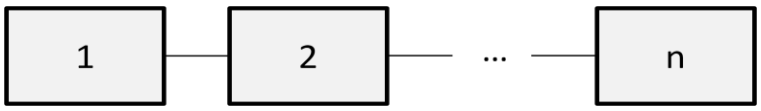


Figure1: Series System

In contrast, in a series-parallel system each subsystem has their components in parallel and then it is connected in series with the other subsystems. As result, if an element fails the other redundant elements will remain operational and the whole system is not going to be affected. The total reliability of a series-parallel system is:

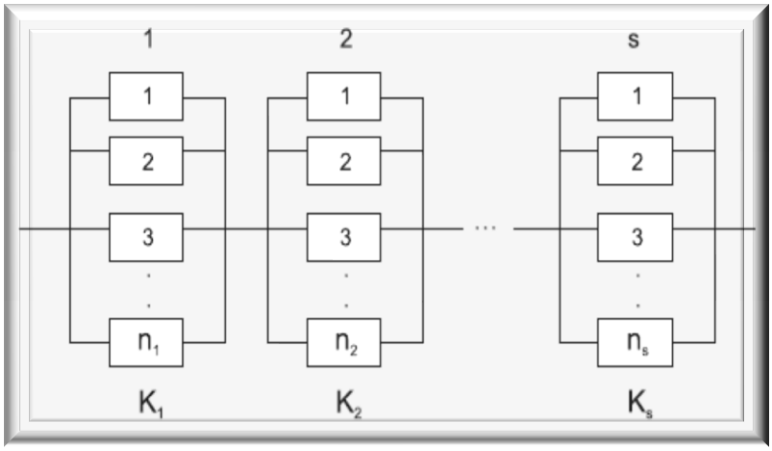


Figure 2: Series-parallel system.

$$1-\prod_{i=1}^{s}\prod_{j=1}^{n}\left[1-R_{ij}\right] \tag{15}$$

**3.1 Single Objective Redundancy Allocation problem (SO-RAP)**

In the paper of Chen & You (2006) and Chen (2006), they suggested an approach employing immune algorithms to solve the redundant allocation problem (RAP). As a result, they proposed the following computational procedures in the implementation of the new immune algorithms-based approach as in Figure 3 are show:

**Step 1:** Generate an initial population of strings (antibodies) randomly.

**Step 2:** Evaluate each individual in current population and calculate the corresponding fitness value for each individual.

**Step 3:** Select the best n individual with highest fitness values.

**Step 4:** Clone the best n individuals (antibodies) selected in Step 3. Note that the clone size for each select individual is an increasing function of the affinity with the antigen. In other words, the number of posterity of each antibody is proportional to their fitness values, i.e., the higher the fitness, the larger the clone size.

**Step 5:** The set of the clones in Step 4 will suffer the genetic operation process, i.e., crossover and mutation.

**Step 6:** Calculate the new fitness values of these new individuals (antibodies) from Step 5. Select those individuals who are superior to the individuals in the memory set, and then the superior individuals replace the inferior individuals in the memory set. While the memory set is updated, the individuals will be eliminated when their structures are too similar. This way the individuals in the memory set are kept diverse.

**Step 7:** Check the stopping criterion, if not stopped then go to Step 2. Otherwise go to next step.

**Step 8:** Stop. The optimal or near-optimal solution(s) can be obtained from the memory set.

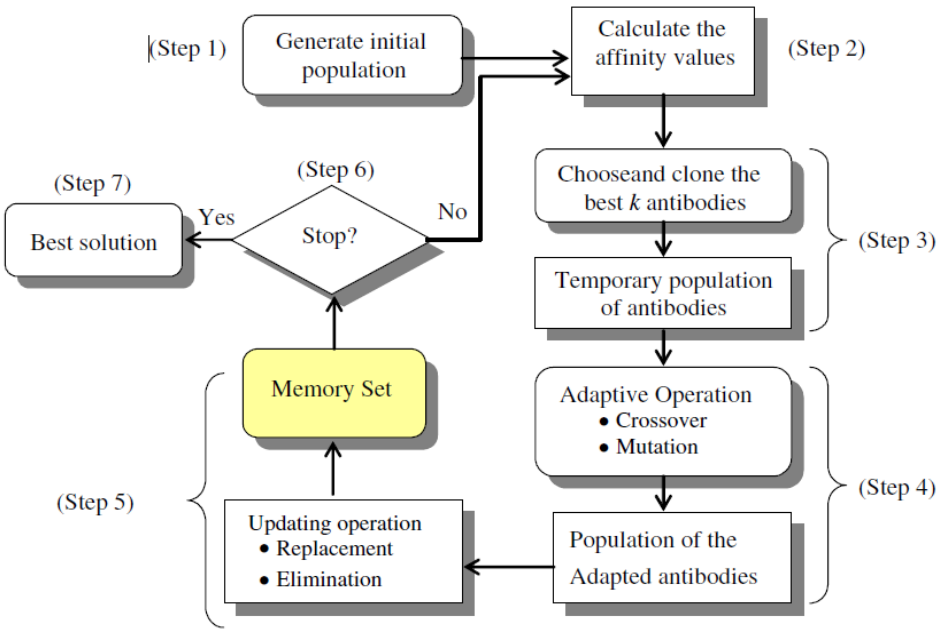


Figure 3: Flow chart of the computational procedures.

The results obtained from the application of the immune system algorithm, compared with other papers which implemented the Genetic algorithm, demonstrated that the proposed approach performs

better in those test problems with larger values of W (weight) obtaining solutions of quality. The disadvantage of this algorithm is that the CPU processing time is longer because IAs required more memory due to the adopting process, each iteration alone requiring more time. In conclusion, the proposed approach has superior performance comparing with the three literatures, as long as the CPU processing time is not a critical factor.

Liang & Smith (2004) were the first authors that implemented the Ant Colony Optimization (ACO) algorithm to the redundancy allocation problem. ACO was coded in Borland C++, and compared with the 33 variations of Fyffe et al. (1968) to see its performance. As result, ACO tends to find better solutions than the GA, and is less sensitive to random number inputs. After, Liang & Smith () implemented an ant system algorithm combined with an adaptive penalty method, which employs the notion of a "Near Feasibility Threshold" (NFT) for each constraint. It is the threshold distance from the feasible region that is considered as being close to feasibility. In addition, they implement an elitist strategy to enhance the magnitude of trails of good selection of components; they also introduced mutation to the AS, which helps explore new search areas. The set of problem select to see the performance of this algorithm are the ones proposed by Fyffe et al. (1968). In those problems, it is needed to select components and redundancy-levels to maximize system reliability give constraint cost and weight. Nahas et al. (2007) proposed a hybridization inspired by the ant colony meta-heuristic optimization method and the degraded ceiling local search technique. It was compared with the Fyffe 33 test problems, and it performs well and competitive.

Evolutionary algorithms (EAs) are stochastic and robust meta-heuristics problem solvers that are useful to solve reliability-redundancy allocation problems. EAs such as genetic algorithms, evolutionary programming, evolution strategies and differential evolution are being used to find global or near global optimal solutions. Coelho (2009) presented an evolution approach based on chaotic sequences using Lozi's map for reliability redundancy allocation problem. The use of deterministic chaotic sequences

instead of random sequences improves the performance of differential evolution. The main characteristic of this approach is its fast converge rate while maintaining the diversity of the population, so as to escape from local optima. The over speed protection system of a gas turbine was the application used to evaluate this approach.

Coit & Smith (1996) proposed an adaptive penalty-guided genetic search, which efficiently and effectively searches over promising feasible and infeasible regions to identify a final optimal solution. Its results conclude that it is powerful and robust for problems with large search spaces and difficult-to-satisfy constraints. The penalty function is a dynamic NFT and it responds to the search history. It showed superiority to the statics NFT in having quality and variance of the final feasible solution; NFT superiority increased as the problem become harder. Hsieh et al. (1998) used genetic algorithms to solve series, series parallel and complex systems. The problem is maximizing the system reliability, while maintaining feasibility with respect to three nonlinear constraints; cost, volume and weight. The genetic algorithms perform well and some solutions are better than best knows quality heuristic methods. The advantages of these algorithms are: generation of multiple design alternatives, lack of a requirement for the computation of derivatives, and relatively efficient computation. Yokota et al. (1996) solve a non-linear mixed integer programming problem using genetic algorithm and aggregating a penalty function. The penalty function evaluates those infeasible chromosomes generated from genetic reproduction and the crossover ensures GA's search in its linear combined domain. After solving the example problems GA demonstrated that GA search has a much wider search space and forces populations to converge to a feasible region in a later generation. Coit & Smith (1996 b) proposed a hybrid genetic algorithm with a neural network. Genetic algorithm searches among candidate designs of the system configuration to identify optimal levels of component redundancy, while neural network provides the objective function value of each candidate design by estimating the system reliability. The problems solved had first the objective of minimizing cost and then to maximize the system reliability of the system. Results showed

that hybrid method proposed is robust to parameter settings and random number seed, while needing computational resources. Coit and Smith (1996c) performed a problem-specific genetic algorithm to solve a nonlinear optimization problem using a dynamic penalty function. The method proposed was applied to a series-parallel system with multiple component choices available for each of several k-out-of-n: G subsystems. This algorithm performed well in the redundancy allocation proposed by Fyffe et al.(1968).

Coelho (2009) presented a Particle swarm optimization algorithm based on Gaussian distribution and chaotic sequence (PSO-GC). Two examples of mixed-integer programming, with the objective of maximizing reliability subject to cost, weight and volume, are compared with PSO-CA and PSO-CO. The results of PSO-GC demonstrated that it has good-trade-off between exploitation in Gaussian distribution and exploration by chaotic sequence. Beji et al. (2010) developed a hybrid algorithm based on particle swarm optimization and local search algorithm. Also, an addition to this algorithm is performed with an adaptive penalty function that encourages exploring within the feasible region near feasible region, while discouraging search beyond that threshold. The hybrid algorithm is apply to variation of 3 different problems and compared to Tabu Search and Multiple Weighted Objectives solutions. Chen (2006 c) proposed a penalty-guided particle swarm optimization approach to solve the mixed-integer reliability design problems. The difficulty that this method faces is to maintain feasibility with respect to three nonlinear constraints (cost, weight and volume and weight products). This approach efficiently and effectively searches over promising feasible and infeasible regions to find the feasible optimal or near optimal solution.

Kulturel-Konak et al. (2003) developed a tabu search meta-heuristic with a penalty function to allow and promote search on the infeasible region with an adaptively changing Near Feasible Threshold (NFT). The way it works is by penalizing the objective function for infeasible solutions with an adaptive penalty to increase and decrease the adaptability of the search by the information stored in

the short and long term memories. The two problems used have the following characteristic: the problem one has the objective of maximizing reliability subject to cost and weight constraints and problem two has the objective of cost minimization and constraint to reliability and weight.

Jianping (1996) proposed a bound heuristic algorithm to solve the nonlinear integer programming refined from the optimal allocation of redundant components, in which the optimal solution has been proved to be in the bound region of the problem. Any heuristic technique could be used to obtain a new and better bound point from any given bound point in the bound region. This was compared using 5 algorithm methods: Sharna-Venkateswaran (SV) heuristic method, Aggarwal-Guta-Misra (AGM) heuristic method, Shi Ding Hua (SDH) heuristic method, Xu-Kuo-Lin (XKL) method, and Misra Integer Programming (MIP). Results concluded the bound heuristic algorithm is very efficient and obtain better solutions than the compared algorithms .Also the algorithm is independent of nature of objective and constraint functions, the initial bound point could be arbitrary.

Jianping (1996) developed a bound dynamic programming. The optimal solution is obtained in the bound region of the problem by using dynamic programming. The characteristics of this method are; the greater the dimension size, the larger the feasible region and it was found more economical and effective than Misra bound search technique to obtain the exact solutions. Misra (1971) presented a dynamic programming formulation for the reliability-redundancy allocation problem. The advantages are that it is much faster than those suggested by Bellmen and the computer program has less memory requirement to solve larger problems. Fyffe et al. (1968) formulated a computational algorithm using dynamic programming (DP-LM). The algorithm used a Lagrange multiplier to reduce the dimensionality of the problem from two to one. The problem solved is a series system with 14 stages, in addition the functional units 1,3,6,9,12 and 14 have 4 design alternatives and functional units 2,4,5,7,8,10,11 and 13 have 3 design alternatives. The objective of the problem is to find the optimum number of redundant components and the design alternative of each stage in order to obtain the maximize system reliability

while keeping the total system cost of 130 and weight of 170. Nakagawa and Miyazaki (1981) presented an algorithm to solve pure-integer separable nonlinear programming problems with two constraints, which is called the N&M algorithm. This algorithm uses surrogate constraints, which attempts to solve a given problem generating values of surrogate multiplier and then solve the surrogate problem. N&M algorithm's performance was tested using 33 variations of the Fyffe's propose problem, in which value of weight varies from 159 to 191. The N&M algorithm find the optimal solutions or good near optimal solutions, demonstrating be superior that DP-LM algorithm in the number of iterations and in the gap region.

### 3.1.1 Exact Methods

Kuo et al. (2009) solved constrained reliability optimization problems using a hybrid of the Lagrange multiplier method and the branch-and-bound technique. Lagrange multiplier method treats the number of redundancies as real numbers, while branch-and-bound technique obtains the integer solution. It has the advantages of an exact method and an enumerative method; it reaches a solution close to optimum and guarantees the global optimum, and doesn't take many interactions. The examples solved where is 4-stage series system with two linear constraints and 5-stage series system with three nonlinear constraints, the results were better than previous methods that solved the redundancy allocation problem and mixed integer-type reliability-redundancy allocation problem. Coit & Konak (2006) proposed a multiple weighted objective heuristic (MWO) which is based on a transformation of the problem into a multiple objective optimization problem, and then into a different single objective problem. This heuristic is based on solving a sequence of linear programming problems that allows linear programming algorithms and software to be used for the reliability-redundancy allocation problem. The multiple objectives are to maximize reliability of each subsystem simultaneously. The MWO is tested with the Fyffe proposed problem (14 subsystems with 3 or 4 component choices in each subsystem). Its result demonstrated that it is a very good heuristic and superior to the max-min heuristic in efficiency

and performance. When compared with GA and TS its advantages was CPU times which were very promising to the applicability to large size problems.

Hwang et al. (1979) used two methods to solve non linear optimization problems for reliability, a generalized Lagrangian function (GLF) method and the generalized reduced gradient (GRG) method. These methods were tested with two problems and compared with Sequential unconstrained Minimization Technique (SUMT) and the direct search approach by Luus and Jaakol (LJ). GRG and GLF demonstrated to be effective when applied to large-scale nonlinear programming problems. Chern & Jan (1985) proposed a 2-phase solution method. In phase 1 the problem is decompose into sub problems and solved by dynamic programming. In phase 2, the problem is solved using a 0-1 multiple choice knapsack and then a branch and bound algorithm produce a combinatorial tree. The problem used to test the algorithm was a 3-stage series system, with objective of maximizing system reliability subject to cost and weight. Misra &Sharma (1991) formulated an algorithm to solve integer-programming problems, based on functional evaluations and a limited search close to the boundary of resources. Prasad & Kuo (2000) presented a search method based on lexicographic order and an upper bound on the objective function. The upper bound helps in reducing the search. The advantages of P&K algorithm are its simplicity and its applicability. Chern & Jan (1986) developed a new three reliability optimization models formulated as parametric nonlinear programming problems. These models were created to clarify what happens if a certain change is made in the constraint values.

### 3.2 Multi-objective Redundancy Allocation Problem (MO-RAP)

Zhao et al. (2007) proposed the multi-objective ant colony system (ACS) a meta-heuristic approach to solve the reliability optimization problem of series-parallel systems. Through random search, constructive local move and long-term dynamic memory strategy, the proposed method efficiently builds good solutions for the RAP. After the ACS-RAP was tested, it was compared with GA-RAP. ACS better performed in terms of best solution, reduced variation and great efficiency. Also

ACS-RAP was compared against ACO-RAP and results showed a potentially higher efficiency and better capacity to handle large-scale problems. As result ACSRAP has a better constructive strategy than other ant colony algorithms. Through the combination of probabilistic search, multi-objective formulation of local moves and the dynamic penalty method, the multi-objective ACS-RAP when allows to obtain an optimal design solution frequently and faster. Its advantage compared with other methods, is that ACS-RAP can be applied to a more diverse problem domain.

Salazar et al. (2006) used a second-generation Multiple-Objective Evolutionary Algorithm (MOEA), to solve a redundancy allocation problem that was reformulated from a single objective, mixed-integer, non-linear programming problem as a multi-objective problem (MOP) The MOEA (NSGAII) demonstrates the ability to identify a set of optimal solutions (Pareto front), which provides the Decision Maker with a complete picture of the optimal solution space. The results showed that MO formulation provides more information than SO, and that MOEA's has the capability to solve MO problems. The RAP problem used to evaluate the performance had two objectives: maximize reliability and minimize cost. Though, MOEA can be used to solve any type and number of objectives.

Li et al. (2009) presented a two-stage approach for solving multi-objective system reliability optimization problems. The first stage is finding the Pareto optimal solution set using a multiple objective evolutionary algorithm (MOEA). In the second stage an integrated multiple objective selection optimization method (MOSO) is used. The process begins by applying a self-organizing map (SOM), that has the capability of preserving the topology of the data, this to classify the Pareto optimal solutions into several clusters with similar properties. Then the data envelopment analysis (DEA) is applied to determine the final representative solutions. Applying this process, the final solutions to the multi-objective system reliability optimization problem can be easily determined in a more systematic and meaningful way; the advantage of this approach is that it reduces the overall set of promising solutions.

Bachlaus et al. (2006) developed a chaos-embedded hierarchical particle swarm optimization (CE-HPSO). The main characteristic of the CE-HPSO is that it uses a chaotic sequences and time-varying acceleration coefficients which help to diversify the search space. A hierarchical particle swarm optimizer is used to restrict the premature convergence. CE-HPSO combines the normalized values of reliability, cost, weight, and volume. It was tested with small and large problems and compared against genetic algorithms. The objectives of the problems used are maximizing system reliability and minimizing cost, weight, and volume. Zavala et al. (2005) proposed an enhanced Particle Swarm optimization algorithm with multi-objective optimization concepts PESDRO. PESDRO was tested with RAPs with the functions to maximize reliability and minimize cost. Finding and found that PESDRO is robust in solving this type of problems.

Liang & Lo (2010) developed a variable neighborhood search (VNS) algorithm to solve the multiple objective redundancy allocation problems (MORAP). A new selection strategy of base solutions that balances the intensity and diversity of the approximated Pareto front is introduced. Also, two neighborhood structures are employed to explore the search space. To evaluate the performance MOVNS is testing on three sets of complex problems with 5, 14 and 14 subsystems. In the first problem the objectives are maximize reliability and minimize cost, with a constraint to weight and volume. The second problem had the objectives of maximizing reliability and minimizes cost, with a constraint to weight. The third example had the objectives of maximizing reliability and minimizes weight, with the constraint of cost. The results showed that MOVS is able to generate more non-dominated solutions in a very efficient manner and performs competitively.

Soylu & Ulusoy (2011) proposed the τ- neighborhood approach to increase the number of references. The proposed approach solved a bi-objective redundancy allocation problem, with the objectives of maximizing the minimum subsystem reliability and minimizes the overall system cost. The problem was solved with the €-constraint approach the Pareto optimal solutions were found. Then

UTADIS, a sorting procedure was used to categorize the solutions into preference ordered classes (A, B, C). Afterwards, separation of classes by thresholds according to the utility function continued. UTADIS can find additional reference solutions, reflecting the Decision maker (DM) preference; this can eliminate the alternative utility function of UTADIS. Also, it can find real references around the infeasible region. The results showed that τ-neighborhood approach is promising to be used with based algorithms.

Mahapatra & Roy (2006) introduced a new fuzzy multi-objective optimization method. The redundancy allocation problem solves a series and complex system with two objectives: maximize the reliability and minimize the cost of the system. Results reflect that the proposed method is effective in achieving a superior compromise between mutually conflicting objectives, when the problem parameters, prescribed goals, and design constraint are not known precisely. Dhingra (1992) solve a multi-objective, nonlinear, mixed-integer mathematical problem with several design constraints. Sequential unconstrained minimization techniques, in conjunction with heuristic algorithms, are used to find an optimum solution. The fuzzy optimization techniques can be useful during initial stages of the conceptual design of engineering systems, where the design goals and design constraints have not been clearly indentified. These can effectively model the information in the objective function and constraints to formulate fuzzy goals and constraints.

Sakawa (1981) proposed a large-scale multi-objective optimization method for a series-parallel system by applying both the surrogate worth trade-off method and the dual decomposition method. The example used to evaluate its performance is a system composed of 16 subsystems operating in series. Hikita et al. (1992) developed a solution method to solve the problem of optimizing redundancy and component reliability simultaneously. The problem was formulated as a mixed-integer nonlinear programming problem and was solved by applying a surrogate dual method, which is solved by dynamic programming. Two countermeasures to surrogate gaps are considered: modifying the original problem

to tighten the constraints and decreasing component reliabilities in the vertical direction to the tangential plane of the objective function.

Kulturel-Konak et al. (2008) presented a Pareto front pruning method to solve different versions of multi-objective system redundancy allocation problems, which assign numerical values to an objective function. The proposed method uses first a tabu search meta-heuristic approach to find the entire Pareto-optimal front, and the Monte-Carlo simulation gives a set of Pareto-optimal solutions based on user-defined objective function preferences. This method requires minimal computational effort.

# Chapter 4: Viral System Algorithm

## 4.1 Virus in nature

A virus is defined as a microorganism smaller than bacteria, which cannot grow or reproduce apart from a living cell. A virus invades living cells and uses their chemical machinery to keep itself alive and to replicate itself . Viruses are intracellular parasites shaped by nucleic acids like DNA or RNA, and proteins. The protein generates a capsule called a Capsid, where the nucleic acid is located. The capsid and the nucleic acid shape the nucleus-capsid, which finally defines the virus. Each type of virus has different capabilities with respect to infecting cells, including mechanisms of replication and spread/inoculation, and strategies to weaken the immune response of the host.

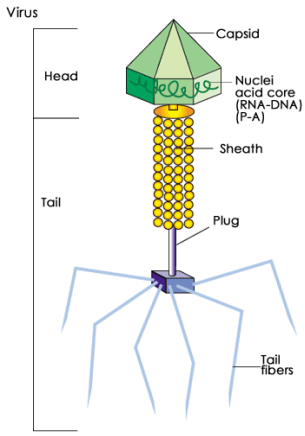


Figure 4: Virus structure.

### 4.1.1 Viral replication process

Viral replication is the term used by virologists to describe the formation of biological viruses during the infection process in the targeted host cells (Cortes et al. 2008). Before viral replication can occur, viruses must first get into the cell and successfully infect the cell, with the sole purpose to allow production and survival of its kind. Replication between viruses varies greatly and is dependent on the type of virus involved. Although the replication mechanism of viruses depends on the type of virus, phage replication process is described in this section.

A phage is defined as a virus that is parasitic that reproduces itself in bacteria and uses the bacterium's machinery and energy to produce more phage until the bacterium is destroyed and phage is released to invade surrounding bacteria, these viruses particularly infect bacteria.

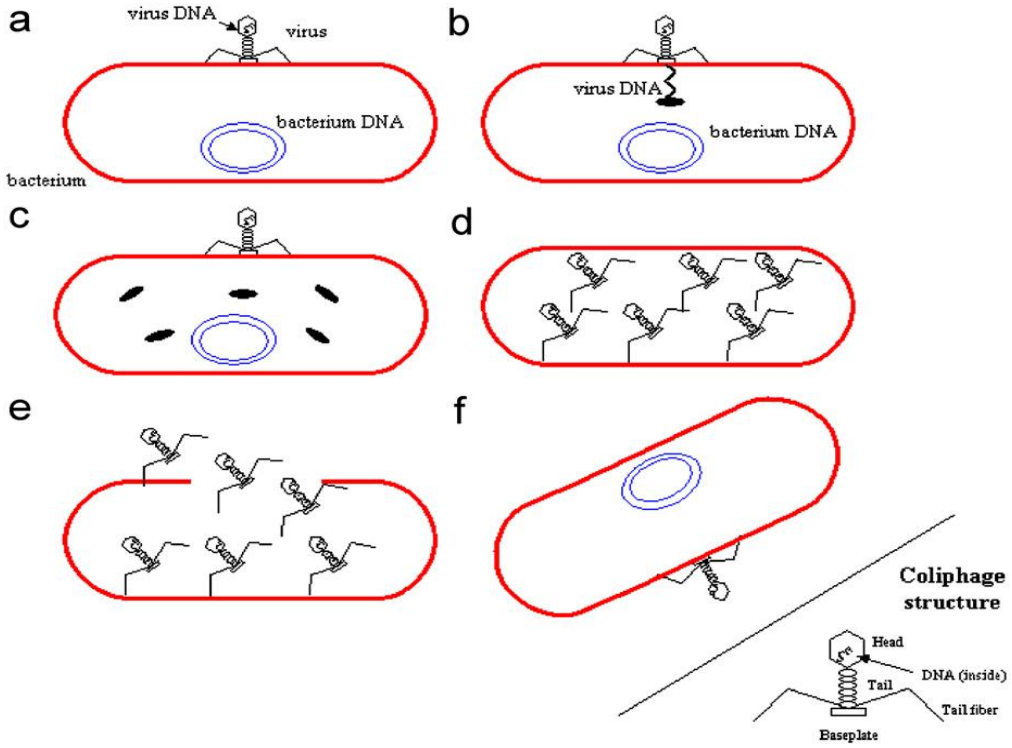


Figure 5: Lytic replication.

The phage life cycle stated by the creators of the VSA is the following, known as lytic replication:

1. The virus is adhered to the border of the bacterium. Subsequently, the virus is able to penetrate the border and inject its genetic material into the cell.

2. The host cell's DNA or RNA is used to replicate the virus and it takes over the cell's metabolic activities. The infected cell stops the production of its proteins and begins to produce the phage's proteins. By doing so, it starts to replicate copies of the virus nucleus-capsids.

3. After replicating a number of nucleus-capsids, the bacterium border is broken down, and new viruses are released, which infect other close cells.

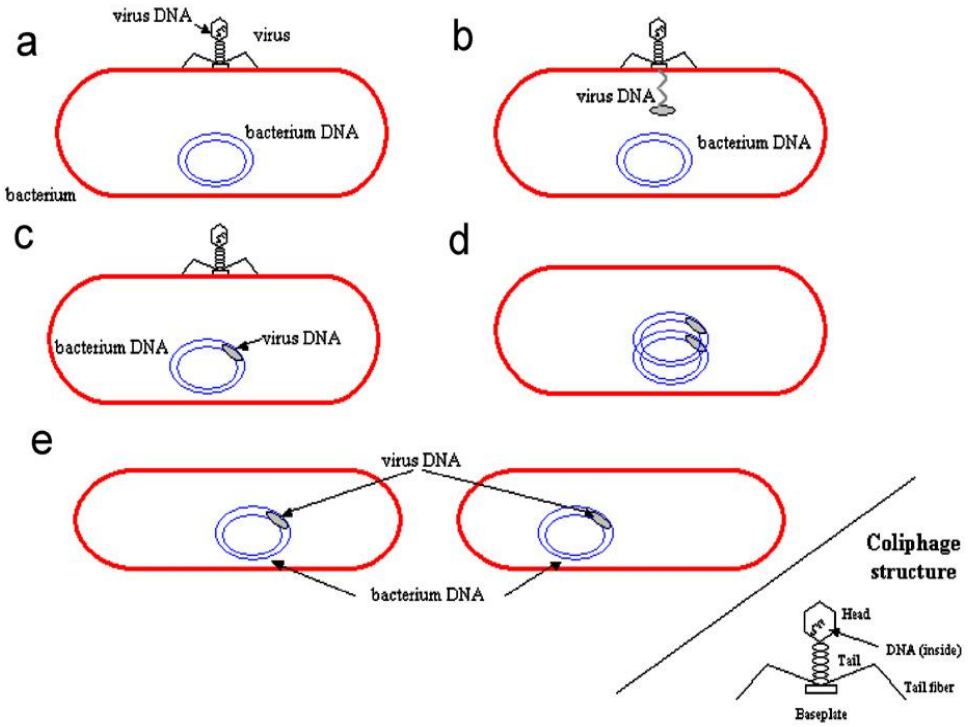


Figure 6: Lysogenic replication.

The life cycle of a virus can be completed in a single step, through this lytic replication, or it can include different steps. Some viruses are capable of lodging in cells giving rise to the lysogenic replication. In this case, the process is as follows:

1. The virus infects the host cell, and lodges in its genome.
2. The virus remains inactive inside the cell during a certain period of time until it is activated by any cause (such as ultraviolet irradiation or X-rays).
3. The replication of the virus resumes and restarts lytic replication continues.

Most viruses lodged in the genome of the host cell can cause alterations in the cell genome, as a consequence the majority of virus infections results in death of the host cell. In contrast, some viruses cause no apparent changes to the infected cell and do not necessarily kill their host, like the Adeno-associated virus.

### 4.1.2 Creation of antibodies and interaction with virus

Before the immune system can react against an antigen, antibodies must be created antibodies. They are used by the immune system to identify and neutralize foreign objects, such as bacteria and viruses. Once antibodies are created, two possibilities can be achieved: the organism defeats the virus, implying that the immune system protects the host, or the virus overcomes the defense capabilities of the organism and the host's death takes place.

## 4.2 Viral system algorithm principles

A virus attacks an organism starting with the weakest neighboring cell. By doing this, there is a higher probability for successful replication of the virus. In this case, the health of the system is represented by f(x) in the application of the redundancy allocation problem; this would be defined as *function of cost*. This way virus optimize their objective and infect less healthy individuals, having as consequence viruses propagating through a major size of the population.

When the virus infection probability decreases, the virus' cell genome (solution encoding) mutates to a different form with the intention of achieving a higher success in the infection process in order to finally succeed with the infection. The main objective of the virus is to infect the host and successfully overcome any immunological response. From the virus adaptation and replication theories, the Viral System was developed. For our purpose our main objective using the Viral Systems are the two following:

I. Minimize our function cost and weight.

II. Maximize reliability.

Both of these are expected to reach satisfactory values by following the viral algorithm which would be explained in the next section.

### 4.2.1Viral system components

This Viral System, according to Cortes et al. (2008), is defined by three components: a set of viruses, an organism, and interactions between them.

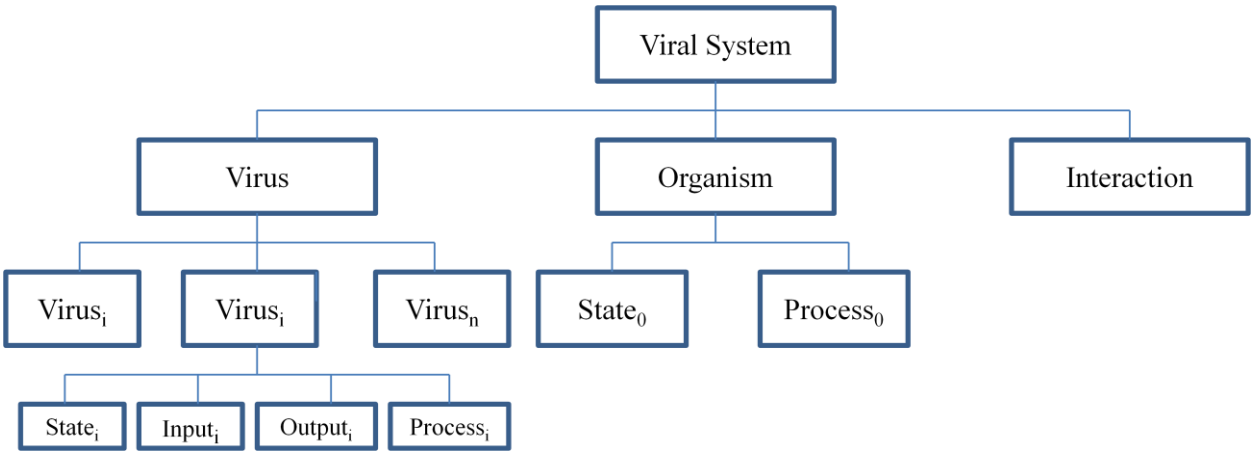


Figure 7: Viral System algorithm components.

*Virus*

The *Virus* component of the VS is a set consisting of individual viruses, each one of them is defined in four components:

- State
- Input
- Output
- Process

Where each component is defined as following:

• *State*$_i$- Defines the cell's condition of the cell infected by the virus.

• *Input*$_i$ - Identifies the information that the virus can obtain from the host.

•*Output*$_i$- Identifies the proceedings that the virus can take.

• *Process*$_i$- Represents the autonomous behavior of the virus, changing the *State of the cell*, it corresponds to the replication operator in computational terms.

*Organism*

The *Organism* components of the VS are defined as following:

- *State$_0$*. - Represents the health of the cell as know by the clinical picture .The virus will attack the least healthy cell. Also, in this state the clinical picture constrains are, considering (eq. 16), adding the *lowest* value of F(x) defined by the phase$_0$ of the organism.

$$K = \{x: g_i(x)\ 0,\ i = 1 \ldots n\}, \quad (16)$$

Where the variables are defined as following:

  - K: Is the clinical picture this contain the overall information of the infection needed by the algorithm in each instant *t*. Also, the clinical picture consists of every three triple genome-NR-IT as the "*State* of each *Virus"*.
  - x: the cell that is observed.
  - $g_i$ (x): Is the encoding of each cell or feasible solution.

- *Process$_0$*- Represents the behavior of the organism that tries to protect itself from the infection threat, consisting of antigen liberation.

*Interaction*

The interaction component of the Viral System is the action of the Input and Output that leads to a process of every virus and the resultant organism response.

#### 4.2.1.1 Neighborhood identification

The input sensor of each virus collects information from the organism and detects the set of cells close to the infected host; this set is named the neighborhood of the feasible solution x. The neighborhood depends on the shape of constrains in the problem previously mentioned, $G_i$ (x).

As previously mentioned, there are two pathways a virus might follow: lytic or lysogenic replication. In our problem the path would be decided by the output of the function. The difference between both cycles in a virus is that lytic replication is executed only in one step and in lysogenic replication the life cycle of the virus is executed in two steps. The probabilities of the lytic replication is $p_{lt}$ and lysogenic probability as $p_{lg}$, where $p_{lt} + p_{lg} = 1$.

#### 4.2.1.2 Process: Lytic replication

For the lytic replication case, we considered two different evolutions for different types of infections: a selective infection that evaluates the individual cells to be infected and once the first cell is infected it will selectively attack neighboring cells. The second type is a devastating infection where a massive number of cells are infected.

The lytic replication starts only after a specific number of nucleus-capsids have been replicated; each time instant (iteration *t*) a number of virus replications (NR) take place. The number of replications per iteration is calculated as a function of the binomial variable *Z* adding its value to the total NR.

After an established number of nucleus-Capsids have been replicated inside the cell (LNR), the bacterium border is broken, thus liberating lodged viruses; these new viruses are active and ready to infect new cells. The value of LNR is dependent on the cell's health conditions. Although a healthy cell (with high value of F(x)) will have a low probability of getting infected, the value of LNR will be higher; this is due to the cell's better capabilities of replicating a virus. If there is a small F(x) then the opposite will occur, resulting in lower values of LNR.

The following eq. (17) Shows the calculation procedure for LNR in a cell *x*:

$$LNR_x = LNR_0\left(\frac{f(x) - f\hat{}(x)}{f\hat{}(x)}\right) \qquad (17)$$

Where *f(x)* is the cell that produces the best-known result and *x* is the infected cell being analyzed. $LNR_0$ is the initial value for LNR.

The number of Nucleus-Capsids replicated by each iteration can be approximated by a binomial distribution given by the maximum level of Nucleus-Capsids replicated, LNR, and the single probability of one replication, pr: Z =Bin (LNR,pr).

Once the bacterium border is broken, it liberates produced viruses and each one of these viruses has a probability (pi) of infecting other neighboring cells.

### 4.3 Viral System Algorithm

The follow diagram shows how Viral System Algorithm VSA works.

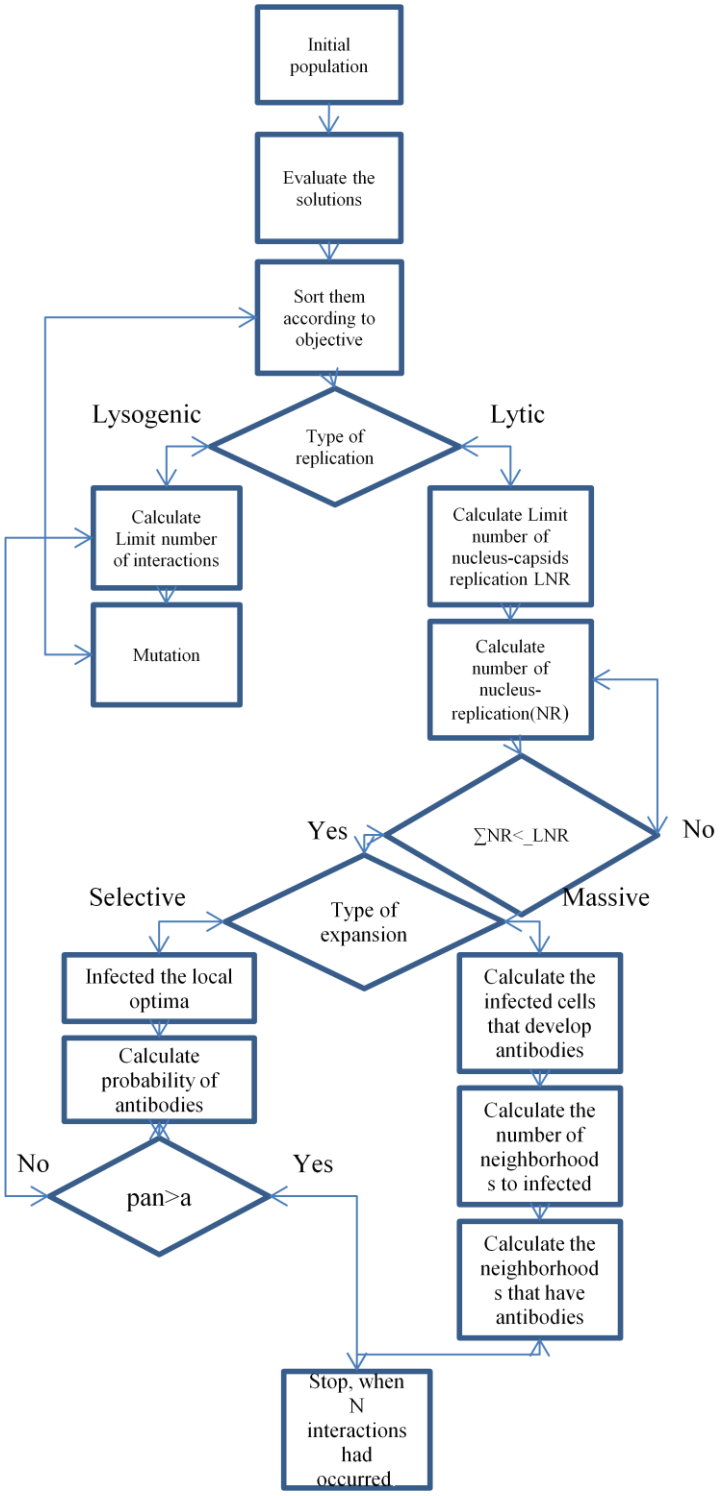


Figure 8: Flow diagram of Viral System algorithm (VSA).

The following steps illustrate the working mechanisms of the viral system algorithm:

1. Determine *n* number of initial viruses (initial population or solutions), and *N* number of interactions.
2. Evaluate each one of these solutions, to check if they are feasible.
3. Sort the solutions according to the objective(s), starting with the best found solution.
4. According to the order, divide the solutions depending on the type of replication, lytic or lysogenic ($p_{lt}+p_{lg}=1$), and order in which they are. The options in the first locations follow lytic replication, because they are in the active stage, and usually offer the best solutions. The remaining solutions follow a lysogenic replication because they are not immediately active and they do not offer the best solutions. In order to consider these viruses, they need to mutate in order for them to produce better solutions.
5. In case of Lytic replication:

   5.1. Calculate the limit number of nucleus-capsids replication ($LNR_x$). If the infected cell has a greater f(x) value its $LNR_x$ is smaller. In contrast, the infected cell with smaller f(x) has a greater $LNR_x$ value. This means that the infected cells with better solutions are going to evolution more quickly.

$$LNR_x = LNR_0\left(\frac{f(x)-f(\hat{x})}{f(\hat{x})}\right) \qquad \begin{array}{l} LNR_0 = iniial \quad value \quad of \quad LNR \\ x = the \quad \text{inf} ected \quad cell \\ \hat{x} = the \quad best \quad know \quad result \end{array} \tag{18}$$

   5.2. Then, calculate the number of nucleus replicated (NR) that takes placed in this interaction, multiplying the Limit nucleus-capsid, $LNR_x$ by the single probability of one replication, *pr*.

$$NR(x) = LNR_x \quad pr \tag{19}$$

   5.3. Check that the total number of nucleus replicated is greater than limit number of nucleus-capsids replication, if this condition is satisfied; the cell is active and can start the expansion process.

$$\sum_{i=1}^{n} NR \geq LNR_x \tag{20}$$

   5.4. Determine what of the following expansions is required:

      5.4.1 In case of Selective expansion:

         5.4.1.1 The active viruses search in their neighborhood for the local optima, which are the one closest to the best know solution.

5.4.1.2 Analyze the neighborhoods, to see the antigenic response to the infection.

5.4.1.3 Calculate the probability of creating antibodies of each neighbor (*pan),* taking in consideration *q* the probability for the worst individual.

$$pan(x) = q \quad (1-q)^{i} \tag{21}$$

5.4.1.4 Add the residual of the difference between the probability of generating antibodies of the local optima and worst individual. To increase the probability of the worst individual for the next interaction; this can be used as a stopping criteria.

5.4.1.5 If the probability of creating antibodies is greater or equal to that antibodies probability *an*, then this cell follow the lysogenic replication.

5.4.2 In case of Massive expansion:

5.4.2.1 Calculate the number of cells that are already infected and that developed antigenic response, *n*. Then, eliminate them from the population.

$$AR_{x} = pan \quad (n) \tag{22}$$

5.4.2.2 Determine the number of neighborhoods that are going to be infected (IC) by the active ones, this is the number of cells in the neighborhood that are part of the next population. This as result of product of neighborhood space *ne* and the probability of infection *pi*.

$$IC = pi \quad (ne) \tag{23}$$

5.4.2.3 Calculate the number of neighborhoods that have created antibodies (ARN), as result they are not going to be infected.

$$ARN = pan \quad (ne) \tag{24}$$

6. In case of Lysogenic replication:

6.1 Calculate the limit number of interaction (LIT) that each cell is following a lysogenic cycle.

$$LIT_{x} = LIT_{0}\left(\frac{f(x) - f(\hat{x})}{f(\hat{x})}\right) \tag{25}$$

$LIT_{0} = iniial \quad value \quad of \quad LIT$
$x = the \quad \text{inf} ected \quad cell$
$\hat{x} = the \quad best \quad know \quad result$

6.2 At the time the cell wakes up, mutate it. A change in the encoding can result in good solutions.

6.3 Return the new mutated cell to the initial process.

7. Stop, at the time that *N* interactions had occurred.

# Chapter 5: Application to the Multi-objective Redundancy Allocation Problem

After the Viral System algorithm was modeled, as Chapter 4 shows, VSA was adapted to solve multi-objective combinatorial optimization problems, such as the multi-objective redundancy allocation problem. To show the Viral System algorithm's steps a small example was formulated that consists of a system with three subsystems in series and a limit of three redundant components for each subsystem as is illustrated in Figure 9. In addition, each subsystem has four different alternative choices with the present characteristics show in Table 1.

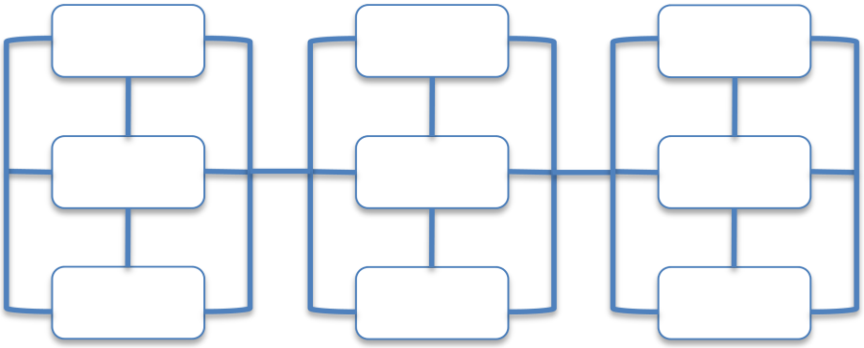

Figure 9: Small example series-parallel system.

Table 1: Alternative choice's characteristics.

| Alternative Design | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| j | R | C | W | R | C | W | R | C | W |
| 1 | 0.85 | 6 | 6 | 0.8 | 2 | 4 | 0.79 | 3 | 2 |
| 2 | 0.81 | 4 | 4 | 0.83 | 4 | 4 | 0.84 | 6 | 3 |
| 3 | 0.83 | 5 | 6 | 0.84 | 5 | 6 | 0.85 | 5 | 5 |
| 4 | 0.79 | 2 | 3 | 0.85 | 6 | 7 | 0.83 | 4 | 4 |

The problem is finding the appropriate number of redundant components and the alternative choices of each subsystem in order to maximize the system reliability and minimize the total cost and weight, with the constraint that the limit number of redundant components for each subsystem is 3. The problem formulation is the following:

$$\max\left[\prod_{i=1}^{s} R_I(x_i)\right], \min\left[\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}x_{ij}\right], \min\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i} w_{ij}x_{ij}\right]$$

Subject to (26)

$$k_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{\max,i} \quad \text{for} \quad \forall i = 1,2,....,s$$

$$x_{ij} \in \{0,1,2,...\}$$

### 5.1 Initialization

The first step of the viral system algorithm is to create a random initial population; this is to create solutions that have diversity. This problem was created with a random initial population of ten solutions as in Figure 10. The way in which they are formulated is nine columns, the numbers in the initial three columns are the alternative choices of the subsystem one, and these columns are followed by subsystem two and three respectively. A cero in a column represents that no element is in this position. As an example in solution 1, the subsystem one has three redundant components and the alternative choices are two, one and three. Subsystem two has three redundant components with alternative choices four, two and three. The subsystem three has two redundant components with three alternative choices.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 | 2 | 3 | 3 | 3 | 0 |
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 |
| 3 | 1 | 1 | 1 | 2 | 3 | 0 | 2 | 3 | 4 |
| 4 | 4 | 2 | 3 | 1 | 3 | 3 | 4 | 3 | 0 |
| 5 | 1 | 2 | 0 | 3 | 2 | 3 | 4 | 3 | 2 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 |
| 7 | 2 | 1 | 3 | 0 | 2 | 3 | 4 | 2 | 3 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 |

Figure 10:Random initial population.

### 5.2 Evaluation

Figure 11 illustrates the series-parallel system solution 1 and the alternative choice's characteristics of it. Followed by the creation of the initial random population, the solutions are

evaluated according to the objectives: maximize system reliability, minimize cost and weight. The formulas to calculate total system reliability, weight and cost are show in eq. 27, eq. 28, and eq. 29. For example, Figure 12 shows how is evaluated the solution 1.

$$\sum_{i=1}^{s}\sum_{j=1}^{n} C_{ij} \tag{27}$$

$$\sum_{i=1}^{s}\sum_{j=1}^{n} W_{ij} \tag{28}$$

$$1-\prod_{i=1}^{s}\prod_{j=1}^{n}\left[1-R_{ij}\right] \tag{29}$$

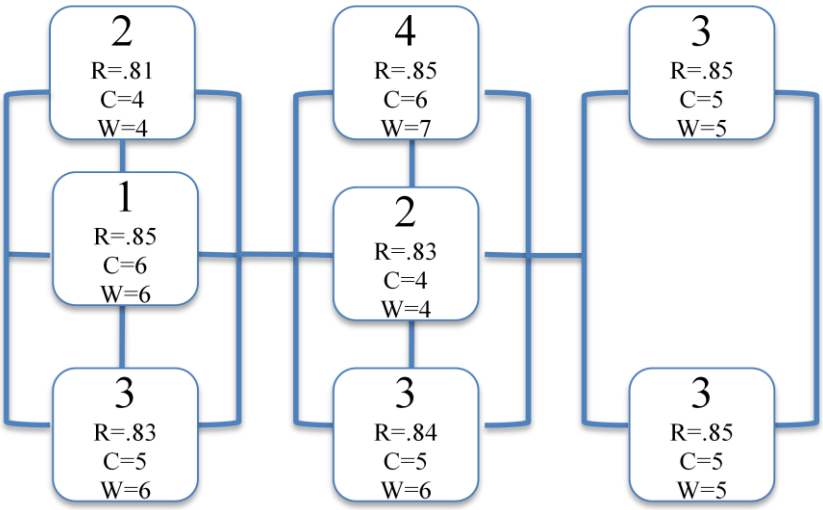


Figure 11. Solution 1's system.

$$1-\prod_{i=1}^{s}\prod_{j=1}^{n}\left[1-R_{ij}\right] = 1-[((1-.81)*(1-.85)*(1-.83))*((1-.85)*(1-.83)*(1-.84))*((1-.85)*(1-.85))] = .99999955$$

$$\sum_{i=1}^{s}\sum_{j=1}^{n} C_{ij} = (4+6+5) + (6+4+5) + (5+5) = 40$$

$$\sum_{i=1}^{s}\sum_{j=1}^{n} W_{ij} = (4+6+6) + (7+4+6) + (5+5) = 43$$

Figure 12: Solution 1's evaluation.

Table 2: Initial population solutions' reliability, cost, and weight.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Reliability | Cost | Weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 | 2 | 3 | 3 | 3 | 0 | 0.999999555 | 40 | 43 |
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 0.999999367 | 36 | 36 |
| 3 | 1 | 1 | 1 | 2 | 3 | 0 | 2 | 3 | 4 | 0.999999625 | 42 | 40 |
| 4 | 4 | 2 | 3 | 1 | 3 | 3 | 4 | 3 | 0 | 0.999999114 | 36 | 38 |
| 5 | 1 | 2 | 0 | 3 | 2 | 3 | 4 | 3 | 2 | 0.999999494 | 39 | 38 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 0.999998893 | 30 | 33 |
| 7 | 2 | 1 | 3 | 0 | 2 | 3 | 4 | 2 | 3 | 0.999999462 | 39 | 38 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 0.999999899 | 41 | 40 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 0.999999789 | 35 | 36 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 0.999999249 | 35 | 35 |

### 5.3 The multi-objective evaluation

We solved the multi-objective redundancy allocation problem with three objectives in conflict (maximize total system reliability, minimize total cost and weight), so is needed to found an evaluation method. Three fitness metrics are implemented to evaluate the initial population; non-dominance count, distance and aggregated fitness.

#### 5.3.1 Non-dominance selection

The first step before calculating the three metrics is the localization of the non-dominant solutions. To be a non-dominance solution one of the two following conditions need to be satisfied: 1) the solution x is no worse than $x_2$ in all objectives or 2) $x_1$ is strictly better than $x_2$ in at least one objective. If a tie happens (both are not dominated or both are dominated) then select one that has more accumulated restrictions As Figure 13 shows, each solution was compared against all the others population solutions to check if it was a dominance or non-dominance solution. The solutions in blue are the non-dominance solutions and the ones in yellow are the dominance solutions.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Reliability | Cost | Weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 | 2 | 3 | 3 | 3 | 0 | 0.999999555 | 40 | 43 |
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 0.999999367 | 36 | 36 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 1 | 1 | 1 | 2 | 3 | 0 | 2 | 3 | 4 | 0.999999625 | 42 | 40 |
| 4 | 4 | 2 | 3 | 1 | 3 | 3 | 4 | 3 | 0 | 0.999999114 | 36 | 38 |
| 5 | 1 | 2 | 0 | 3 | 2 | 3 | 4 | 3 | 2 | 0.999999494 | 39 | 38 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 0.999998893 | 30 | 33 |
| 7 | 2 | 1 | 3 | 0 | 2 | 3 | 4 | 2 | 3 | 0.999999462 | 39 | 38 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 0.999999899 | 41 | 40 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 0.999999789 | 35 | 36 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 0.999999249 | 35 | 35 |

Figure 13: Dominance and non-dominance solutions.

### 5.3.2 Fitness metric 1

The first fitness is a dominance count-based metric, which is the number of solutions that are dominated by each non-dominated solution. The dominance count-based metric is used to find proximity of the non-dominated solutions to the true Pareto front. The non-dominated solutions that dominated more solutions of the entire initial random solution are more close to the Pareto front, than the ones with lower number of dominated solutions. When the dominance count-based is finished, the intervals of dominance count for the fitness values is determined. As the count number increases, increase the fitness value. In Figure 14 the dominance counts of each nondominance solution is calculated; these are in the last column.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Reliability | Cost | Weight | Dominance count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 | 2 | 3 | 3 | 3 | 0 | 0.999999555 | 40 | 43 | |
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 0.999999367 | 36 | 36 | 1 |
| 3 | 1 | 1 | 1 | 2 | 3 | 0 | 2 | 3 | 4 | 0.999999625 | 42 | 40 | |
| 4 | 4 | 2 | 3 | 1 | 3 | 3 | 4 | 3 | 0 | 0.999999114 | 36 | 38 | |
| 5 | 1 | 2 | 0 | 3 | 2 | 3 | 4 | 3 | 2 | 0.999999494 | 39 | 38 | |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 0.999998893 | 30 | 33 | 0 |
| 7 | 2 | 1 | 3 | 0 | 2 | 3 | 4 | 2 | 3 | 0.999999462 | 39 | 38 | |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 0.999999899 | 41 | 40 | 1 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 0.999999789 | 35 | 36 | 6 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 0.999999249 | 35 | 35 | 1 |

Figure 14: The dominance count-based metric.

When the dominance count metric is finished, the fitness value is determined according to the interval of dominance count solutions. The solution with greater number of dominance solutions is going to have a greater fitness value because it is more close to the Pareto front. For this example five fitness values were assigned, and the intervals are determined by the range of the dominance count of the non-dominance solutions, this can be seen in Figure 15. In this example the width of each interval is 1.2 dominance solutions.

| Interval | Fitness Value |
|---|---|
| 0≤ dc <1.2 | 1 |
| 1.2≤ dc <2.4 | 2 |
| 2.4≤ dc <3.6 | 3 |
| 3.6≤ dc <4.8 | 4 |
| 4.8≤ dc ≤6 | 5 |

Figure 15: Fitness value 1's intervals.

Then, according to the interval table, assign the fitness value 1 for each non-dominance solution. Solutions two, six, eight, and ten have a fitness value of 1 and solution nine has a fitness value of five because it is close to the true Pareto front; as seen in Table 3.

Table 3: Nondominance solutions' fitness value 1.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Dominance count | Fitness Value 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 1 | 1 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 0 | 1 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 1 | 1 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 6 | 5 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 1 | 1 |

### 5.3.3 Fitness metric 2

The second fitness metric is a distance-based evaluation to insure diversity of solutions. In this metric the solutions with greater distance are explored, this is because other solutions with good distance are going to be far away; this ensures the search space is covered. As a result, the solutions that are going to have faster evolution are going to explore different directions and we are going to be looking for a global optimal instead of a local optimal. To determine the fitness value 2, first normalize all the objectives in order to have the non-dominance solutions in the same terms. As in Figure 16 shows, the non-dominance solutions are normalized to the objectives.

| Solution | Reliability | Normalization | Cost | Normalization | Weight | Normalization |
|---|---|---|---|---|---|---|
| 2 | 0.999999367 | 0.47151277 | 36 | 0.45454546 | 36 | 0.571429 |
| 6 | 0.999998893 | 0 | 30 | 1 | 33 | 1 |
| 8 | 0.999999899 | 1 | 41 | 0 | 40 | 0 |
| 9 | 0.999999789 | 0.89071906 | 35 | 0.54545455 | 36 | 0.571429 |
| 10 | 0.999999249 | 0.35363458 | 35 | 0.54545455 | 35 | 0.714286 |

Figure 16: Normalization.

Then the Euclidean distance is evaluated from each solution; to find the solution's distance, the distance of each solution are added to the others, and then the minimum and maximum distance are used to determine the width distance of each interval. As Figure 18 shows, the width distance is .432508 having five ranges. The solutions with greater distance are going to obtain better fitness value because our purpose in this fitness evaluation is to find the solutions with more distance to the others to cover all the search space.

| Solution | 2 | 6 | 8 | 9 | 10 |
|---|---|---|---|---|---|
| 2 | 0 | 0.83876005 | 0.90135506 | 0.42895032 | 0.20631988 |
| 6 | 0.83876005 | 0 | 1.73205081 | 1.08796391 | 0.6428854 |
| 8 | 0.90135506 | 1.73205081 | 0 | 0.79749207 | 1.10702891 |
| 9 | 0.42895032 | 1.08796391 | 0.79749207 | 0 | 0.55575885 |
| 10 | 0.20631988 | 0.6428854 | 1.10702891 | 0.55575885 | 0 |
| Euclidean distance | 2.37538531 | 4.30166017 | 4.53792684 | 2.87016515 | 2.51199304 |

Figure 17: Euclidean distance.

| Interval | Fitness Value |
|---|---|
| 2.37538531≤ ed <2.80789361 | 1 |
| 2.80789361≤ ed <3.24040192 | 2 |
| 3.24040192≤ ed < 3.67291022 | 3 |
| 3.67291022≤ ed <4.10541853 | 4 |
| 4.10541853≤ ed ≤4.53792684 | 5 |

Figure 18: Fitness value 2's intervals.

Finally, as the Euclidean distance of each solution to the other solutions are calculated, looking at the interval table, with the fitness value 2 of each solution, can be determined. As Figure 19 illustrates, solution two and ten have a fitness value of 1, solution nine has fitness value of 2, and solutions six and eight have a fitness value five. In conclusion, solutions six and eight have a greater fitness value 2, while also having a good fitness value 1; these solutions are going to be selected to expand in their neighborhood.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Euclidean distance | Fitness Value 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 2.37538531 | 1 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 4.30166017 | 5 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 4.53792684 | 5 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 2.87016515 | 2 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 2.51199304 | 1 |

Figure 19: Solution's fitness value 2.

### 5.3.4 Aggregation fitness

The last fitness value calculated is the aggregated fitness value. This is the summation of the two fitness metrics already evaluated (dominance count, and Euclidean distance). The solutions with the

greater number of aggregated fitness value are the ones that are closest to the true Pareto front and with greater distance to the others solutions, which insure the diversity of solutions. Finally, in Figure 20 the fitness value 1 and 2 are aggregated to determine the total fitness value of each solution.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Fitness Value 1 | Fitness Value 2 | Aggregation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 1 | 1 | 2 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 1 | 5 | 6 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 1 | 5 | 6 |
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 5 | 2 | 7 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 1 | 1 | 2 |

Figure 20: Solution's total fitness.

### 5.4 Replication selection

The following step of the Viral System Algorithm is to rank the solutions according to our objectives, then in this case we are going to maximize the total fitness value. After the multi-objective evaluation from our initial random population of ten solutions, only five non-dominance solutions are obtained and these solutions are evaluated for the two fitness metrics. These five solutions are ranked according to their total fitness value as in Figure 21 shows.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Fitness Value 1 | Fitness Value 2 | Aggregation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 5 | 2 | 7 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 1 | 5 | 6 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 1 | 5 | 6 |
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 1 | 1 | 2 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 1 | 1 | 2 |

Figure 21. Rank solutions.

Then determine which type of replication (lytic or lysogenic) the solution is going to follow. The solutions in the first places are going to follow lytic replication, because lytic replication has a faster evolution than lysogenic replication. And the solutions in last places are going to follow a lysogenic

replication. The method to determine how many are in lytic and lysogenic replication is by the product of the number of solutions in the initial population by the lytic's or lysogenic's probability. As a result the summation of the lytic and lysogenic replication's probabilities is one, as in eq. 30 shows.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Aggregation Fitness | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 7 | Lytic replication |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 6 | |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 6 | |
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 2 | Lysogenic replication |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 2 | |

Figure 22: Solutions in lytic and lysogenic replication.

$$p_{\lg} + p_{lt} = 1 \tag{30}$$

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Aggregation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 7 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 6 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 6 |

Figure 23: Solutions with lytic replication.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 2 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 2 |

Figure 24: Solutions with lysogenic replication.

### 5.4 .1 Lytic replication

In case of lytic replication first calculate the Limit number of nucleus-capsid replication for each solution using eq. 31, where LNR is the limit number of replication that can occur inside the bacterium before bacterium border is broken down and releases the new virus to infect one or more neighbors; this is dependent in the type of expansion that is taking place. The LNR of a solution depends in how close or far is the solution from the best solution known until now. Then the number of nucleus-capsid

replications that occur in this iteration, eq. 32, is calculated. This is the number of nucleus replication that occurs during this iteration. Finally, the limit number of nucleus-capsids replication and the summation of the already nucleus replicated are compared, eq. 33, and in case of having greater or equal nucleus replicated, the solution is going to be an active virus, which means that this solution is going to search in its neighborhood to select one or more neighbors to infect.

$$NR(x) = LNR_x \quad pr \tag{31}$$

(32)

$$\sum_{i=1}^{n} NR \geq LNR_x \tag{33}$$

As Figure 25 shows, the Limit number of nucleus-capsids replication and the number of nucleus replication had been calculated, followed by the evaluation to determine the active virus. Solution nine is the best know solution known until now, so it passes automatically to the expansion process.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Aggregation Fitness | LNR | NR | ∑NR≥LNR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 7 | 0 | 0 | YES |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 6 | 2 | 1 | NO |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 6 | 2 | 1 | NO |

Figure 25: LNR & NR calculation and comparison.

#### 5.4.1.1 Selective expansion

As was explained in chapter 4 two types of expansion occurs: selective and massive. In the selective expansion only one neighbor is infected, and as result this expansion is slower that massive expansion. In the example used, the solution nine is an active and selective virus, so it is going to infect one neighbor. The neighborhood's creation is taking one random element of the solution's configuration and changes it for the other alternative choices existing. The element solution's configuration that was taken in this example is the second position of the subsystem two; the alternative choices are one, three and four. When the neighborhood is created, the neighbor's solutions are evaluated according to the

fitness metrics used for the initial population's evaluation. After obtaining the total fitness values for each neighbor, the one with greatest value is chosen and it is going to be part of the new population, as in Figure 26 and 27 is illustrated.

| 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|

Figure 26: Active virus.

| Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Reliability | Cost | Weight | Aggregation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 1 | 3 | 4 | 1 | 2 | 0.99999975 | 33 | 36 | 10 |
| 4 | 2 | 3 | 1 | 3 | 3 | 4 | 1 | 2 | 0.9999998 | 36 | 38 | 6 |
| 4 | 2 | 3 | 1 | 4 | 3 | 4 | 1 | 2 | 0.99999981 | 37 | 39 | 3 |

Figure 27: Neighbor's total fitness values.

In the host, the organism produces the antigenic response which is the creation of antibodies to attack the virus. In order to see if the neighbors have created antibodies, the probability of the host creating them is calculated according to the eq. 34. Then the probability of creating antibodies is compared with the limit of creating of antibodies. In the case of a greater probability of creating antibodies than the limit, the host has created antibodies and it is going to follow a lysogenic replication in the VSA.

$$pan(x) = q \quad (1-q)^i \tag{34}$$

| Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Total Fitness | Rank | pan(x) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 1 | 3 | 4 | 1 | 2 | 10 | 3 | 0.1029 |
| 4 | 2 | 3 | 1 | 3 | 3 | 4 | 1 | 2 | 6 | 2 | 0.147 |
| 4 | 2 | 3 | 1 | 4 | 3 | 4 | 1 | 2 | 3 | 1 | 0.21 |

Figure 28: Neighbor's probability of antibodies.

#### 5.4.1.2 Massive expansion

The first step in the massive expansion is to find which solutions in the lytic replication have already created antibodies; to determine if it has use eq. 35. This equation illustrates this number as the product of the probability of creating antibodies by the number of solutions in the lityc replication. In

our case we found one solution with antibodies, so the last solution in the rank position is eliminated because it has a lower probability of finding good solutions.

$$AR_x = pan \quad (n) = .3*3 = 1 \tag{35}$$

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Total Fitness |
|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 7 |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 6 |
| 8 | 0 | 1 | 2 | 1 | 2 | 2 | 3 | 4 | 2 | 6 |

Figure 29: Solutions with antibodies.

Then to the other solutions the limit number of nucleus-capsids replication and the number of replications that occur in this iteration is calculated. In lytic replication, the limit number of nucleus-capsids replication is based in how far or close each solution is to the best known solution. Then the limit number of replication and the nucleus replicated are compared. The solutions with equal or greater number of replications are going to pass to the expansion process, which in this case is massive.

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Total Fitness | LNR | NR | ∑NR≥LNR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 4 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | 7 | 0 | 0 | YES |
| 6 | 4 | 2 | 3 | 1 | 1 | 0 | 2 | 3 | 4 | 6 | 2 | 1 | NO |

Figure 30: LNR& NR calculation and comparison in massive expansion.

As a result the solution nine is the best known solution until now, so it passes automatically to the expansion process. As seen before, the neighborhood is created choosing one configuration component and changed for the others alternatives. Then the neighbors are evaluated according to the fitness metrics already presented.

| Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Reliability | Cost | Weight | Aggregation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 1 | 3 | 4 | 1 | 2 | 0.99999975 | 33 | 36 | 10 |
| 4 | 2 | 3 | 1 | 3 | 3 | 4 | 1 | 2 | 0.9999998 | 36 | 38 | 6 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 4 | 3 | 4 | 1 | 2 | 0.99999981 | 37 | 39 | 3 |

Figure 31: Neighborhood.

The last step is to calculate the number of neighbors that are going to be infected, these are going to pass to the next population and the number of neighbors that have already created antibodies. These numbers are calculated using the following equations eq. 36 and eq. 37. As in the example, it is shown that two solutions are going to be infected and one solution is going to create antibodies, so it is eliminated.

$$IC = pi \quad (ne) = .75 * 3 = 2 \tag{36}$$

$$ARN = pan \quad (ne) = .25 * 3 = 1 \tag{37}$$

| Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Reliability | Cost | Weight | Aggregation Fitness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 1 | 3 | 4 | 1 | 2 | 0.99999975 | 33 | 36 | 10 |
| 4 | 2 | 3 | 1 | 3 | 3 | 4 | 1 | 2 | 0.9999998 | 36 | 38 | 6 |
| 4 | 2 | 3 | 1 | 4 | 3 | 4 | 1 | 2 | 0.99999981 | 37 | 39 | 3 |

Figure 32: Neighbor's infection and elimination.

### 5.4.2 Lysogenic replication

For the solutions that follow a lysogenic replication the number of iteration during these solutions will be dormant, this is determined by eq. 38, in which if the solution is closest to the best solution known, its limit number of iterations for the dormant virus is going to be less than the solutions far away from the best solution known. The two solutions that in our example problem follow a lysogenic replication are going to remain asleep until the three iterations occur.

$$LIT_x = LIT_0\left(\frac{f(\hat{x}) - f(x)}{f(\hat{x})}\right) \tag{38}$$

| Solution | Subsystem 1 | | | Subsystem 2 | | | Subsystem 3 | | | Aggregation Fitness | LIT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 0 | 2 | 3 | 1 | 4 | 3 | 2 | 2 | 3 |
| 10 | 2 | 2 | 0 | 1 | 2 | 4 | 4 | 2 | 3 | 2 | 3 |

Figure 33: LIT's calculation.

### 5.5 Mutation

When the limit number of iterations had passed, the dormant solution is going to mutate. In the mutation process, two configuration components are chosen randomly. Then, the mutated virus will cause a change from position in the solution, as on the Figure 34 shows. When the solution mutates, it goes back to the initial evaluation process.

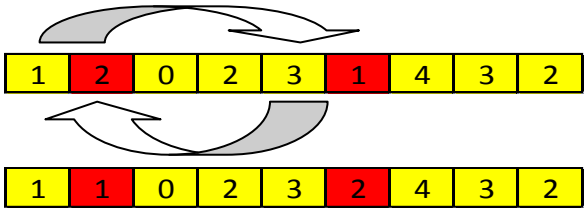


Figure 34: Mutation process.

# Chapter 6. Problem

After literature review is performed, an example of a well-known multi-objective redundancy allocation problem is taken to evaluate the performance of the Viral System Algorithm. The problem selected was solved using NSGA-II by Taboada & Coit (2007).

## 6.1 Problem Formulation

The multi-objective Redundancy Allocation Problem has the objective of founding the optimal design configuration that will maximize system reliability, minimize the total cost and minimize the system weight. In this case

$$\max\left[\prod_{i=1}^{s} R_I(x_i)\right], \min\left[\sum_{i=1}^{s}\sum_{j=1}^{m_i} c_{ij}\, x_{ij}\right], \min\left[\sum_{i=1}^{s}\sum_{j=1}^{n_i} w_{ij}\, x_{ij}\right]$$

Subject to

$$k_i \le \sum_{j=1}^{m_i} x_{ij} \le n_{\max,i} \qquad \text{for} \quad \forall i = 1,2,....,s \tag{39}$$

$$x_{ij} \in \{0,1,2,...\}$$

The example system configuration consists of 3 subsystems with different number of available options in each subsystem. Subsystems one, two, and three each have the possibility of 5, 4, and 5 design alternatives respectively; this is show in table 4. In order to make the system function, constraints with a maximum number of 8 components and a minimum of 1 is necessary. The optimization involves selection of the redundant levels and component choices for each subsystem.

Table 4: Subsystem components options.

| Design Alternative | Subsystem *i* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | | | 2 | | | 3 | | |
| *j* | *R* | *C* | *W* | *R* | *C* | *W* | *R* | *C* | *W* |
| **1** | 0.94 | 9 | 9 | 0.97 | 12 | 5 | 0.96 | 10 | 6 |
| **2** | 0.91 | 6 | 6 | 0.86 | 3 | 7 | 0.89 | 6 | 8 |
| **3** | 0.89 | 6 | 4 | 0.70 | 2 | 3 | 0.72 | 4 | 2 |
| **4** | 0.75 | 3 | 7 | 0.66 | 2 | 4 | 0.71 | 3 | 4 |
| **5** | 0.72 | 2 | 8 | | | | 0.67 | 2 | 4 |

### 6.2 Results of Multi-Objective RAP

The multi-objective redundancy allocation problem with three objectives was formulated with the following parameters:

Population size: 100

Number of iterations: 1000

Probability of infection: 30%

Probability of generating antigens: 0.6%

Probability of Lytic replication: 60%

Probability of Lysogenic replication: 40%

$LNR_0$: 5

$LIT_0$:1

These objectives maximize system reliability, minimize total cost and minimize system weight. The VS algorithm was used to solve the problem and 247 solutions were found in the Pareto set. The algorithm was configured using Matlab. The time than it takes to render 1000 iterations is one hour and a half. Figure 35 shows the entire Pareto-optimal set of solutions which are the non-nominated solutions. It is possible to see all the non-dominated solutions that are close to the optimal point (1, 0, 0); taking in consideration reliability of 1 and weight and cost of 0.

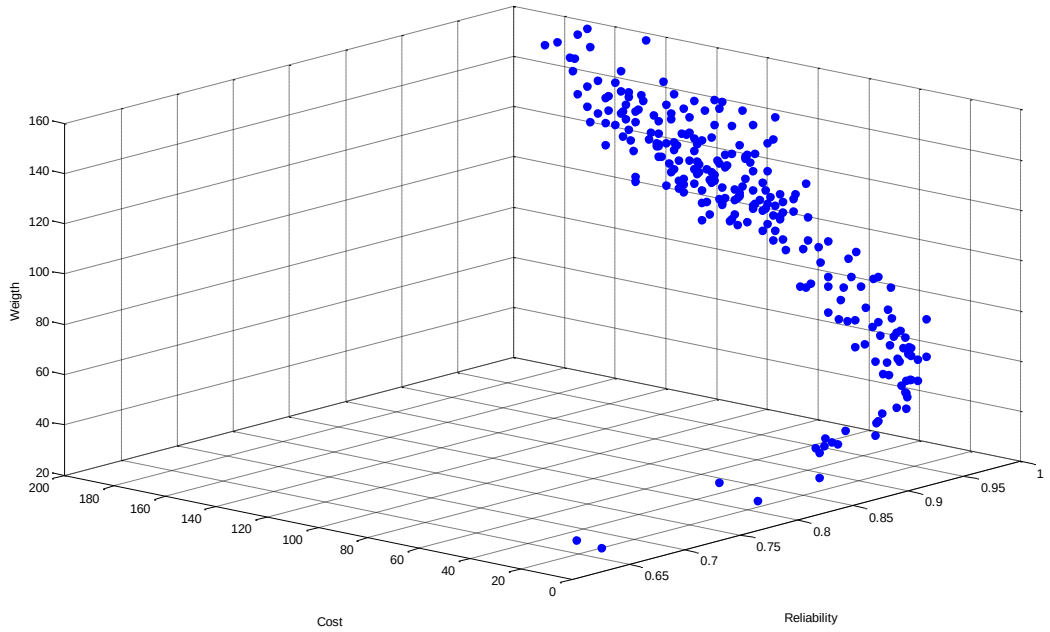


Figure 35 :Pareto-optimal set for the multi-objective RAP.

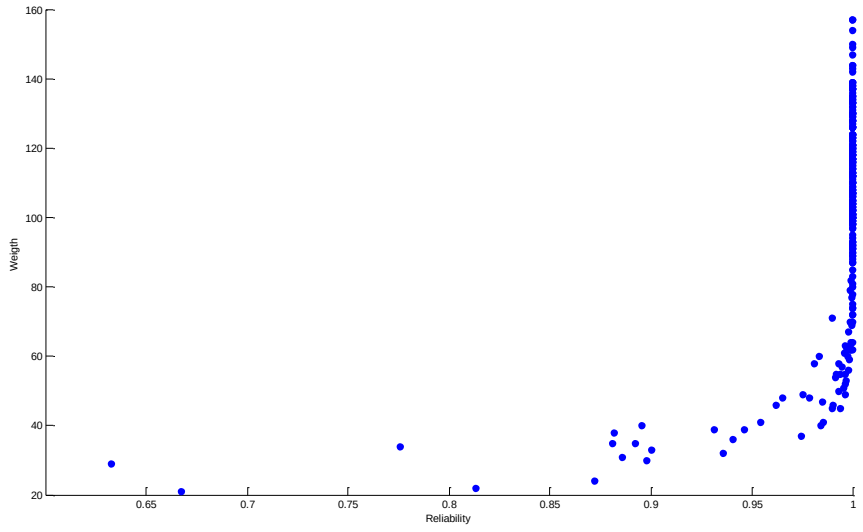


Figure 36:Two dimensional view of reliability vs. weight.

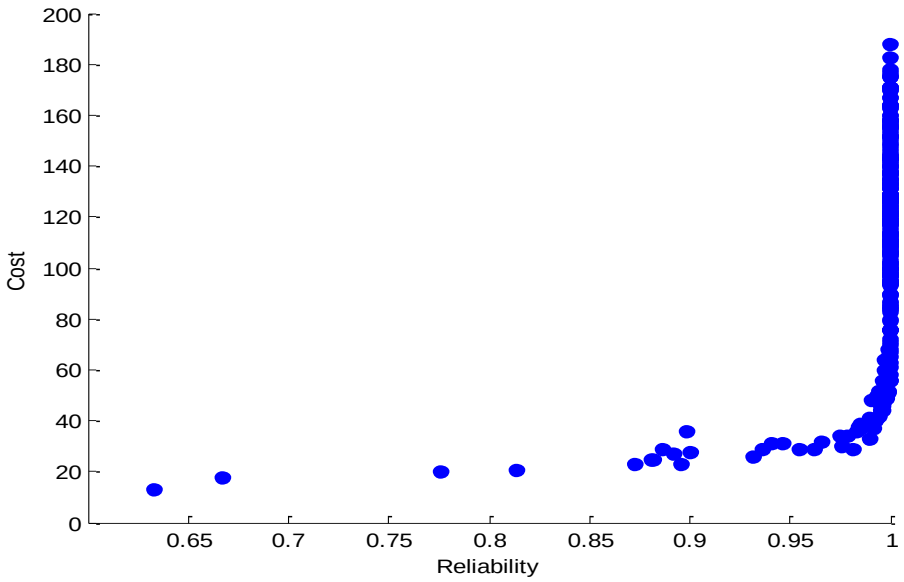


Figure 37 :Two dimensional view of reliability vs. cost.

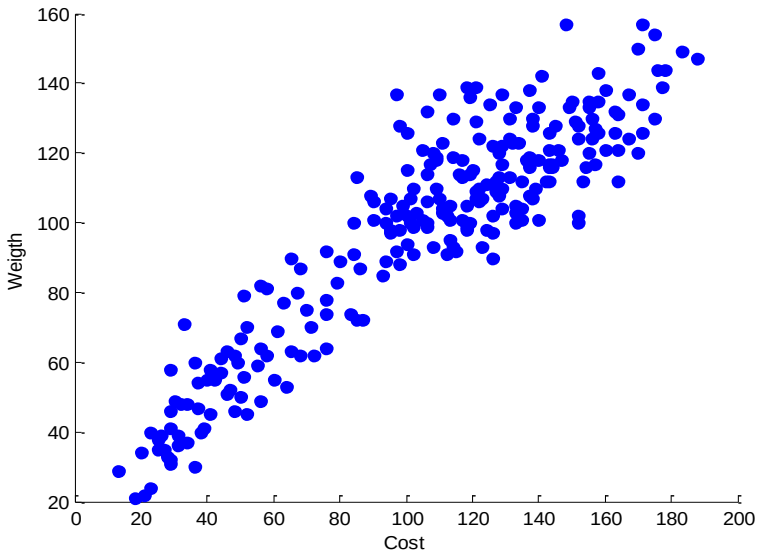


Figure 38: Two dimensional view of cost and weight.

The table below shows 20 example solutions of the Pareto optimal set. The configuration of the solutions are; the first eight columns are the component's combination of subsystem 1, followed by columns 9 to 16 that are the component's combination of subsystem 2 and columns 17 to 24 for the

subsystem 3; the system reliability, cost and weight are calculated to each solution. The optimal solution is found after a normalization process to the objective point (1, 0, 0) of the 247 configurations. The optimal solution found has a system reliability of .821433, cost of 21 and weight of 22, the structure of the solution is two elements in subsystem 1 with choice 3, two elements in subsystem 2 with choice 3 , and two elements in subsystem 3 with choices 5 and 4, as is show in Figure 39 .

Table 5: Example solution of the Pareto-optimal set.

| Subsystem 1 | | | | | | | | Subsystem 2 | | | | | | | | Subsystem 3 | | | | | | | | Reliability | Cost | Weight |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.974422454 | 34 | 37 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0.93565314 | 29 | 32 |
| 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0.954166467 | 29 | 41 |
| 5 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 5 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0.931429674 | 26 | 39 |
| 4 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.940697597 | 31 | 36 |
| 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 4 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0.984051741 | 38 | 40 |
| 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0.812955753 | 21 | 22 |
| 2 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0.946108288 | 31 | 39 |
| 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 4 | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0.985223874 | 39 | 41 |
| 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0.872057088 | 23 | 24 |
| 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 4 | 1 | 0 | 0 | 0 | 0 | 0.984821988 | 37 | 47 |
| 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 4 | 2 | 0 | 0 | 0 | 0 | 0.961830539 | 29 | 46 |
| 5 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 2 | 2 | 0 | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.978543388 | 34 | 48 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 5 | 5 | 3 | 3 | 0 | 0 | 0 | 0 | 0.900038218 | 28 | 33 |
| 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 3 | 3 | 0 | 0 | 4 | 4 | 3 | 3 | 0 | 0 | 0 | 0 | 0.989679502 | 41 | 45 |
| 5 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 5 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0.975275082 | 30 | 49 |
| 5 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0.775618623 | 20 | 34 |
| 5 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0.891964825 | 27 | 35 |
| 4 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 0.885883392 | 29 | 31 |
| 5 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0.880871804 | 25 | 35 |

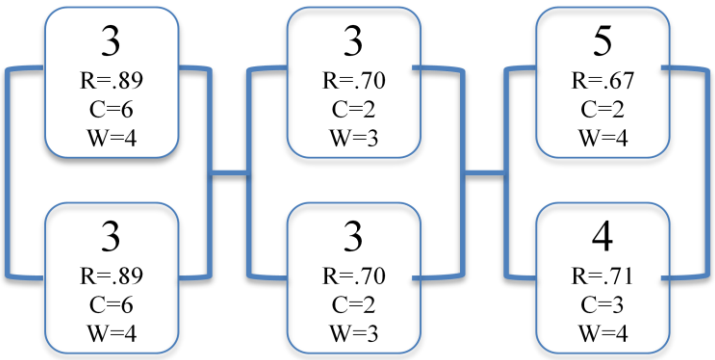


Figure 39: Optimal solution system structure.

In order to test the performance of the Viral System algorithm a visual comparison between the Pareto-optimal set found by the NSGA-II algorithm and the Pareto-optimal set obtained by the VSA is developed. As a result, Viral System algorithm shows superiority searching optimal solutions, NSGA-II found 75 solutions and VSA found 247 solutions. In addition, the VSA obtained more diversity of solutions and better spread than the multi-objective evolutionary algorithm (MOEA).

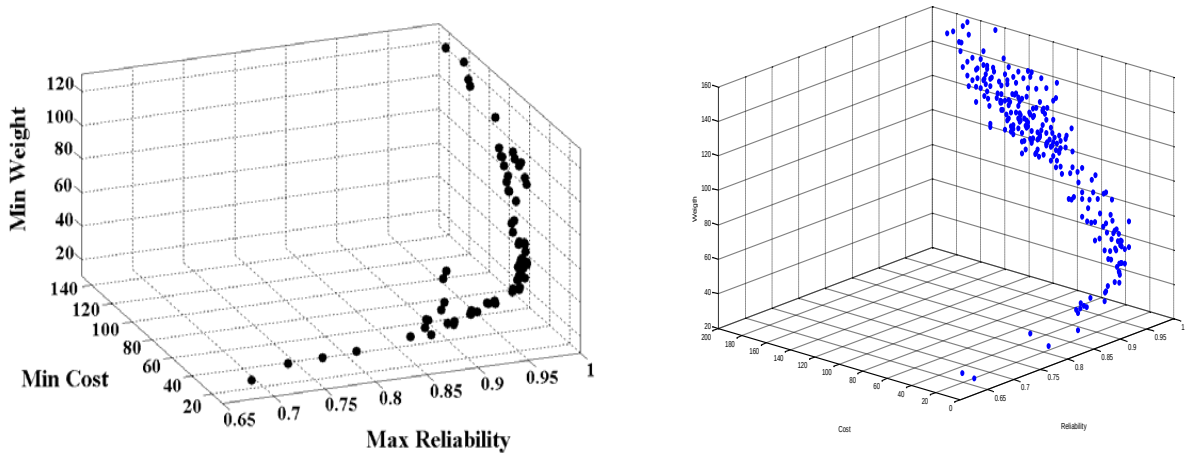


Figure 40 :Visual comparison of the NSGA-II and VSA Pareto-optimal sets.

# Chapter 7.Conclusion

To achieve the research objective of this thesis, developing a new algorithm to solve combinatorial optimization problems, few steps were followed. The first step was, as presented in Chapter 2, the understanding of the most common approaches developed until now. These approaches are divided in mathematical and metaheuristic approaches; the Viral System algorithm (VSA) following in the latter category. Afterwards, the Redundancy allocation problem (RAP) was used to evaluate the performance of the VSA. The methods proposed to solve the RAP as single or multi-objective problem are demonstrated in chapter 3. To the best of our knowledge, there has been no attempt to implement the VSA algorithm as a method of solving the multi-objective redundancy allocation problem (MO-RAP). Followed by this, chapter 4 presents the biological process that inspired the VSA and the procedures of the algorithm were explained. In chapter 5, the VSA was implemented to a small multi-objective redundancy allocation problem to illustrate how the solutions are evaluated with multiple objectives in conflict and the process, followed by these solutions, are explained for the VSA. The final step was to evaluate the VSA's performance using the MO-RAP; this problem was the same as H. Taboada & D. Coit solved in the NSGA-II algorithm.

In conclusion, the viral system algorithm was modified to solve multi-objective combinatorial optimization problems using coding in Matlab. To test the performance of the VSA, using the well-known redundancy allocation problem, we considered multiple objectives: to maximize the system reliability and to minimize the total cost and weight. For this reason we applied the MO-RAP to the VSA, a visual comparison was performed. Our findings revealed that the VSA was better in three points: first, more solutions where obtain in the Pareto front, NSGA-II found 75 and VSA 247.Second, these solutions had more diversity and thirdly, a better spread in the Pareto optimal front was found.

The contribution that this thesis brings to the Industrial Engineering community is the developing of a new meta-heuristic method to solve combinatorial problems such as the Redundancy

allocation problem. This was used to evaluate the algorithm performance and the algorithm provided a diverse Pareto optimal set of solutions to the user. Some of the future research opportunities are testing the VSA's performance in more combinatorial optimization problems and carry out a formal performance comparison.

# References


Bachlaus, M., Shukla, N., Tiwari, M., Shankar, R. 2006: Optimization of system reliability using chaos-embedded self-organizing hierarchical particle swarm optimization. *Proceedings of the Institution of Mechanical Engineers.* **220**(2), 77-91.

Beji, N., Jarboui, B., Eddaly, M., Chabchoub, H. 2010: A hybrid particle swarm optimization algorithm for the redundancy allocation problem. *Journal of Computational Science.* 1(3), 159-167.

Bullnheimer, B., Hartl, R., Strauss, C. 1996: A new rank-based version of the ant system: a computational study. *Central European Journal for Operations Research and Economics.* **7**(1), 25-38.

Bullnheimer, B., Hartl, R., Strauss, C. 1999: An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research.* **89**, 319-328.

Castro, L., Zube, F. 2000: The clonal selection algorithm with engineering applications. *Genetic and Evolutionary Computation Conference (GECCO'00)*- Workshop Proceeddings, Las Vegas, Nevada, USA.

Castro, L., Timmis, J. 2002: Artificial Immune Systems: A New Computational Intelligence Approach. *Springer*, 57-58

Charnes, A., Cooper, W., Ferguson, R. 1955: Optimal estimation of executive compensation by linear programming. Management Science. 1, 138-151.

Charnes, A., Cooper, W. 1977: Goal programming and multiple objective optimization; part 1. Eur. J. Oper. Res. 1, 39-45.

Chen, T., You, P. 2006 : Immune algorithms-based approach for redundant reliability problem with multiple component choices. *Computers in Industry.* **56**, 195-205.

Chen, T. 2006: IA's based approach for reliability redundancy allocation problems. *Applied*

*Mathematics and Computation.* **182**, 1556-67.

Chen, T. 2006 c: Penalty guided PSO for reliability design problems. PRICAI 2006: Trends in artificial intelligence. *Lecture Notes in Computer Science*. **4099/2006**, 777-786.

Chern, M., Jan, R. 1985: Parametric Programming Applied to Reliability Optimization Problems. *IEEE Transactions on Reliability.* **34**(2), 165-170.

Chern, M., Jan, R. 1986: Reliability Optimization Problems with Multiple Constraints. *IEEE Transactions on Reliability*. **35**(4), 431-436.

Clerc, M. 1999: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proc. 1999 Congress on Evolutionary Computation*, Washington, DC. 1951-57.

Coelho, L.2009: An efficient particle swarm approach for mixed-integer programming in reliability-redundancy optimization applications. *Reliability Engineering and System*

Coelho, L. 2009: Reliability-redundancy optimization by means of a chaotic differential evolution approach. *Chaos Solitons & Fractals.* **41**, 594-602.

Coit, D. , Smith, A. 1996: Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. *Computers & Operations Research.* **23**(6), 515:526.

Coit, D., Smith, A. 1996: Penalty Guided Genetic Search for Reliability Design optimization. *Computers industrial Engineering*. **30** (4), 895-904.

Coit, D., Smith, A. 1996 c: Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability.* **45**(2), 254-266.

Coit, D., Kona, A. 2006: Multiple Weighted Objectives Heuristic for the Redundancy Allocation Problem. *IEEE Transactions on Reliability*. **55**(3), 551-558.

Colorni, A. , Dorigo, M., Maniezzo, V., Trubian, M. 1994: Ant system for job-shop scheduling. *JORBEL- Belgian Journal of Operations Research, Statistics and Computer Science.*

**34**(1), 39-53.

Cortes, P., Garcia, J., Muñuzuri, J., Onieva, L. 2008: Viral systems: A new bio-inspired optimisation approach. *Computers & Operations Research*. **35** , 2840-60

Cortes, P., Garcia, J., Onieva, L., Muñunuzuri, J., Guadix, J. 2008: Viral System to Solve Optimization Problems: An Immune-Inspired Computational Intelligence Approach .Artificial Immune Systems. *7th International Conference, ICARIS.* 83-94.

Costa, D., Hertz, A. 1997: Ants can colour graphs. *Journal of the Operational Research Society.* **48**, 295-305.

Dasgupta, D. 2006: Advances in artificial immune systems. Computational Intelligence Magazine, IEEE. 1(4), 40-49.

Dasgupta, D., Yu, S., Nino, F. 2010: Recent Advances in Artificial Immune Systems: Models and Applications. *Applied Soft Computing*. **11**(2), 1574-87.

Ded, K. 2001: Chapter 2 Multi-objective optimization. Multi-objective optimization using evolutionary algorithms. Wiley :13-29

Deneuboung, J., Aron, S., Goss, S., Pasteels, J. 1990: The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*. 3(2), 159-168

Dhingra, M. 1992: Optimal Apportionment of Reliability Redundancy in Series Systems Under Multiple Objectives. *IEEE Transactions on Reliability.* **41**(4), 576-582.

Dorigo, M. 1992:Optimization, learning and natural algorithms. Ph D thesis, *Politecnico di Milano*, Italie.

Dorigo, M., Maniezzo, V. 1996: Ant system: optimization by a colony of cooperating agents. *IEEE Transaction on Systems, Man, and Cybernetics –Part B.* **26**(1), 29-41.

Dorigo, M., Gambardella, L. 1997: Ant colony system: a cooperating learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation.* **1**(1), 53-

66.

Eberhart, R., Shi, Y. 2001: Particle Swarm Optimization; Developments, Applications and Resources, *Proceedings of the 2001 Congress on Evolutionary*, 81-86

Farmer, J., Packard, N., Perelson, A. 1986: The immune system, adaption and machine learning. *Physica D*. **2**, 187-204

Fishburn, P. 1967: Additive utilities with incomplete product set: applications to priorities and assignments. *Operations Research Society of America CORSA,* Baltimore, MD. , USA.

Forrest, S., Pereson, A., Allen, L., Cherukui, R. 1994: Self-nonself discrimination in a computer. Proc. *IEEE Symposium on Research Security and Privacy* ,202-212.

Fourman, M. 1985: Compaction of symbolic layout using genetic algorithms. *Proc. of the 1st ICGA* ,141-153.

Fyffe, D., Hines, W., Lee, N. 1968: System Reliability Allocation and a Computational Algorithm. *IEEE Transactions on Reliability*. **17**(2), 64-69.

Gembick, F. 1974: Performance and sensitivity optimization: a vector index approach. Ph. D Dissertation. *Case Western Research University, Cleveland, OH.*

Gambardella, L., Taillard, E., Agazzi, G. 1999: A multiple ant colony system for vehicle routing problems with time windows. *New ideas in Optimization.* 63-76.

Golberg, D. 1989: Genetic Algorithms in Search, Optimization, and Machine Learning Addison-Wesley.

Hikita, M., Nakagawa, Y., Nakashina, K. Narihisa, H. 1992: Reliability Optimization of Systems by a Surrogate-Constraints Algorithms. *IEEE Transactions on Reliability*. **41**(3), 473-480.

Holland, J. 1975: Adaptation in Natural and Artificial Systems. *The University of Michigan Press, Ann Arbor.*

Hsieh, Y. ,Chen, T. , Bricker, D. 1998: Genetic algorithms for reliability design problems.

*Microelectronics Reliability*. 38, 1599-1605.

*Safety*. **94**, 830-837.

Hwang, C., Tillman, F., Kuo, W. 1979: Reliability Optimization by Generalized Lagrangian-Function and Reduced-Gradient Methods. *IEE Transactions on Reliability*. **28**(4), 316-319.

Ishida, Y. 1990 : Fully distributed diagnosis by PDP learning algorithm: towards immune network PDP Model. *IEEE International Joint Conference on Neural Networks*, San Diego, USA.

Jianping, L. 1996: A bound dynamic programming for solving reliability redundancy optimization. *Microelectronics and Reliability*. **36**(10), 151-20.

Jianping, L. 1996: A bound heuristic algorithm for solving reliability redundancy optimization. *Microelectronics and Reliability*. **36**(3), 335-339.

Kennedy, J.,Eberhart, R.1995 : Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*. **6**, 1942-48

Kita, H., Yabumete, Y., Mori, N., Nishikawa, Y. 1995: Multi-objective optimization by means of the thermodynamical genetic algorithm. *Structural Optimization*. **10**(2), 94-99.

Kulturel-Konak, S., Smith, A., Coit , D. 2003: Efficiently Solving the Redundancy Allocation Problem Using Tabu Search. *IIE Transactions 2003.* **35**(6), 515-526.

Kulturel-Konak, S., Coit, D., Baheranwala, F. 2008: Pruned Pareto-optimal sets for the system redundancy allocation problem based on multiple prioritized objectives. *J. Heuristics.* 14, 335-357.

Kuo, W. Lin, H., Xu, Z. Zhang, W. 2009: Reliability Optimization with the Lagrange-Multiplier and Branch-and –Bound Technique. *IEEE Transactions on Reliability*. **R-36**(5), 624-630.

Kursawe , F. 1991 :A variant of evolution strategies for vector optimization. *Parallel Problem Solving from Nature,* 193-197.

Lessing, L., Dumitrescu, I., Stutzle, T. 2004: A comparison between ACO algorithms for the set covering problem. *ANTS 2004, fourth international workshop on ant algorithms and swarm intelligence.* **1**(12) Springer-Verlang.

Li, Z, Liao, H., Coit, D. 2009: A two-stage approach for multi-objective decision making with applications to system reliability optimization. *Reliability Engineering and Systems Safety.* **94** , 1585-92.

Liang, Y. , Smith A.1999: An Ant System Approach to Redundancy Allocation. *Proceedings of the 1999 Congress on Evolutionary Computation.* 1478-84.

Liang, Y. , Smith, A. 2004: An Ant Colony Optimization Algorithm for the Redundancy Allocation Problem (RAP). *IEEE Transactions on Reliability*. **53**(3), 417-423.

Liang, Y., Lo, M. 2010: Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm. *Jo Heuristics.* **16**, 511-535.

Mahapatra, G., Roy, T. 2006: Fuzzy multi-objective mathematical programming on reliability optimization model. *Applied Mathematics and Computation.* **174**, 643:659.

Manfrin, M., Birattari, M., Stutzle, T., Dorigo, M. 2006: Parallel ant colony optimization for the travelling salesman problem. *In Proceedings of ANT 2006.* 224-234.

Maniezzo, V., Colomi, A., Dorigo, M. 1994: The ant system applied to the quadratic assignment problem. Technical Report IRIDIA , 9428.

Marlen, R., Aurora, J. 2004: Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*. **26**(6), 369-395

Marlen, R. Aurora, J. 2009: The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization.* **41**(6), 853-862

Mendes, R., Cortes, P.,Rocha, M., Neves, J. 2000: Particle swarms for feed forward neural net training. *In Proceedings of the international joint conference on neural networks.* 1995-99.

Misra, K. 1971: Dynamic Programming Formulation of the Redundancy Allocation Problem. *International Journal of Mathematical Education Science and Technology.* **2**(3), 207-215.

Misra, K., Sharma, U. 1991: An Efficient Algorithm To Solve Integer-Programming Problems Arising In Sytem-Reliability Design. *IEEE Transactions on Reliability*. **40**, 81-91.

Mullen, R., Monekosso, D., Barman, S., Remagino, P. 2009: Review: A review of ant algorithms. *Expert Systems with Applications.* **36**(6), 9608-17.

Nahas, N., Nourelfath, M. , Ait-Kadi, D. 2007: Coupling ant colony and the degraded ceiling algorithm for the redundancy allocation problem of series-parallel systems. *Reliability Enginnering and System Safety*, **92**, 211-22.

Nakagawa, Y., Miyazaki, S. 1981: Surrogate constraints algorithm for reliability optimization problem with two constraints. *IEEE Transactions on Reliability.* R30 (2), 175-180.

Ogryczak, W. 1994: A goal programming model of reference point method. *Ann. Operation Research.* **51**, 33-44.

Osyczka, A. 1984: Multicriterion Optimization in Engineering with Fortran Programs. *New York: John Wiley and Sons.*

Poli, R., Kennedy, J., Blackwell, T. 2007: Particle swarm optimization: On overview. *Swarm Intelligence.* **1** ,33-57

Prasad, U., Kuo, W.2000: Reliability Optimization of Coherent Systems. *IEEE Transactions on Reliability*. **49**(3), 323-330.

Princenton Dictionary http://wordnetweb.princeton.edu/perl/webwn?s=phage

Sakawa, M. 1981: Optimal Reliability-Design of a Series- Parallel System by a Large-Scale

Multiobjective Optimization Method. *IEEE Transactions on Reliability*. 30(2), 173-174.

Salazar, D., Rocco, C., Galvan, B. 2006: Optimization of constrained multiple-objective reliability problems using evolutionary algorithms. *Reliability Engineering and Systems Safety* . **91**, 1057-70.

Sawaragi, Y., Nakayama, H., Tanino, T. 1985: Theory of Multiobjective Optimization. *Mathematics in Science and Engineering* 176, Orlando FL. Academic Press inc.

Schaffer, J. 1985: Multiple objective optimization with vector evaluated genetic algorithms. *Proc. of the 1'st ICGA*, 93-100.

Shi, Y., Eberhart, R. 1998: Parameter selection in particle swarm optimization. *In Evolutionary Programming: VII: Proc. EP* **98**, 591-600.

Socha, K. 2004: ACO for continuous and mixed-variable optimization. In Ant colony optimization and swarm intelligence. *4th international workshop, ANTS 2004*. 25-36. Springer-Verlag.

Socha, K., Blum, C. 2007: An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. *Neural Computing and Applications.* **16**(3), 235-247.

Soylu, B., Ulusoy, S. 2011: A preference ordered classification for multi-objective max-min redundancy allocation problem. *Journal Computers & Operations Research.* **38**(12), 1855-66.

Stadler, W. 1988: Fundamentals of multicriteria optimization. In: *Stadler W. (ed.) Multicriteria Optimization in Engineering and in the Sciences.* 1-25. New York: Plenum Press.

Stutzle, T., Hoos, H. 2000: Max-min ant system. *Future Generation Computers Systems.* **16**(8), 889-314.

Taboada, H., Coit, D. 2007: Data Clustering of Solutions for Multiple Objective System Reliability Optimization Problems. *Quality Technology & Quantitative Management* . **4**(2), 191-210.

Talbi, E., Roux, O., Fonlupt, C., Robillard, D. 1999:Parallel ant colonies for combinatorial optimization problems. *In Parallel and distributed processing IPPS/SPDP-99.***1586**, 239-247.

Tamaki, H., Kita, H., Kobayashi, S. 1996: Multi-objective optimization by genetic algorithms: a review. *Proceedings of IEEE International Conference on Evolutionary Computation.*517-522.

Waltz, F. 1967: An engineering approach: hierarchical optimization criteria. *IEEE Transactions Automatization Control.* **AC-12**, 179-180.

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

Zavala, A. Villa, E., Hernandez, A. 2005: Particle Evolutionary Swarm for Design Reliability Optimization. *Lectures Notes in Computer Science.* **3410**: 856-869.

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

Zeleny, M. 1982: Multiple Criteria Decision Making. New York: Mc Graw Hill.

Zykina, A. 2003: A Lexicographic Optimization Algorithm: *Automation and remote control*. 65(3),363-368

# Curriculum Vita

Claudia Evangelina Valles Sosa was born in Hidalgo del Parral, Chihuahua, Mexico in 1987. Claudia received her bachelor's degree in Industrial Engineering in the University of Texas at El Paso in August 2009. Then she started her master's degree in Industrial Engineering, and worked in the Industrial, Manufacturing and Systems Engineering Department, with Dr. Heidi Taboada as Research Assistant and with Dr. Jaime Sanchez and Dr. Douglas Rittman as Teaching Assistant. She had the opportunity to present her research with the Viral System Algorithm in the 2010 Institute for Operational Research and Management Science (INFORMS) Conference in Austin, Texas and in the 2011 In Proceedings of the Industrial Engineering Research Conference (IERC) in Reno, Nevada.