2012-01-01

Designing Optimal Aviation Baggage Screening Strategies Using the Monkey Search Algorithm

Edgar Ivan Jimenez
University of Texas at El Paso, eijimenez@miners.utep.edu

DESIGNING OPTIMAL AVIATION BAGGAGE SCREENING STRATEGIES USING THE MONKEY SEARCH ALGORITHM

EDGAR JIMENEZ

Department of Industrial, Manufacturing, and Systems Engineering

APPROVED:

Jose F. Espiritu, Ph.D., Chair

Luis R. Contreras, Ph.D.

Salvador Hernandez, Ph.D.

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

For my family, friends and my future wife, for their friendship and support always

DESIGNING OPTIMAL AVIATION BAGGAGE SCREENING STRATEGIES USING THE MONKEY SEARCH ALGORITHM

By

EDGAR JIMENEZ ESTRADA

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 and Systems Engineering
THE UNIVERSITY OF TEXAS AT EL PASO
May 2012

ACKNOWLEDGEMENTS

Thanks to my mother Bertha Estrada, my brothers and my sister Lizzeth Jimenez to awake in me the interest to start this new challenge in my life and give the support to complete it. In addition, I want to acknowledge the support and motivation that my fiancée Teresa Curiel gave me to finish my Master's degree. When I started this journey, I never imagined that I could finish this new challenge in my life. I had considered a Master's degree as really good tool which will help me to get better opportunities in my life. At the beginning, I was living just with my sister in a new country with no friends and family, but it changed quickly. There were many people that gave me support, advices and guidance in this project; these persons are all my colleagues in the industrial laboratory. A very important person that gave me the opportunity to study my Master's degree is Dr. Heidi Taboada as well as my advisor Dr. Jose Espiritu. Currently that I am at the end of this work, I remember all the help that I received for all my professors, family, friends and my lovely fiancé. Finally, thanks to god for always support me and show me right way to do all the things.

ABSTRACT

This thesis addresses the aviation baggage screening design problem considering several baggage screening devices which may be used for system implementation, the devices have different false clear and false alarm rates, throughput and purchase costs. In the present research, a comprehensive cost function which not only includes the cost associated with purchase and operation of baggage security devices, but also includes the indirect costs associated with device errors is used. A new monkey search based evolutionary algorithm is presented to determine the best selection of baggage screening security devices in order to minimize the expected annual total cost. The final solution to the aviation baggage screening problem specifies the number and type of devices to be installed, an example applying the proposed method is presented using data from previous studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Currently, aviation security is a matter of concern for national security, due to the events of September 11, 2001 that occasioned in the destruction of the World Trade Center in New York City and significant damage to the Pentagon. These terrorist events changed the operational system and the aviation security policies at all commercial airports *(Mead 2002, 2003a)*. Additionally, due to the United States is geographical nearness to Canada so, the United States government has several reasons for trying to guarantee the security in all the Canadian airports in order to avoid any other terrorist attack *(Lyon, 2006).* In 1996, the Commission on Aviation Safety and Security, headed by Vice-President Albert Gore, recommended that the aviation industry improve security by using existing explosive detection technologies, automated passenger prescreening, and passenger baggage matching. Unfortunately, concerns over the cost of security procedures have resulted in the airline industry's uncertainly to implement many of these recommendations.

On November 19, 2001, the Aviation and Transportation Security Act (ATSA) was signed into legislation, which required additional aviation security procedures to be implemented. One of the requirements was the screening of 100% of checked baggage for explosives and weapons. In order to accomplish the objective, previously mentioned, it was necessary to purchase and install security devices at all commercial airports throughout the United States. The government started deployment of more than 6,000 baggage screening security devices around the entire country *(Mead, 2003)* .The deadline was later extended to the end of 2004, and in 2011 all the baggage has been screened for explosives and weapons with devices at every commercial airport in the United States *(Butler et al. 2002).*

The ATSA established the Transportation Security Administration (TSA), currently a part of the Department of Homeland Security (DHS), to offer a government division dedicated to the security of the transportation system throughout the United States *(Mclay et al. 2003).* Due to the complexity of the airspace structure and the numerous threats, it is a challenge to determine the type and number of security devices to deploy in each airport around the United States in order to have the greatest impact on security.

The checked baggage screening model (CBS) is an operational cost approach designed for use by the Federal Aviation Administration (FAA) personnel to forecast expected costs associated with implementing various baggage screening strategies at airports. A baggage screening strategy involves several levels of security screening devices, up to 10 levels, that a checked bag may pass through and presented in Figure 1. These devices can either clear a bag or alarm a bag and one security device is permitted per level. An alarmed bag would pass to the next screening device while a bag that has been cleared, would be loaded on the aircraft. The CBS model can be used to generate and evaluate different baggage screening strategies in any airport. In addition, the total cost of a project strategy can be generated based on different false clear and false alarm rates, the throughput of each device, purchase cost, operating cost and cost associated with device errors *(Jacobson et al. 2004).*

Figure 1: Baggage Screening Structure

There are few previous journal articles that have been using different approaches to solve the deployment of airport baggage screening security devices problem such as *(Mclay et al. 2005; Mclay et al. 2007)* used integer programming models for the deployment of baggage screening devices and also, *(Jacobson et al. 2004)* used integer programming for modeling and analyzing multiple station baggage screening security system performance. Moreover, *(Jacobson et al. 2004)* used simulated annealing algorithm for designing optimal aviation baggage screening strategies.

1.1 Thesis objective

It is very important to explain the problem that is being solved in this thesis. A baggage screening problem consisted in generated the best security strategy and only one security device is permitted for each level and multiple copies of the same type of device are allowable at each level. Since there are ten levels of devices, there are a lot of possible combinations of screening strategies. An effective security baggage system, the costs associated with the purchasing, and operation of these devices, should also include the costs of false alarms and false clears as performance measures to be included in the decision making process to determine the best selection of technology and optimal number of screening devices that minimizes the expected total annual cost.

In addition, the objective of this thesis is to demonstrate how Monkey Search algorithm can be used to obtain optimal deployment of baggage screening security devices. The present thesis is divided into 5 chapters. Chapter 2 presents a literature review about the most common methods used to solve optimization problems.

Chapter 3 is the problem statement of baggage screening deployment and model framework which contains equations to solve the deployment of baggage screening security devices. Chapter 4 describes the monkey search algorithm approach for the problem described in Chapter 2. In addition, Chapter 4 shows the computational results, sensitivity analysis, and provides an illustrative example for the monkey search algorithm for the deployment of baggage screening security devices problem. Chapter 5 presents concluding comments and directions for future research.

.

CHAPTER 2: META-HEURISTIC OPTIMIZATION METHODS

For this research several meta-heuristic methods have been studied to compare their advantages and disadvantages and evaluate which method is adequate to solve the baggage screening problem. Some of these methods are Bees algorithm, Genetic algorithm, Firefly algorithm, Viral System, Ant Colony optimization, Cuckoo Search, and Particle Swarm.

2.1. Ant colony optimization

The ant colony optimization is one of the most common meta-heuristic search methods based by foraging behavior of real ants and proposed by (*Dorigo et al. 1992; Dorigo et al. 1996*). This behavior allows ants to find shortest paths between food sources and their nest.

ACO system was first applied to traveling sales problems to find good solutions, for example (*Dorigo and Gambardella, 1997),* they used ACO as indirect system of communication mediated by a pheromone that ants deposit on the boundaries of the traveling sales problem graph while building solutions. Afterwards, ACO was used in other optimization problems such as, project scheduling in job-shops, flow-shops and open-shop problems (*Merkle et al. 2002*), for complexity of the optimal data aggregation in wireless sensor network (*Misra and Mandal, 2006*), and for solving facility layout problems with budget constraints (*Baykasoglu et al. 2004*).

The behavior of these artificial ants is inspired from real ants. Originally, ants search the areas surrounding their nest in random manner, and soon an ant finds a source of food and transports some of this food to the nest. During the return trip, the ant deposits a pheromone trail on the ground. The quantity of pheromone deposited, which might depend on the quantity and quality of the food, will guide other ants to the food source. There is an indirect communication between the ants via the pheromone trials permits them to find shortest paths between their nest and food sources.

In addition, the ants have some extra features that real ants do not have. Using the ACO in different cases, the pheromone is updated only after having constructed a whole path, and the amount of pheromone deposited represents the quality of the completed paths. Moreover, they have memory to save all their previous actions and could use that information to improve the quality of calculated paths. Figure 2 shows how the ants choose the best path considering the pheromone trail which is the main concept for Ant Colony Optimization.

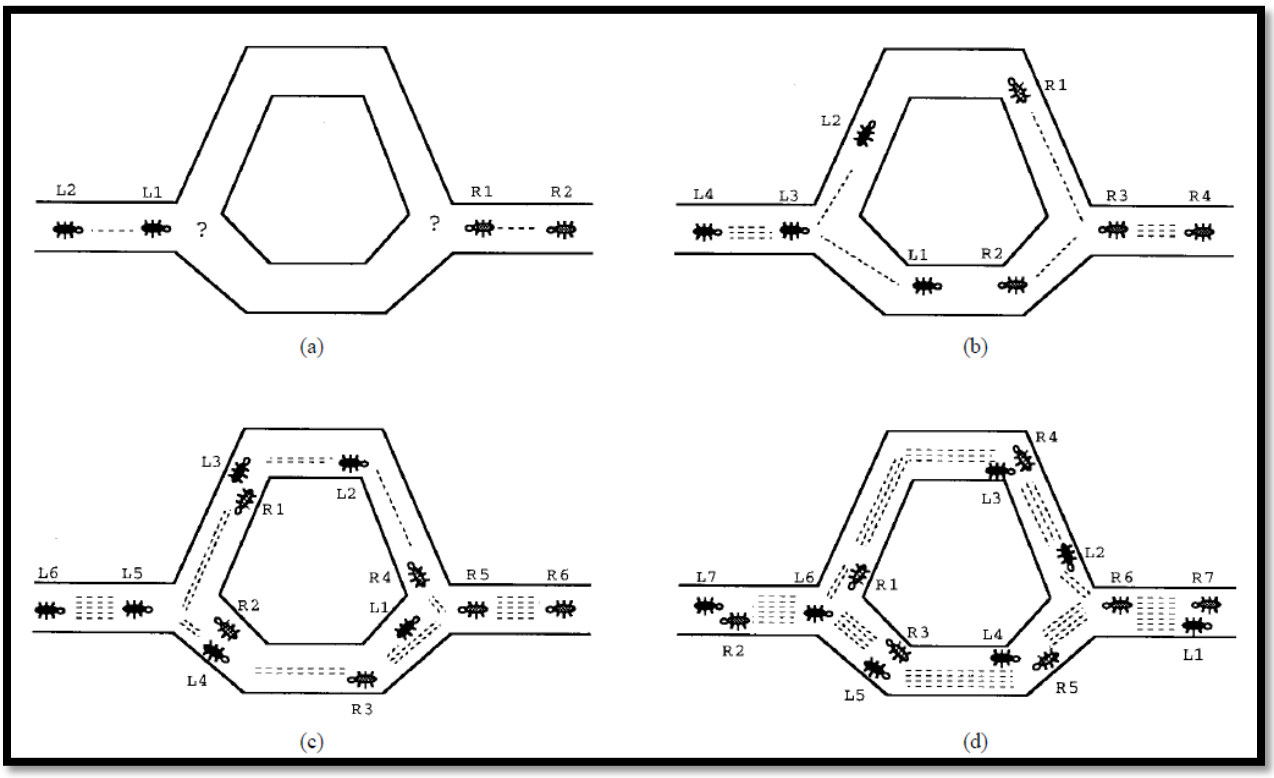

Figure 2: Ant Colony Optimization Concept

A) The ants are walking on a ground with different paths until they arrive at a decision point of which path is the best one.

B) Some ants randomly choose the upper path, while others choose the lower path.

C) All the ants are walking at a continuous speed, the ants which chose the lower path or shorter path, are going to reach the other corner faster than the ants which chose the upper path or longer path.

D) The number of dashed lines is roughly proportional to the amount of pheromone deposited by ants in all the different paths.

2.2. Bees Algorithm

Bees algorithm is based by the food foraging behavior of honey bees and could be considered as an intelligent optimization tool and was proposed by (*karaboga, 2005*). Their behaviors such as foraging, mating, and nest site location have been used by researches to solve many difficult combinatorial optimization problems. The honey bee colony behavior allows honey bees to explore the environment in search of flower patches (food sources) and then indicate the food source to the other bees of the colony when they return to the hive. It is a kind of swarm optimization algorithm that mimics nature's methods to drive the search towards the optimal solution.

BA has been used in different types of problems to find the nearest result to the optimal solution such as in scheduling to assign jobs for a single machine (*Pham et al. 2008*). In addition, it was used to find the optimal power flow to minimize the total production cost (*Kwannetr et al. 2010*), and loss minimization in power transmission network (*Othman et al. 2010*). Moreover, BA was applied in order to determine the forecast world carbon dioxide emissions and primary energy demand equations based on socioeconomic indicators (*Behrang et al. 2011*).

The proposed Bees algorithm starts with randomly sampled in the solution space, looking for areas of high performance. These areas are selected for further local search, until either a satisfactory solution is found or a predefined number of iterations have passed. During the

search, the bees balance random explorative and local exploitative searches, using their foraging behavior mechanism to find the best solution. Figure 3 shows how the bees find the closest solution toward optimal solution considering food foraging behavior of honey bees which is the main concept for Bees Algorithm.

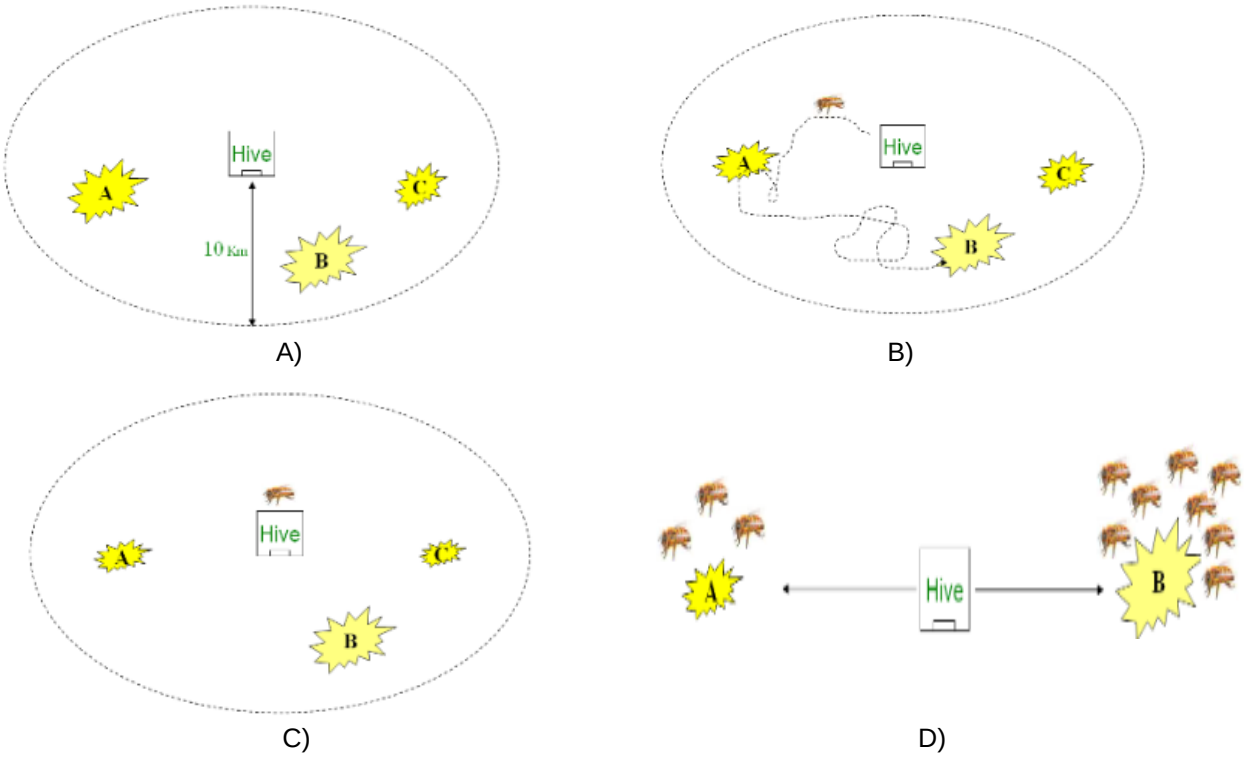

Figure 3: Bees Algorithm Optimization Concept

A) During the harvesting season, a bee colony employs part of its population to scout the fields surrounding the hive. A colony of honey bees can extend itself over long distances in multiple directions (more than 10 km).

B) Scout bees search randomly from one patch to another. In particular, scout bees look for flower patches where nectar is abundant, easy to extract, and rich in sugar content.

C) The bees who return to the hive, evaluate the different patches depending on certain quality threshold (measured as a combination of some elements, such as sugar content)

and they deposit their nectar or pollen on the "dance floor" to perform a "waggle dance" and communicate three basic pieces of information regarding the flower patch: the direction where it is located, its distance from the hive, and its quality rating.

D) Follower bees go after the dancer bee to the patch to gather food efficiently and quickly. More bees visit flower patches with plentiful amounts of nectar or pollen. Thus, according to the fitness, patches can be visited by more bees or may be abandoned.

Some parameters of the Bees algorithm are the number of scout bees *(ns*), number of elite sites (*ne*), number of best sites (*nb*), recruited bees for elite sites (*nre*), recruited bees for remaining best sites *(nrb*) and the initial size of neighborhood, all these parameters are necessaries to start looking closest solutions toward optimal solution.

2.3. Genetic Algorithms

The basic principles of Genetic Algorithm were first proposed by *(Holland, 1992).* Genetic algorithm was inspired by the mechanism of natural selection where stronger individuals are likely the winners in a competing environment. Genetic algorithms are stochastic optimization tools that work on Charles Darwin models of population biology *(Dobzhansky, 1959).* and are capable of solving for near optimal solution for multivariable functions without the usual mathematical requirements of strict continuity.

GA has been successfully applied in different areas to find the near optimal solution, such as in traveling sales problems to find an optimal route for visiting (n) cities and returning to point of origin *(Chatterjec et al. 1995).* In addition, it was used for single machine for scheduling different jobs and optimization *(Bean, 1994)* and for fuzzy logic with engineering applications *(Ross, 1995).* Moreover, GA was applied for optimizing the performance of a laser system for solid-state or gaseous *(Carroll, 1996).*

GA has several advantages when it is applied in any type of problem such as, optimizes with continuous or discrete parameters, simultaneously searches from a wide sampling of the cost surface. In addition, it does not require derivate information and may encode the parameters so, that the optimization is done with encoded parameters. Furthermore, it provides a list of optimum parameters so, not just a single solution and is well suited for parallel computers. Finally, it works with numerically generated and experimental data.

Figure 4 shows how the basic structure of the Genetic Algorithm is to find the near optimal solution for multivariable functions.

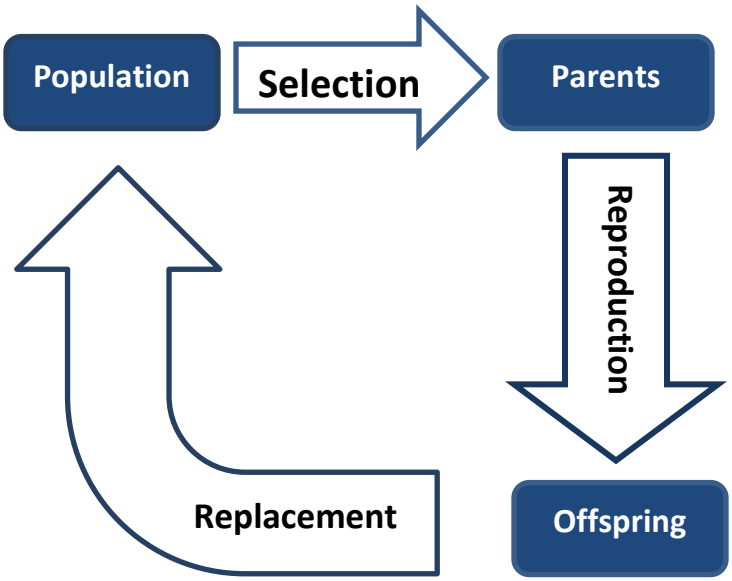

Figure 4: Genetic Algorithm Structure

The Generic algorithm is divided in four main stages which are: initialization, evaluation, selection, and reproduction. The first phase of the GA, a first population of individuals is generated normally randomly in which each individual is considered as possible solution. The next step is the selection which consists in the variations among individuals in the population result in some individuals being more fit than others and these differences are used to bias the selection of a new set of candidate solutions. During selection, a new population is created by

making copies of more successful individuals and deleting the less successful ones. The last stage is the reproduction in which ensures mixing and recombination among genes in their offspring. The algorithm continues until a stopping criterion is met. The next section explains the several strategies used in initialization, selection and crossover.

2.3.1 Encoded Strategies.

The encoded strategy is very important for problem solution to select proper encoding. Encoding represents transformation of solved problem to N-dimensional space of real numbers. The first stage in a genetic algorithm is the initialization of the first population in which there are several possible solutions. These possible solutions are called chromosomes which represent a legal solution to the problem and composed of a string of genes. Some of the most common encoded strategies are presented below *(Cheng et al. 2000)*.

Value Encoding: it can be used in problems where values such as real number are used. In addition, it is often necessary to develop some new types of crossovers and mutations specific for the problem. The values can be anything connected to the problem such as real numbers, characters or objects. An example of this type of encoding is presented in Figure 5.

Figure 5: Value Encoding

Permutation Encoding: it can be used in ordering problems, such as traveling salesman problems or task ordering problems. In permutation encoding every chromosome is a string of numbers that represent a position in a sequence. For some problems, crossover and mutation

corrections must be made to leave the chromosome consistent. This encoding strategy is presented in Figure 6.

Figure 6: Permutation Encoding

Binary Encoding: it is the most common to represent information contained and for genetic algorithms, it was first used because of its relative easiness. Sometimes corrections must be made after crossover and mutation. In binary encoding, every chromosome is a string of bits: 0 or 1. An example of the binary encoding is presented in Figure 7.

Figure 7: Binary Encoding

2.3.2 Selection Strategies.

Selecting the chromosomes that will either survive into the next generation, recombine to create offspring, or be rejected is one of the most important functions of the GA. Starved of a good selection strategy, the GA may meander away from its target problem and create random chromosomes that have no bearing on the problem. Certain of the most common selection strategies used in GA are: *(Nikolay et al. 2006).*

Rank selection: takes each chromosome, according to their arrangement in the population in increasing or decreasing order of fitness. The best chromosome is assigned rank one, and the worst is given rank equal to the population size.

Tournament selection: takes a certain number of chromosomes, normally two or three, and only chooses those chromosomes with the best fitness.

Roulette wheel selection: works like a roulette wheel where all chromosomes in the population are placed, every individual has its place big accordingly to its fitness function and parents are selected according to their fitness. The better the chromosomes are, the more chances they have to be selected.

2.3.3 Crossover strategies.

Crossover is a genetic operator that combines two chromosomes to produce a new chromosome. The main idea about crossover is that the new chromosome may be better than both of the parents, if it takes the best characteristics from each of the parents. The most common crossover strategies are presented below.

Single point crossover: is a crossover that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring. An example of the single point crossover is presented in Figure 8.

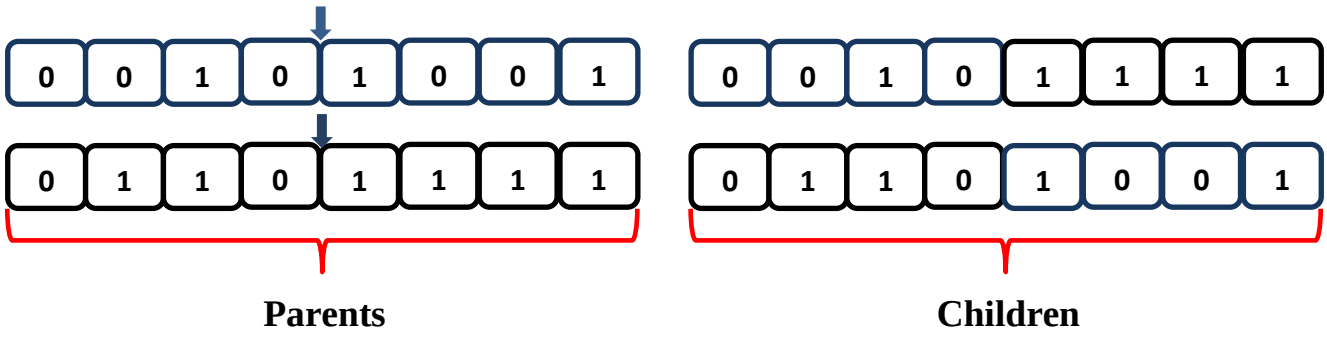

Figure 8: Single Point Crossover

Double point crossover: is a crossover that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring. An example of the double point crossover is presented in Figure 9.

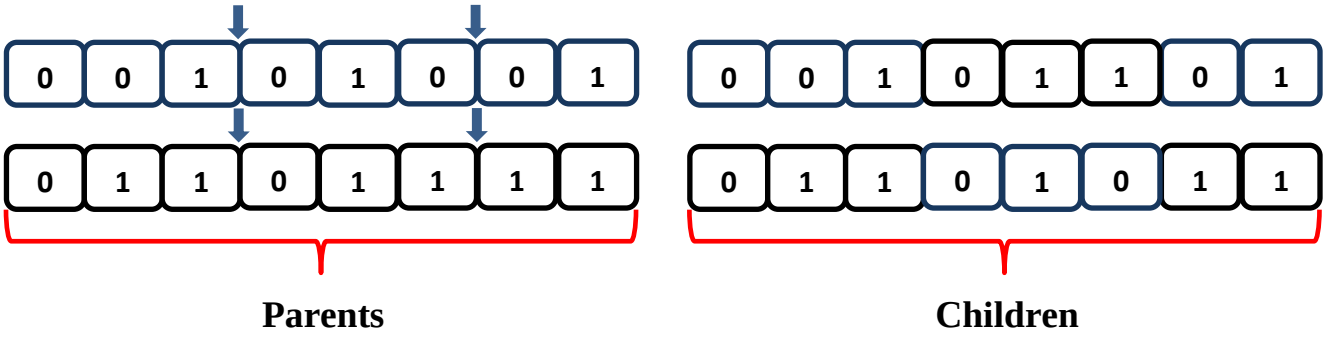

Figure 9: Double Point Crossover

Uniform crossover: is a crossover that chooses which parent will donate for each of the gene values in the offspring chromosomes using probability known as the mixing ratio. This allows the parent chromosomes to be mixed at the gene level rather than the segment level. An example of the uniform crossover is presented in Figure 10.

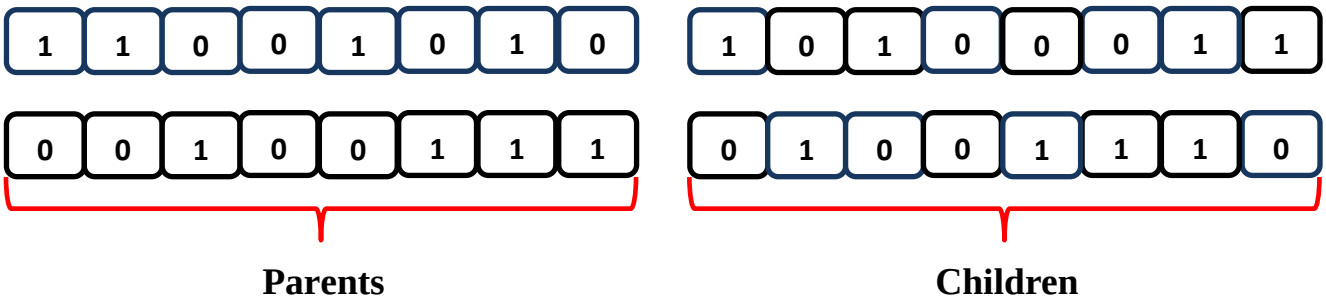

Figure 10: Uniform Crossover

2.4. Particle Swarm Optimization.

The particle swarm optimization technique was developed based on the social behavior of gathering birds and schooling fish when searching for food and proposed by *(Kennedy et al. 1995).* Some distinguish characteristic of PSO are high performance to find solutions and its flexibility to solve any type of problem. The PSO method is simple in concept and easily implemented with few coding lines to be programmed.

The PSO methodology has been applied in different areas such as power dispatch with pumped hydro *(Chen, 2009),* and electromagnetics *(Robinson et al. 2004)*. In previous research, it was used to solve the reactive power and voltage control considering voltage stability *(Yoshida et al. 1999)* and nonconvex economic dispatch problems *(Selvakumar et al. 2007).*

PSO approach consisted in using a number of agents as particles that establish a swarm moving around in search space, looking for the best solution. Every particle is preserved as a point in a N-dimensional space which adjusts its "flying" regarding to its own flying experience as well as the flying experience of other particles. Each particle retains track of its coordinates in the solution space which are connected with the best solution which is calculated by fitness and this value is called personal best (pbest). Another best value that is tracked by the PSO is the best value acquired so far by any particle in the community of that particle and this value is called global best (gbest). Finally, the simple concept of PSO falls in accelerating each particle toward its *pbest* and the *gbest* positions, with a random weighted acceleration at each time step. The main concept of modification of a searching point by PSO is presented in Figure 11.

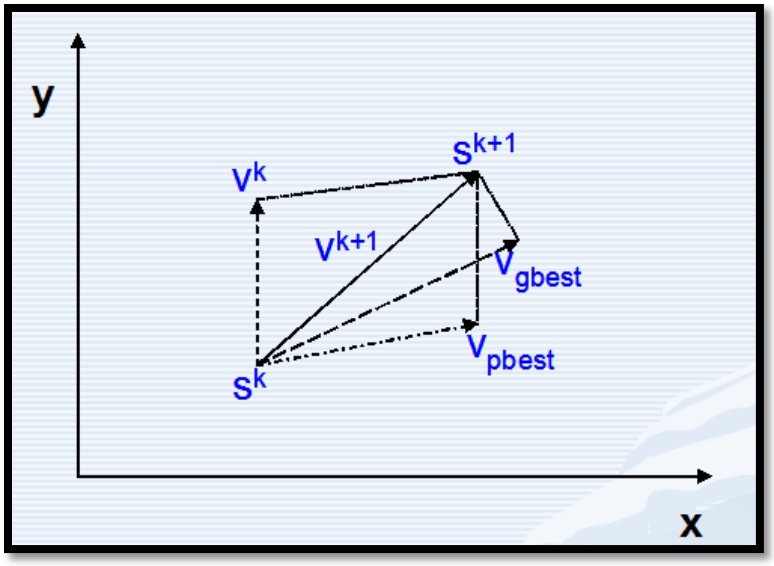

Figure 11: Concept of modification of a searching point

$\mathbf{s}^{k}$: current searching point.

$\mathbf{s}^{k+1}$: modified searching point.

$\mathbf{v}^{k}$: current velocity.

$\mathbf{v}^{k+1}$: modified velocity.

$\mathbf{v}_{pbest}$: velocity based on pbest.

$\mathbf{v}_{gbest}$: velocity based on gbest.

PSO concept has several advantages such as easy in its model and coding implementation as well as less sensitive to the nature of the objective function. In addition, less sensitive to parameters and less dependent on initial points. Additionally, PSO can produce high quality solutions with shorter calculation time. Some drawbacks of PSO are such as lacking rather of a solid mathematical foundation for analysis. Moreover, it still having some difficulties of dependency on initial conditions, parameter values, and finding the optimal design parameters

for stochastic characteristics. Finally, PSO is a strong tool for function optimization which is very easy to implement and offer a robust environment for different types of difficult problems.

2.5. Firefly Algorithm.

The firefly algorithm was developed based of simulating the social behavior of fireflies and proposed by *(Yang, 2009; 2010).* Those fireflies generated luminescent flashes as a signal system to communicate with other fireflies. The objective function in the firefly algorithm is related with the flashing light intensity which helps the swarm of fireflies to move toward brighter and more attractive locations in order to get the better solution nearest to the optimal solution.

The FA technique has been used to solve different problems such as synchronization in small overlay networks *(Bojic et al. 2010)* and illuminating future network on chip with nanophotonics *(Kumar et al. 2009).* In previous works done, the FA algorithm was applied for continuous constrained optimization task *(Laukasik et al. 2010)* and multi swarm and learning automata in dynamic environments *(Abshouri et al. 2011).* Finally, it was used in scheduling problems to solve a local search for makespan minimization in permutation of flow shop *(Ramezanian, 2010).*

The FA algorithm has three specific idealized rules which are based of the social behavior of fireflies *(Yang, 2009; 2010).* The first rule is that all the fireflies are unisex, and they will move towards more attractive and brighter ones regardless their sex. The second rule is that the grade of attractiveness of a firefly is proportionate to its brightness which decreases as the distance from the other firefly increases due that the air engrosses light. If there is not a brighter firefly than a previous one, it will start moving randomly. The last rule is that the brightness of a firefly is determined by the value of the objective function.

- **Attractiveness.**

In order to calculate the attractiveness function of a firefly is the following decreasing function presented in Equation 1 *(Yang, 2009; 2010).*

$$\beta(r) = \beta_0 \exp(-\gamma r^m), with.....m \geq 1 \quad (1)$$

Where: *r* is the distance between two fireflies, β_0 is the initial attractiveness and γ is an absorption coefficient which controls the decrease of the light intensity.

- **Distance.**

To determine the distance between two fireflies *i* and *j,* at location x*i* and x*j,* can be defined as a Euclidean distance formula presented in Equation 2 *(Yang, 2009; 2010).*

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2)$$

Where: *r* is the distance between two fireflies, x_i, y_i, x_j, y_j are the Cartesian coordinates.

- **Movement.**

To estimate the movement of a firefly *i* which is attracted by a more attractive light intensity, firefly *j,* can be determined by the following Equation 3 *(Yang, 2009; 2010).*

$$x_i = x_i + \beta_0 * \exp(-\gamma r_{ij}^2) * (x_i - x_j) + \alpha * (rand - \frac{1}{2}) \quad (3)$$

Where: x_i is the current position of the firefly, β_0 is the initial attractiveness and γ is an absorption coefficient which controls the decrease of the light intensity. The coefficient α is a randomization parameter determined by the problem of interest.

2.6. Viral System.

The viral system was proposed by *(Cortes et al. 2007)* and based in a new naturally approach simulating the natural biological process starting when organism has to provide a reaction to an external infection. A natural immune system defends the organism from hazardous external agents such as viruses. The antibodies try to keep the organism safe from any virus or bacteria. Viruses may be defined as cellular organisms whose genomes consist of nucleic acid, and which are obliged to replicate inside host cells in order to assemble into particles which serve to protect the genome and to transfer it to other cells *(Rybicki, 2007).*

There are few journals papers in which the VS concept has been applied such as solving constraint-satisfaction problems by a genetic algorithm adopting viral infection *(Kanoh et al. 1997)* and optimization of wind turbine placement using the viral based optimization algorithm *(Espiritu et al. 2011).* In previous research done, it was used to solve a traveling sales problem *(Kubota et al. 1996)* and in job scheduling problems *(Cortes et al. 2010).*

2.6.1 Viral system description

Every viral system is defined by three components: a group of viruses, an organism and the interaction among them *(Cortes et al. 2007).* All the viruses are defined in four components such as state, input, output and process.

- **State:** defines the cell infected by the virus and the evolution of the residence time of the virus taking in consideration the number of nucleus capsids replicated.
- **Input:** recognizes the information that the virus would be collected from the organism.
- **Output:** detects the actions that the virus can take.
- **Process:** represents the independent behavior of the virus.

The organism component in the viral system is defined by two components:

- **State:** consists of the clinical status and the lowest healthy of cells.
- **Process:** signifies the independent behavior of the organism that tries to protect itself from the infection risk.

The interaction component in the viral system is conditioned by the input and output actions that lead to a process of each virus and as a result of the organism reaction. The interaction is the combination of both actions. All the viruses have different capabilities regarding to infecting cells, including the mechanisms of replication and strategies to weaken the immune response of the host. Even though, the replication mechanism of viruses depends on the type of virus. The life cycle of the viruses can be developed in one step which is the lytic replication and also, it can include more than one step which is lysogenic replication. Those replications are presented in Figures 12 and 13 .

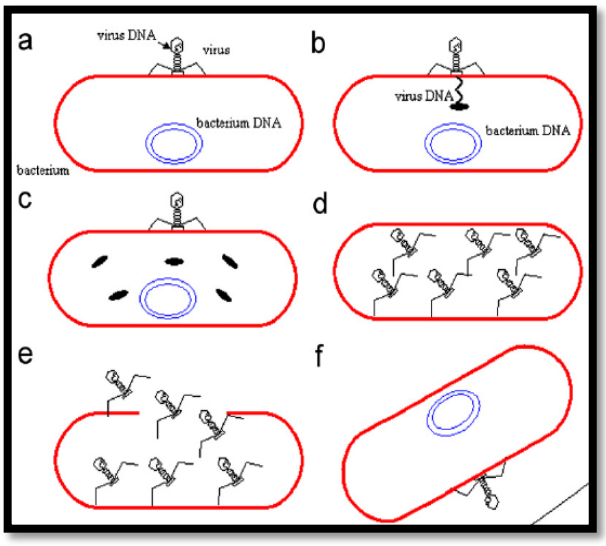

Figure 12: Lytic replication

- **Lytic Replication**

1. The viruses stick to the edge of the cell and then the viruses achieve to penetrate the border being injected inside its DNA, Fig. 12(a) and (b).

2. The infected cell stops making its proteins and starts to produce the phage proteins. Then, it starts to reproduce copies of the virus nucleus-capsids, Fig. 12(c) and (d).

3. Afterward duplicating a number of nucleus-capsids, the cell border is broken, and the new viruses are released which could infect other near cells, Fig. 12(e) and (f).

- **Lysogenic replication**

1. The virus infects the host cell which is presented in its genome, Fig. 13(a) and (b).

2. The virus stays concealed inside the cell during a time until it is triggered by different causes, Fig. 13(c).

3. The duplication of cells transformed with proteins from the virus, starts, Fig. 13(d) and (e).

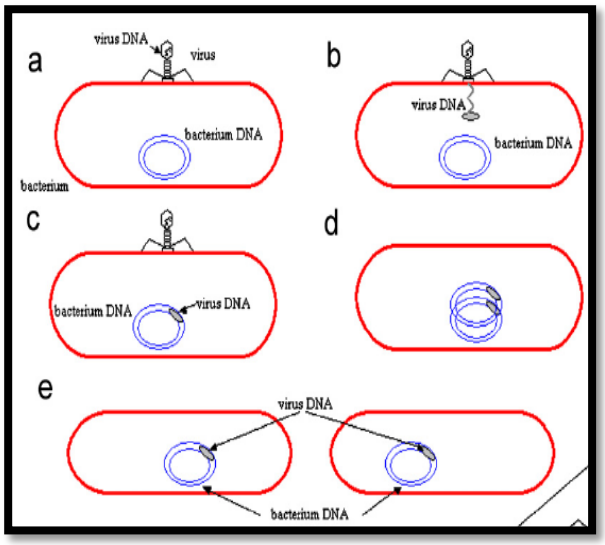

Figure 13: Lysogenic replication

2.7. Cuckoo search

Cuckoo search was developed based by the obligate brood parasitism of some cuckoo species by placing their eggs in the nests of host birds and proposed by *(Yang and Deb, 2009).* The host takes care of the eggs presuming that the eggs are in its own attention. If the host discovers that an egg is not in its own attention, it may both destroy the egg or the nest and then build a new nest at different location.

The CS approach has been used in different types of problems such as solving structural optimization problems *(Yang et al. 2011)* and it was implemented for energy efficient computation of data fusion in wireless sensor networks *(Dhivya et al. 2011).* In addition, it was applied to design optimization for reliable embedded system *(Kumar and Chakarverty, 2011)* and for solving unconstrained optimization problems *(Tuba, et al. 2010).*

In order to explain how the cuckoo analogy was used for developing the optimization algorithm the following cuckoo search flow chart presented in Figure 14 explains step by step its process.

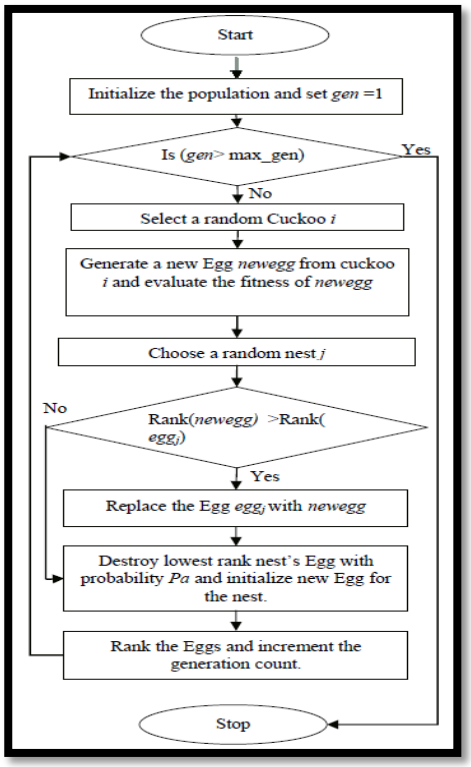

Figure 14: Cuckoo search flow chart

1. A generation is represented by a set of host nests and each nest carries an egg which is a solution.

2. Each generation carries the best eggs as best solutions to next generation.

3. The quality of eggs is improved by generating a new egg which is randomly selected from the nest and that new egg is molded by a random move on the selected egg.

4. If the new egg is found as superior egg than the current egg in another randomly chosen nest then the old egg is substituted with the new one and the number of nests remains fixed in each generation.

5. A certain probability *Pa*, a new egg randomly generated replaces the lowest ranked of the nest's egg and this process helps in avoiding local minima. In this situation, the host bird could either throw the egg away or abandon the nest and construct a totally new nest.

Using cuckoo search for maximization problem, the fitness of a solution can be proportional to the value of its objective function. There are other methods of fitness which can be defined in a similar way that the fitness function is defined in genetic algorithms and other meta-heuristic algorithms. Finally, CS algorithm holds good solutions in the population while trying to locate the best one and also, CS is significantly less sensitive to variation in the evolution parameters.

In conclusion, every different meta-heurist optimization method has some advantages and drawbacks such as the Genetic Algorithm has some advantages which are: it can solve every optimization problem which the chromosome encoding. In addition, it solves problems with multiple solutions and multi-dimensional, non-differential, non-continuous and even non-parametrical problems. Some disadvantages are: certain optimization problems cannot be solved by means of genetic algorithms. This occurs due to poorly known fitness functions which generate bad chromosome blocks and there is no absolute assurance that a genetic algorithm finds a global optimum solution. The Ant Colony optimization has some advantages such as: it can be used in dynamic applications, positive feedback leads to rapid discovery of good solutions, and distributed computation avoids premature converge. Some drawbacks are convergence is guaranteed, but time to convergence is uncertain and coding is not straightforward. Moreover, Particle Swarm and Cuckoo search algorithm is easy to preform and it has few parameters to adjust. Also, this algorithm is really good in global search and not in local search. Some drawbacks are slow convergence in refined search stage and weak local search ability. Finally, all the meta-heuristics methods have advantages and some drawbacks. Some methods are really good for global search and some for local search in order to solve the baggage screening problem the Monkey Search which is really good for local search was chose.

CHAPTER 3: PROBLEM STATEMENT.

3.1 Problem Description

On November 19, 2001, the Aviation and Transportation Security Act (ATSA) was signed into legislation, which needed additional aviation security procedures to be implemented. One of the requirements was the screening of 100% of checked baggage using screening devices for explosives and weapons. The new security system paradigms that optimally utilize and simultaneously coordinate several security technologies and processes needed to be developed to ensure the effectiveness of the security at airports throughout the nation. In order to accomplish the objective already previously mentioned, it was necessary to purchase and install security devices at all commercial airports through the United States. The government started deployment of more than 6,000 baggage screening security devices around the entire United States *(Mead, 2003).*

This research is based on concepts of previous work done by (*Candalino, 2001)* and also in the checked baggage screening model (CBS) developed by the Federal Aviation Administration *(Berrick, 2005).* The CBS model generates the annual purchase and operating cost of a screening strategy based in a selection of screening devices. Currently, there are 39 different screening devices in the CBS model that will be used to generate the best security strategy and only one security device is permitted for each level and multiple copies of the same type of device are allowable at each level. Since there are ten levels of devices, there are a lot of possible combinations of screening strategies.

In addition, it is highly time consuming to enumerate and analyze all the possible combinations for obtaining an optimal solution so for that reason the monkey search algorithm (MS) will be used to find the best solution nearest the optimal solution *(Mucherino and Seref,*

2008). Table 1 below shows the 39 different screening devices and their costs, throughput for each device, false alarm rate and false clear rate.

In order to design an effective security baggage system the costs associated with the purchasing and operation of these devices should also include the costs of false alarms and false clears as performance measures to be included in the decision making process to determine the best selection of technology and optimal number of screening devices that minimizes the expected total cost model including both the direct cost and the indirect cost associated with system errors. Uniform screening can possess a high purchasing cost from applying new technologies to all bags as equal. Therefore, creating a set of security levels for baggage screening may be more cost effective. The different security levels would apply to select technologies and procedures to a subset for bags that have not been clear in the first screening. Finally, this is an NP-hard optimization problem which makes this problem more complicate to solve in order to find the a set of solutions.

Table 1: Different Devices Attributes

Device ID	Cost($)	Throughput	FA rate (%)	FC rate (%)
1	900,000.00	1200	40	7.50
2	330,000.00	600	25	9.50
3	250,000.00	100	20	8.50
4	850,000.00	100	15	7.50
5	850,000.00	254	30	6.50
6	3,000.00	1000	50	5.50
7	2,000.00	2	10	5.45
8	500.00	1000	50	5.35
9	30,000.00	10	0	8.25
10	5,000.00	1000	50	7.50
11	965,000.00	50	25	7.50
12	850,000.00	251	15	9.50
13	965,000.00	260	25	8.50
14	200,000.00	62	20	7.50
15	80,000.00	80	25	6.50
16	70,000.00	80	15	7.55
17	90,000.00	1000	50	8.35
18	560,000.00	24	20	5.25
19	80,000.00	600	40	7.50
20	800,000.00	50	15	5.00
21	50.00	1	20	9.50
22	8,300.00	180	20	8.50
23	16,600.00	360	30	7.50
24	1,500,000.00	12	0	6.50
25	965,000.00	120	15	7.55
26	850,000.00	100	15	9.45
27	330,000.00	20	18	4.35
28	450,000.00	100	25	8.25
29	2,000,000.00	62	25	5.30
30	80,000.00	80	12	9.50
31	57,000.00	57	10	9.50
32	57,000.00	80	25	8.50
33	70,000.00	80	15	7.50
34	45,000.00	66	15	9.65
35	200,000.00	1	0	5.50
36	600,000.00	80	20	6.00
37	60,000.00	1000	50	6.35
38	200.00	12	10	8.25
39	450,000.00	550	25	7.50

3.2 Baggage Screening Definitions

A baggage screening strategy involves several levels of security screening devices and up to ten levels in which a bag will be checked passing through them. These devices can either clear a bag or alarm a bag. An alarmed bag would pass to the next level screening device while a bag that has been cleared would be loaded on the airplane. For this research four different situations in baggage screening are being considered (*Candalino, 2001)*:

- **True Clear:** a bag that has cleared the screening and contains no threat.
- **True Alarm:** the device alarms a bag and legitimately contains a threat.
- **False Alarm:** the device alarms a bag and does not contain a threat.
- **False Clear:** a bag that has cleared the screening and legitimately contains a threat.

There are some costs associated with false alarms and false clears. For true clears the cost can be associated directly to the purchase of the devices. False alarms and true alarms share the costs that can be directed to the amount of extra time and resources that must be utilized to inspect the suspected baggage. For instance, the cost to utilize a special bomb squad or devices to examine the piece of baggage and evacuating a section or the entire airport. Furthermore, false clears have an extra and considerable higher cost in the case of fatal event.

Each baggage screening device has five different attributes in order to determine a specific baggage screening strategy.

- **Identification number:** helps as a unique tag for each baggage screening security device type.
- **False alarm rate** (r_{FA}): it is the probability that the device alarms a certain bag and that the bag does not contain a threat, and sends the bag to the next level for screening.

- **False clear rate** (r_{FC}): it is the probability that the device clears a certain bag and that the bag contains a threat.
- **Throughput rate** (Bags/hr): it is the number of bags that the device is able to screen per hour of operation.
- **Purchase Cost** ($): these are the different prices for all the different baggage screening security devices.

There are more necessary parameters that cooperate with the five attributes of the baggage screening security devices to determine the cost of a certain baggage screening strategy. All these additionally parameters and its values were taken from the following source *(Virta et al. 2001).*

- **Probability of a threat** (r_T) = 5 X 10^{-10}
- **Expected annual operating cost of a device** (C_O) = $125,000.
- **Cost of a system false alarm** (C_{FA}) = $30.
- **Cost of a system false clear** (C_{FC}) = $1, 400, 000, 000.

The cost function of the baggage screening deployment problem involved not just the purchase cost, there are more important costs that need to be taken into consideration to generate a baggage screening strategy. The cost function for this model has four main important costs:

- **Purchase cost:** it will be based on the number of screens that are required at each level.
- **False Alarm Cost:** it will be based on the false alarm rates, probabilities of threat, the expected number of bags screened in a year and the cost of a false alarm system.

- **False Clear Cost:** it will be based on the false clear rates, probabilities of threat, the expected number of bags screened in a year and the cost of a clear alarm system.
- **Operating Cost:** it will be based on the number of devices needed at each level to screen the expected number of bags per hour and the annual operating cost which are the expenses related to the operation of screening baggage devices. For example: maintenance cost, personnel cost, training cost.

3.3 Model Framework

There are ten different levels of baggage screening devices which help to make the baggage screening strategy better. At each level, there can be only one type of device and similar copies of the same type of device to screen the expected number of bags arriving at a specific level. In order to identify the ID of the security device at any level *(i)* is: *ID(Device(Level i)).* In the same way tha false alarm and false clear rates of a device at any level *(i)* are: *FA(Device(Level i))* and *FC(Device(Level i)).* At each security level *(i)*, there are three calculations which need to be performed (*Candalino, 2001)*:

1. Number of bags to be screened at level *(i)* is presented in Equation 4:

$$Bags(level(i)) = Peakbags * \left[(1 - PT) \prod_{j=1}^{i-1} (FA(Device(Level(j)))) + PT \prod_{j=1}^{i-1} (1 - FC(Device(Level(j)))) \right] \quad (4)$$

Where: *Peakbags* = is the maximum number of bags that will arrive at the first level of devices in a given hour, *PT*= probability of a threat, *FA*= false alarm rate, and *FC*= false clear rate.

An example of how the structures of expected bags are sent into the aircraft or to the next security level is presented in Figure 15.

		LEVELS		EXPECTED NUMBER OF BAGS CLEARED TO GO ON THE AIRCRAFT
PEAK BAGS	→	LEVEL 1	→	$(FC1*PT+(1-FA1)*(1-PT))$ $*PeakBags$
	$FA1*(1-PT)+(1-FC1)*PT)*$ $Peakbags$	↓		
		LEVEL 2	→	$(FC2*(1-FC1)*PT+(1-FA2)*$ $FA1*(1-PT))*PeakBags$
	$(FA2*FA1*(1-PT)+(1-FC2)$ $*(1-FC1)*PT)*Peakbags$	↓		
		LEVEL 3	→	$(FC3*(1-FC2)*(1-FC1)*PT+$ $(1-FA3)*FA2*FA1*(1-PT))*PeakB.$
	$(FA3*FA2*FA1*(1-PT)+(1-$ $FC3)*(1-FC2)*(1-FC1)*PT$ $*Peakbags$	↓		
		LEVEL n		

Figure 15: Structure of expected bags sent into the aircraft or to the next level.

2. Number of devices required for level *(i)* is presented in Equation 5:

$$Screens(level(i)) = Bags(Level(i))/Throughput(Device(Level(i))) \quad (5)$$

Where: *Bags*= number the bags for each level, and *Throughput* = number of bags that the device is able to screen per hour of operation.

3. Purchase cost of the level *(i)* is presented in Equation 6:

$$Level\cos t(level(i)) = Screens(level(i))*PurchaseCost(Device(Level(i)) \quad (6)$$

Where: *Screens*= number of security devices, and *PurchaseCost*= cost of each baggage screening security device.

The summation of this last calculation for all the levels determines the total cost for (n) number of levels and is presented in Equation 7:

Total Purchase Cost= $\sum_{i=1}^{n}\left[LevelCost(Level(i))\right]$ (7)

Where: *LevelCost*= cost for each security level.

A device is assumed to have ten years of operating life. Consequently the annualized purchase cost is the Total Purchase Cost is divided by 10. The average number of bags per hour is assumed to be 10 times less than the peak value. Therefore, the average number of bags screened in a year can be calculated:

- Average bags screened in a year is presented in Equation 8:

$$S = (PeakBags/10) * 24 * 365 \quad (8)$$

Where: *Peakbags* = is the maximum number of bags that will arrive at the first level of devices in a given hour.

The annual cost of False Alarms and False Clears are calculated:

- Expected Annual False Clear Cost is presented in Equation 9:

$$FCCost = CostFalseClear * S * (PT) * \quad (9)$$

$$\left[FC(Device(Level(1))) + \sum_{i=2}^{n} FC(Device(Level(i))) * \prod_{j=1}^{i=1} (1 - FC(Device(Level(j)))) \right]$$

Where: *CostFalseClear*= cost of a system false clear, *S*= average bags screened in a year, *PT*= probability of a threat, and *FC*= false clear rate.

- Expected Annual False Alarm Cost is presented in Equation 10:

$$FACost = CostFalseAlarm * S * (1 - PT) * \prod_{i=1}^{n} FA(Device(Level(i))) \quad (10)$$

Where: *CostFalseClear*= cost of a system false clear, *S*= average bags screened in a year, *PT*= probability of a threat, and *FA*= false alarm rate.

Finally, the total expected annual strategy cost will be the summation of the Total Purchase Cost over all levels divided by ten. Consequently, multiplication for the summation of all screen levels plus the expected annual Operating Cost, False Clear Cost, and False Alarm Cost is presented in Equation 11:

$$\text{Total Annual Strategy Cost} = TotalPurchaseCost/10 + CO * \sum_{i=1}^{n}(Screens(level(i))) + FCCost + FACost \tag{11}$$

Where: *TotalPurchaseCost*= total cost for all the security levels, *CO*= operating cost, *Screens*= number of security devices, *FCCost*= expected annual false clear cost, and *FACost*= expected annual false alarm cost.

The False Alarm and False Clear probabilities are determined for (n) number of levels and are presented in Equations 12, 13, 14 and 15:

$$P(FC(i)) = PT * FC(i) * \prod_{j=1}^{i-1}(1 - FC(j)) \tag{12}$$

$$P(TC(i)) = (1 - PT) * (1 - FA(i)) * \prod_{j=1}^{i-1}(FA(j)) \tag{13}$$

$$P(FA(i)) = (1 - PT) * \prod_{j=1}^{i}(FA(j)) \tag{14}$$

$$P(TA(i)) = PT * \prod_{j=1}^{i}(1 - FC(j)) \tag{15}$$

Where: *PT*= probability of a threat, *FA*= false alarm rate, and *FC*= false clear rate.

3.4 Model Assumptions

The baggage screening model has several assumptions in order to make the problem solvable in a practical way and time. For example, all the constants used in this model are estimates based on data from the Federal Aviation Administration such as: probability of a threat, expected annual operating cost of a device, cost of a system false alarm and cost of a system false clear. In the probability of a threat, the constant is not the same for all the periods of times for all airports. In addition, the cost of false alarm system and false clear system could vary depending on each scenario. Moreover, the region of the objective function that has possible optimal solutions can be considered as a jungle or forest and it is not restricted with a single tree. The annual operating cost is just an average; each device has different costs associated with its operation. To determine the number of bags at each level assumes that false alarm and false clear rates are independent of each other. To finish, the average number of bags screened in a year is an approximation so the annual bags incoming to all airports may not be modeled in this way.

CHAPTER 4: MONKEY SEARCH ALGORITHM APPROACH

The Monkey Search algorithm is based on the similarity of the behavior of a monkey looking for food by climbing up trees and proposed by (*Mucherino and Seref, 2007*). The general assumption is that the monkey explores the trees and learns which branches lead to better food resources. The found food on the trees is represented as possible solutions. They are arranged in the order of the final solution value from objective function and the branches of the tree of solutions are represented as perturbations between two neighboring feasible solutions. By saving the optimum solutions, along with some other better solutions, a crossover is performed to replace the inferior monkey's food resource with new and better locations.

There are few journals papers in which the Monkey Search algorithm concept has been applied such as supply chain network design under uncertainty *(Huang et al. 2008)* and research on fault diagnosis *(Zhang et al. 2009).* In previous research done, it was used to solve a molecular distance geometry problem *(Mucherino et al. 2009)* and in transmission network expansion planning *(Wang et al. 2010).*

4.1 Initialization

In the Monkey algorithm its position *i* is denoted as a vector X_i that will be used to find a solution of the optimization problem. In order to initially place a monkey, assuming that the region of the objective function that has possible optimal solutions can be considered as a jungle or forest and it is not restricted with a single tree, Figure 16 shows the jungle which has all the possible optimal solutions. Each monkey position is generated randomly inside the solutions region and will be assumed that every monkey position is feasible.

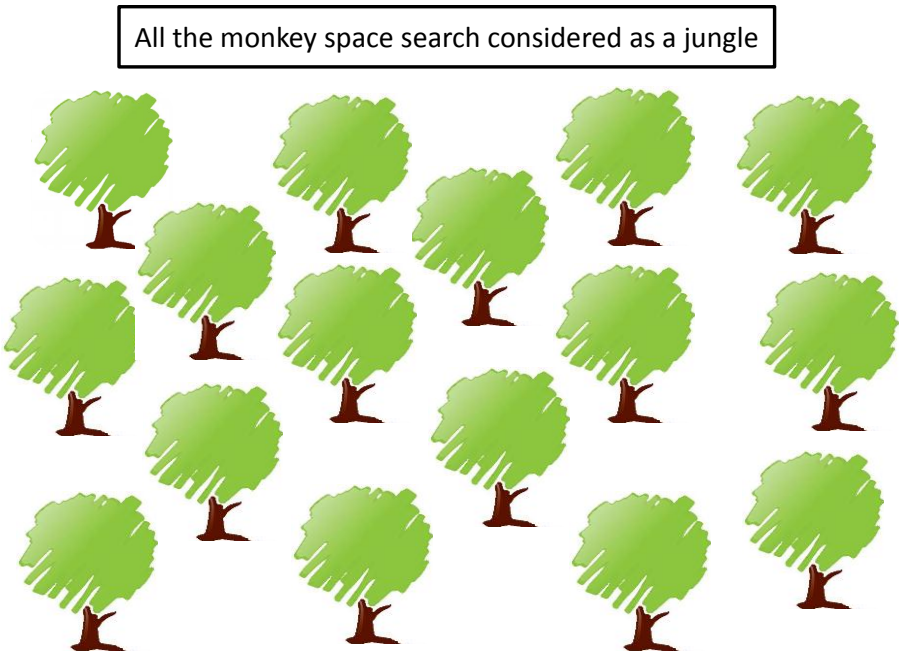

Figure 16: The objective function of the monkey search algorithm.

4.2 Climb Process

1. The climbing process begins in the moment a monkey starts climbing up to search for food. While climbing, the monkey will find two branches which will lead to a group of different possible solutions. The monkey will climb up both branches, represented as left branch and right branch, Figure 17 (a).

2. Every tree is considered to have a number of paths from the root that reach the whole height of the tree. Each time the monkey reaches the permitted height of the tree, one of the several paths is supposed to be revised, Figure 17 (b).

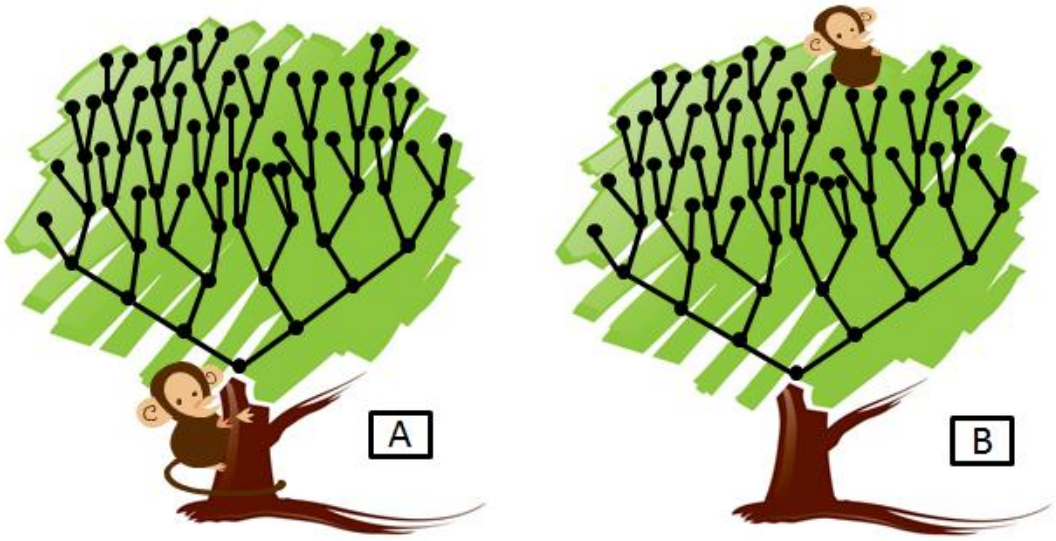

Figure 17: Climb process

3. When a better solution is found, the monkey starts climbing down to the main path that connects the solution to the root, and save this path as possible solution. If the new solution improves the objective function value of the last solution, it stores this new solution as the current best solution (X_{curr}). When the monkey is coming back, it will search for new braches which will be marked for be inspected, Figure 18 (c).

4. After the monkey reaches the bottom of tree, he will start climbing up the tree again for a (n_t) desired number of times to revise the previous marked branches randomly with more probability towards the branches with better optimization values. Depending of the performance of the branches with better values, the monkey tries to look in that specific area which the global solution could be found. Each time the monkey finds a solution with a better objective function value, it stops climbing and stores the best solution (X_{best}), Figure 18 (d) and (e).

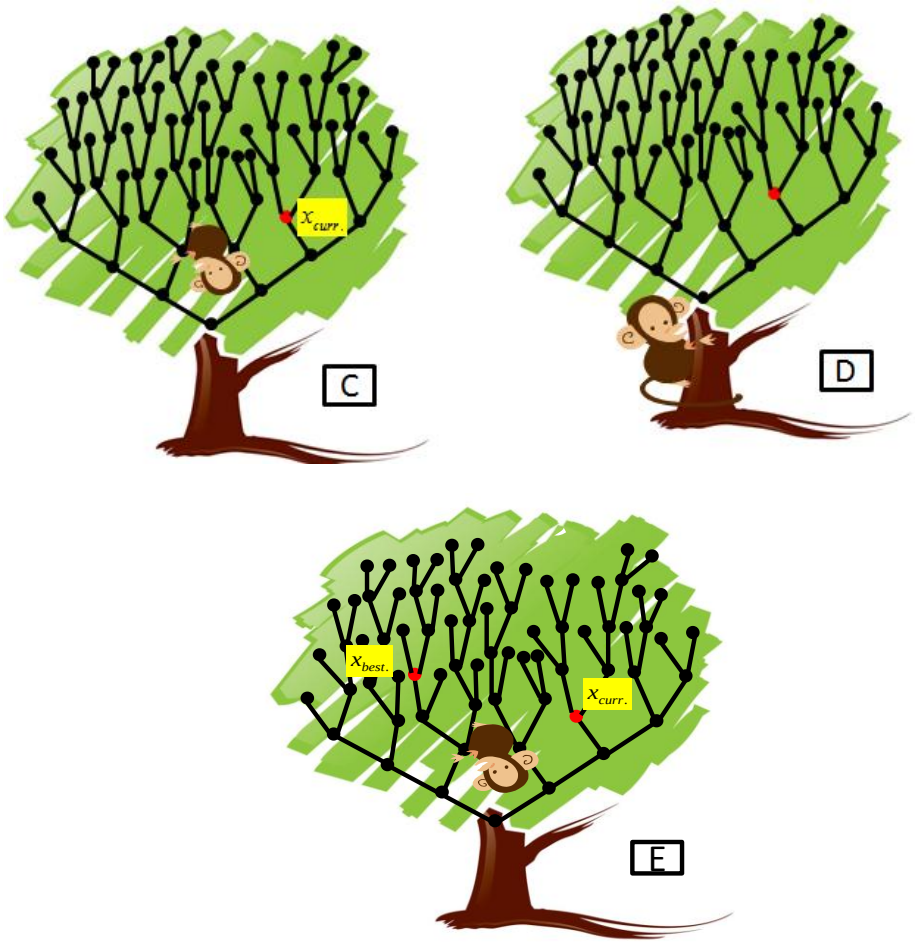

Figure 18: Climb process

4.3 Somersault Process

Once the monkey reaches the permitted height of the tree and search in all paths. The monkey then makes a somersault to a new tree and begins climbing. The monkey starts from the best solution, or a combination of better solutions, as shown in Figure 19. New feasible solutions are created starting from the best solution and using a randomized perturbation function. The immediate information available to the monkey at any point is the current solution X_{curr} and the best solution X_{best} on the current tree. These two solutions can be used in the perturbations for generating new solutions.

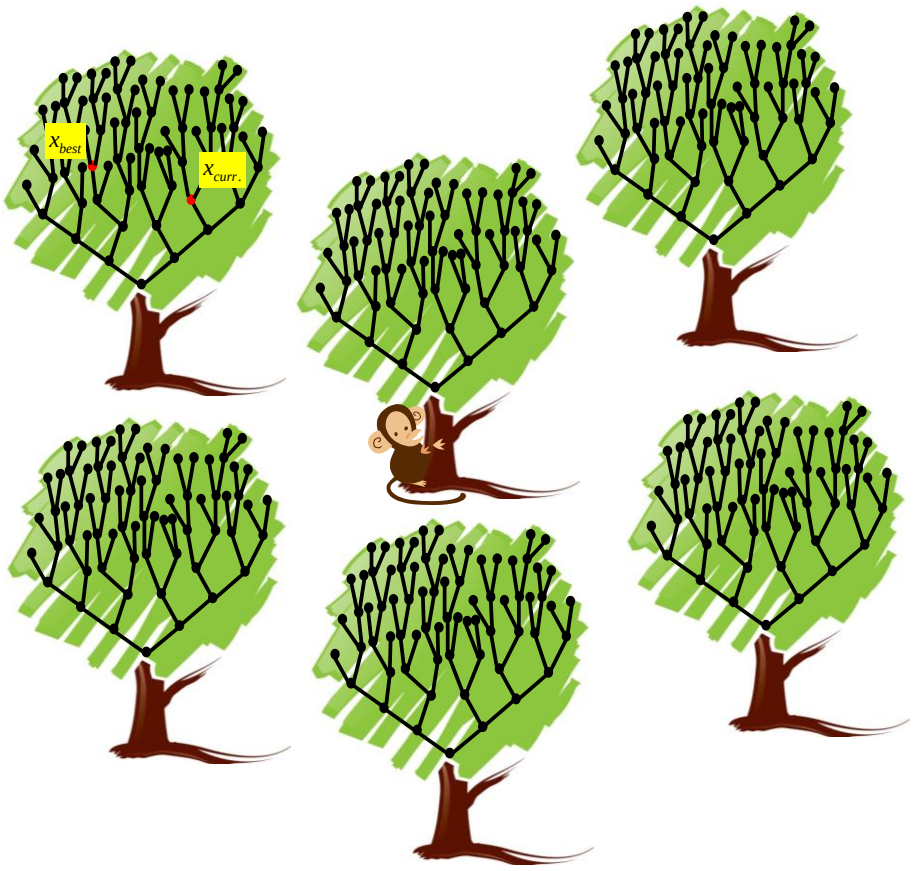

Figure 19: Somersault process

According to (*Mucherino and Seref, 2007*), there are several perturbations which can be applied on the Monkey Search algorithm such as:

- Simulated Annealing methods: random changes in X_{curr} .
- Genetic Algorithms: crossover operator (double point) applied for generating a child solution from the parents X_{curr} and X_{best}.
- Ant Colonization: the mean solution made from X_{curr} and X_{best} .

For this research, the double point crossover operator perturbation from Genetic Algorithm was used to find new feasible solutions. The double point in a crossover is that it randomly selects two crossover points within a chromosome, then interchanges the two parent

chromosomes between these points to produce two new offspring. An example of the double point crossover implemented in Monkey Search algorithm is presented in Figure 20.

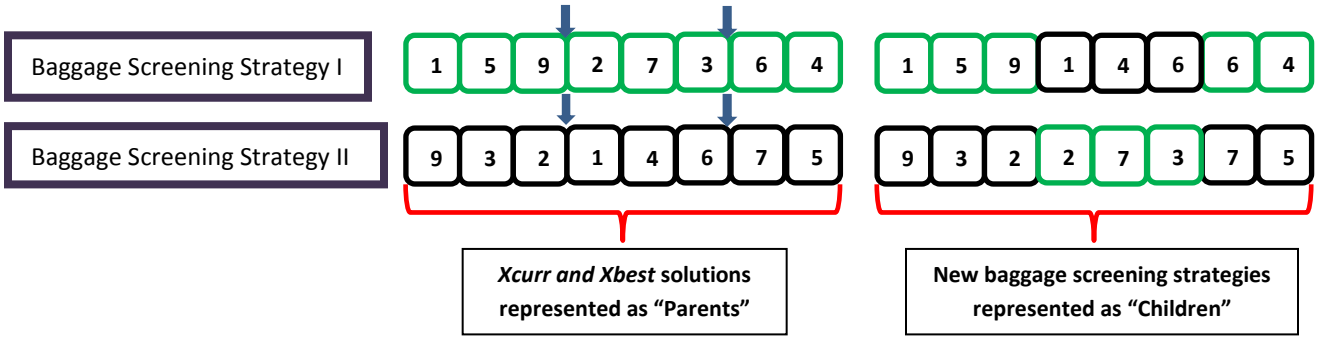

Figure 20: Double point crossover implemented in Monkey Search algorithm

Finally, the monkey stops when it finishes searching solutions in a certain number of trees or the desired number of iterations.

4.4 Monkey Search Parameters

The Monkey Search is based on the following parameters which influence the performance of this algorithm:

- The height of the trees (h_t)
- The number of times that the monkey reaches the top of the tree (n_t)
- The size of the memory (n_m)
- The starting number of random trees (n_w)
- The number of iterations (n_i)

In order to avoid local optima, the algorithm retains a number of best solutions updated by each consecutive tree revised in the memory of the monkey (n_m). He updates his memory whenever a new solution is better than the ones that are already in the memory and removing the

closer solution in memory that is worse than the new one. In order to force this algorithm to search in different spaces, the Monkey Search algorithm starts with a certain number of trees (n_w) and randomly generated solutions.

4.5 Monkey Search Algorithm Implemented in Baggage Screening Problem

In order to understand how the Monkey Search algorithm was implemented in the baggage screening problem, the following flow chart explains step by step. See Figure 21.

1. First of all, all the input data entered which are the device ID, cost, throughput, FA rate, and FC rate for each device.
2. The program starts calculating the baggage screening evaluation in order to determine the total annual strategy cost.
3. The initiation begins when each monkey position is generated randomly inside the solutions region and it will be assumed that the every monkey position is feasible.
4. The monkey starts climbing up the tree for (n_t) desired number of times until it searches in all the paths that each tree has. If the new solution improves the objective function value, it replaces the previous solution as the best new solution.
5. When all the multiple paths are explored, the monkey makes a somersault to another new tree and begins climbing, starting from the best solution, or a combination of better solutions.
6. Finally, the monkey stops when it finishes searching solutions in a certain number of trees or the desired number of iterations and the final best baggage screening strategy is obtained.

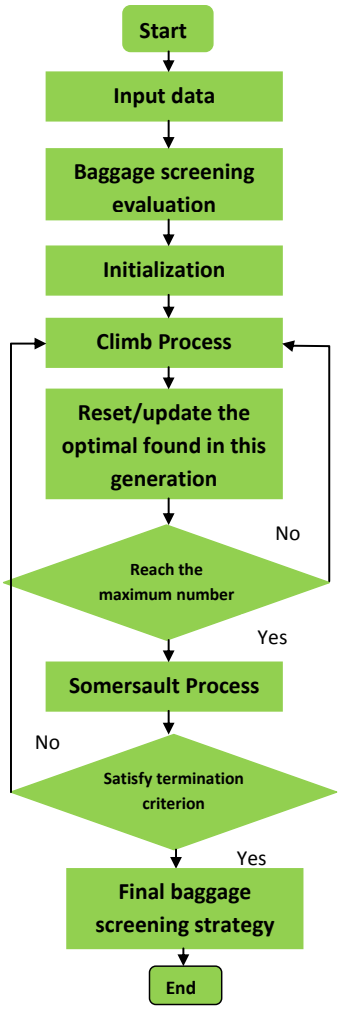

Figure 21: Flow Chart of Monkey Search and Baggage Screening Problem.

4.6 Baggage Screening Evaluation Example

To better demonstrate the process of this algorithm, a small example was performed. First, several parameters must be defined, including the number of screening levels as well as the devices available and their characteristics. In this example, there are four possible levels; the device characteristics are listed in Table 2. Five thousand bags will need to be screened in an hour. The Monkey Search algorithm started by selecting a screening strategy at random.

Table 2: Devices and characteristics

Device ID	Cost($)	Throughput	FA rate (%)	FC rate (%)
7	2,000.00	2	10	5.45
13	965,000.00	260	25	8.50
22	8,300.00	180	20	8.50
32	57,000.00	80	25	8.50

Random strategy generated:

Case 1: Baggage strategy of 2 levels

Level 1: Device 13

Level 2: Device 32

After a strategy is generated several calculations must be performed before the overall strategy cost can be found.

Calculations:

Bags(Level 1) = 5000

Screens(Level 1) = Bags(Level 1)/Throughput(Device 13) = 5000/260 = 19.2308 = **20**

Cost(Level 1) = Screens(Level 1)*Cost(Device 13) = 20*965,000.00 = **$1.93e+007**

Bags(Level 2) = Bags(Level 1)*((FA(Device 13)*(1-PT)+(1-FC(Device 13)*(PT))

= 5000*((.25)*(1-5 X 10^{-10})+(1-.085)*(5 X 10^{-10})) = **1250**

Screens(level 2) = Bags(Level 2)/Throughput(Device 32) = 1250/80 = 15.625 = **16**

Cost(Level 2) = Screens(level 2)* Cost(Device 32) = 16*57,000.00 = **$912,000.00**

FCCost = Cost of False Clear*S*PT* ((FC(Device 13))+((FC(Device 32)*(1-FC(Device 13)))

= 1, 400, 000, 000*4.38 X 10^{06}*5 X 10^{-10} *((.085)+((.085)*(1-.085))) = **$260,610.00**

FACost = Cost of False alarm*S*(1-PT)*((FA(Device 13)*(FA(Device 32)))

= 30*4.38 X 10^{06} *(1- 5 X 10^{-10})*((.25)*(.25)) = **$8.21e+006**

Total Annual Strategy Cost = Total Purchase Cost/10*((Screens(Level 1)+ Screens(Level 2)+FCCost+FACost

= (1.93e+007+912,000.00)/10*((20+16)+260,610.00+8.21e+006 = **$8.12e+007**

Now the Monkey Search algorithm will take this strategy and manipulate it by adding, subtracting, or switching devices.

Case 2: Baggage strategy of 3 levels

Device 22 has been added to the strategy.

Level 1: Device 13

Level 2: Device 32

Level 3: Device 22

Calculations:

Bags(Level 1) = 5000

Screens(Level 1) = Bags(Level 1)/Throughput(Device 13) = 5000/260 = 19.2308 = **20**

Cost(Level 1) = Screens(Level 1)*Cost(Device 13) = 20*965,000.00 = **$1.93e+007**

Bags(Level 2) = Bags(Level 1)*((FA(Device 13)*(1-PT)+(1-FC(Device 13)*(PT))

= 5000*((.25)*(1-5 X 10^{-10})+(1-.085)*(5 X 10^{-10})) = **1250**

Screens(level 2) = Bags(Level 2)/Throughput(Device 32) = 1250/80 = 15.625 = **16**

Cost(Level 2) = Screens(level 2)* Cost(Device 32) = 16*57,000.00 = **$912,000.00**

Bags(Level 3) = Bags(Level 2)*((FA(Device 32)*(1-PT)+(1-FC(Device 32)*(PT))

= 1250*((.25)*(1-5 X 10^{-10})+(1-.085)*(5 X 10^{-10})) = 31.25 = 312.5 **= 313**

Screens(Level 3) = Bags(Level 3)/Throughput(Device 22) = 313/180 = 1.73 = **2**

Cost(Level 3) = Screens(level 3)* Cost(Device 22) = 2*8,300.00 = **$16,600.00**

FCCost = Cost of False Clear*S*PT* ((FC(Device 13))+((FC(Device 32)*(1-FC(Device 13))+ ((FC(Device 22)*(1-FC(Device 13))*(1-FC(Device 32))

= 1, 400, 000, 000*4.38 X 10^{06}*5 X 10^{-10} *((.085)+((.085)*(1-.085))+ ((.085)*(1-.085))*(1-.085)) = **$716,831.00**

FACost = Cost of False alarm*S*(1-PT)*((FA(Device 13)*(FA(Device 32)*(FA(Device 22))

= 30*4.38 X 10^{06} *(1- 5 X 10^{-10})*((.25)*(.25)*(.20)) = **$1.6425e+006**

Total Annual Strategy Cost = Total Purchase Cost/10*(((Screens(Level 1)+ (Screens(Level 2))+ (Screens(Level 3)))+FCCost+FACost

= (1.93e+007+912,000.00+16,600.00)/10*((20+16+2)+716,831.00+1.6425e+006=

$7.9228e+007

Now this new cost is compared to the previous strategy of two levels cost of $8.12e+007. Since the new cost of three levels $7.9228e+007 is less than the previous cost, the more expensive strategy is discarded.

Case 3: Switching two devices.

Here the devices at Levels 1 and 2 have been switched.

Level 1: Device 32

Level 2: Device 13

Level 3: Device 22

Calculations:

Bags(Level 1) = 5000

Screens(Level 1) = Bags(Level 1)/Throughput(Device 32) = 5000/80 = 62.5 = **63**

Cost(Level 1) = Screens(Level 1)*Cost(Device 32) = 63*57,000.00 = **$3.591e+006**

Bags(Level 2) = Bags(Level 1)*((FA(Device 32)*(1-PT)+(1-FC(Device 32)*(PT))

= 5000*((.25)*(1-5 X 10^{-10})+(1-.085)*(5 X 10^{-10})) = **1250**

Screens(level 2) = Bags(Level 2)/Throughput(Device 13) = 1250/260 = 4.80 = **5**

Cost(Level 2) = Screens(level 2)* Cost(Device 13) = 5*965,000.00 = **$4.825e+006**

Bags(Level 3) = Bags(Level 2)*((FA(Device 13)*(1-PT)+(1-FC(Device 13)*(PT))

= 1250*((.25)*(1-5 X 10^{-10})+(1-.085)*(5 X 10^{-10})) = 312.5 = **313**

Screens(Level 3) = Bags(Level 3)/Throughput(Device 22) = 313/180 = 1.73 = **2**

Cost(Level 3) = Screens(level 3)* Cost(Device 22) = 2*8,300.00 = **$16,600.00**

FCCost = Cost of False Clear*S*PT* ((FC(Device 32))+((FC(Device 13)*(1-FC(Device 32))+ ((FC(Device 22)*(1-FC(Device 13))*(1-FC(Device 32))

= 1, 400, 000, 000*4.38 X 10^{06}*5 X 10^{-10} *((.085)+((.085)*(1-.085))+ ((.085)*(1-.085))*(1-.085)) = **$716,831.00**

FACost = Cost of False alarm*S*(1-PT)*((FA(Device 32)*(FA(Device 13)*(FA(Device 22))

= 30*4.38 X 10^{06} *(1- 5 X 10^{-10})*((.25)*(.25)*(.20)) = **$1.6425e+006**

Total Annual Strategy Cost = Total Purchase Cost/10*(((Screens(Level 1)+ (Screens(Level 2))+ (Screens(Level 3)))+FCCost+FACost

= (3.591e+006+4.825e+006+16,600.00)/10*((63+5+2)+716,831.00+1.6425e+006= **$6.1387e+007**

Finally, this new cost is compared to the previous strategy of three levels cost of $7.9228e+007. Since the new cost of three levels $6.1387e+007 is less than the previous cost, the more expensive strategy is discarded.

4.7 Monkey Search Algorithm Pseudo Code

In order to solve the baggage screening problem a Matlab program was developed using the following Monkey Search pseudo code:

Begin;

$MS\ G_{MS} = (V_{MS}, A, w), V_0, \max\ paths, \max\ levels;$

$let\ V = \{v_o\};$

let level=0, npaths=0;

let (termination=false);

While (termination=false) do

 # climbing up new tree branches

let npaths=npaths + 1;

While (level<=maxlevels) do

$x' = random\_perturb(y);$

$x'' = random\_perturb(y);$

if $(f(x') < f(y_{best}))$ *then*

$let\ y_{best} = x';$

end if

if $(f(x'') < f(y_{best}))$ *then*

$let\ y_{best} = x'';$

end if

end while

if (npaths < maxpaths) then

 # climbing down

$let\ w_{best} = w_{ylevel*ylevel-1};$

let p=0;

 # climbing up pre-visited tree branches

$let\ y = y_{current}$

while (y is not a leaf node)

$choose\_arc(w_{yx'}, w_{yx''});$

let level =level +1;

end while

else

let termination = true

end if

end while

4.8 Computational Results of Baggage Screening Problem

In order to solve the baggage screening problem using the previous data of Table 1, which contains 39 screening devices mentioned in Chapter 2 (Table 1), and the Monkey Search algorithm. A program was developed using the pseudo code of above in Section 4.7. The program was run in Matlab R2010a by Matworks® on a 2.7 GHz Intel Core i7m, 4GB 1333MHz DDR3 computer and using the following parameters:

- The height of the trees (h_t) = for each different level, levels 2 through 10.
- The number of times that the monkey reaches the top of the tree (n_t) = 50 times
- The size of the memory (n_m) = 20 best solutions
- The starting number of random trees (n_w) = 1000
- The number of iterations (n_i) = 100

The baggage screening strategies solutions, the completion time for each solution level and the total annual strategy cost for every different level obtained are summarized in the Table 3.

Table 3: Baggage Screening Solutions

Baggage Levels	Baggage Screening Strategy Devices	CPU	Total Cost
2 Levels	2 12	6 min	7.8766E+07
3 Levels	2 39 30	8 min	4.0650E+07
4 Levels	8 39 2 30	9 min	3.2502E+07
5 Levels	37 10 39 2 32	12 min	2.9781E+07
6 Levels	6 17 2 37 22 30	14 min	2.8798E+07
7 Levels	37 8 2 22 33 38 23	15 min	3.0823E+07
8 Levels	1 17 2 5 22 15 32 16	17 min	3.5294E+07
9 Levels	10 2 17 13 33 16 8 9 7	19 min	3.6637E+07
10 Levels	8 37 2 12 34 22 6 16 37 30	21 min	3.7484E+07

As shown in Table 3, the best solution using the parameters previously mentioned is: 6, 17, 2, 37, 22, and 30 at a cost of 2.8798e + 07 dollars. The best solution uses 6 levels of security where at the first level the device 6 is used. Since the device 6 has a throughput of 1000 bags per hour and the maximum number of bags expected per hour are 5000, using this solution at level 1 needs to be 5 machines using device 6. Then the alarmed bags will go to level 2 where the device 17 is used. In level 3, device 2 is used and so on until reaching level 6, where device 30 is used.

The second best solution found is: 37, 10, 39, 2, and 32 at a cost of 2.9781e+07 dollars. This solution uses 5 levels of security where at first level the device 37 is used. Since device 37 has a throughput of 1000 bags per hour also and the maximum number of bags expected per hour are 5000. Using this solution at level 1, 5 machines need to be situated again, using device 37. Then the bags alarmed will go to level 2, where device 10 is used and so on until reaching level 5 where device 32 is used.

The third best solution found is: 37, 8, 2, 22, 33, 38, and 23 at a cost of 3.0823e+07 dollars. This solution uses 7 levels of security where at first level device 37 is used again. Using this solution at level 1 needs to have 5 machines again. Then the bags alarmed will go to level 2, where device 8 is used, and so on until reaching level 7 where device 23 is used.

The worse solution found is: 2 and 12 at a cost of 7.8766e+07 dollars. This solution uses 2 levels of security where at level one device 2 is used. Since device 2 has a throughput of 600 bags per hour and the maximum number of bags expected per hour is 5000, using this solution at level 1 needs to be located 9 machines using device 2. Then the alarmed bags will go to level 2, where device 12 is used and there are no more security levels.

4.9 Sensitivity Analysis of Baggage Screening Problem

A sensitivity analysis was performed in which the parameters were changed to know how and where they affected the outcome of a solution. The first parameter to be changed was the starting number of random trees. By setting the starting number of random trees from 100 through 1100 trees at level 6. The best solution was found at 700 starting random trees with a completion time of 7 minutes and the total annual strategy cost is 2.8798E+07 with the following baggage strategy: 6, 17, 2, 37, 22, and 30. The second best solution was found at 1100 starting random trees, with a completion time of 18 minutes and the total annual strategy cost is 2.9167E+07 with the following baggage strategy: 37, 17, 2, 23, 6, and 31. The worst solution was found at 100 starting random trees with a completion time of 1 minute and the total annual strategy cost of 3.4718E+07 with the following baggage strategy: 6, 23, 5, 39, 23, and 16. All the solutions which were found and the completion times from 100 through 1100 trees are represented in Figure 22.

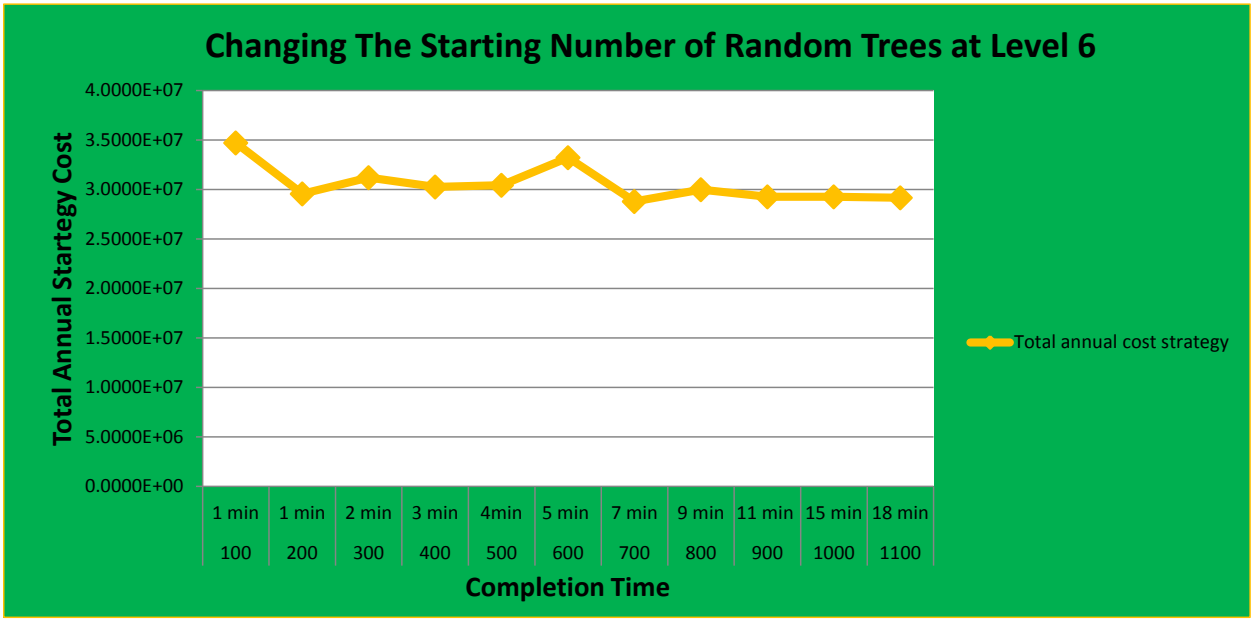

Figure 22: Sensitivity Analysis Changing the Starting Number of Random Trees.

Other parameter changed was the number of iterations, setting from 100 through 1000 iterations at level 5. The best solution was found at 900 iterations, with a completion time of 36 minutes and the total annual strategy cost is 2.8621E+07, with the following baggage strategy: 8, 10, 2, 39, and 33. The second best solution was found at 200 iterations with a completion time of 9 minutes and the total annual strategy cost is 2.8692E+07, with the following baggage strategy: 8, 10, 2, 22, and 30. The worst solution was found at 500 starting random trees with a completion time of 24 minutes and the total annual strategy cost of 2.9936E+07, with the following baggage strategy: 8, 2, 1, 13, and 30. Finally, increasing the number of iterations increases the duration of the completion time. All the solutions which were found and their completion time from 100 through 1000 iterations are represented in Figure 23.

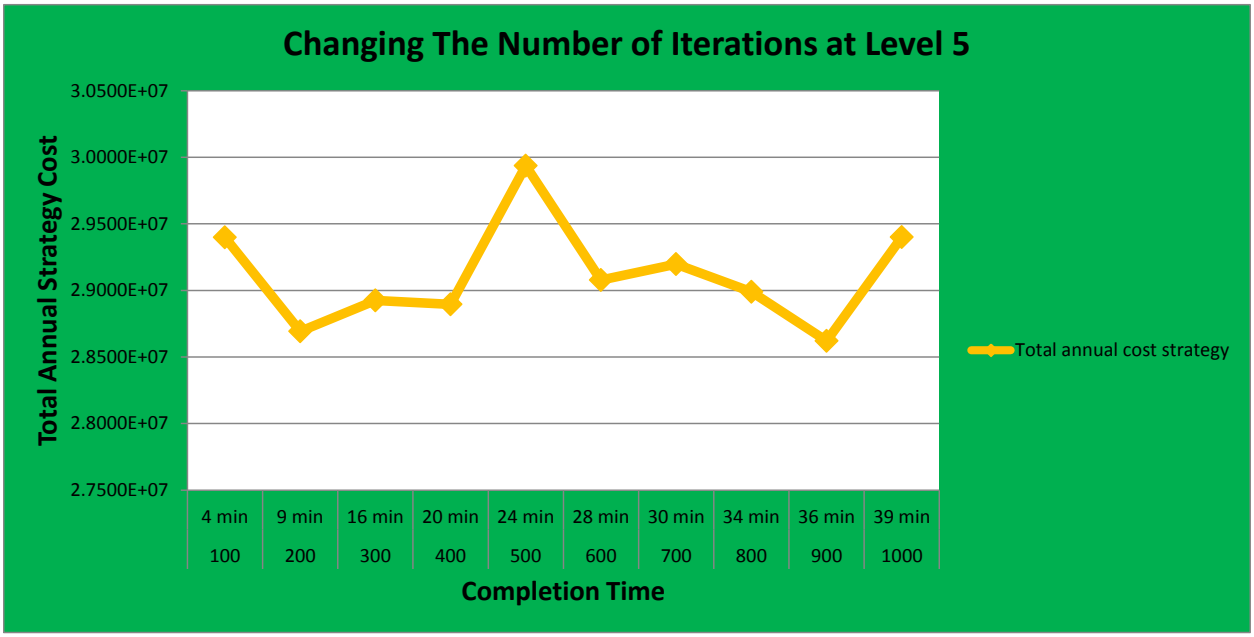

Figure 23: Sensitivity Analysis Changing the Number of Iterations.

CHAPTER 5: CONCLUSIONS AND FUTURE RESEARCH

After presenting the problem to be solved in Chapter 1 and Chapter 3, reviewing several meta-heuristic optimization methods in Chapter 2, a new meta-heuristic method was proposed to solve the baggage screening strategy problem in Chapter 4. In order to show how the algorithm works an example was presented for baggage screening problem using 39 devices and 10 different levels of security in Chapter 4.

This work introduced a single objective evolutionary algorithm to solve the baggage screening strategy problem considering the minimization of the total annual cost strategy. The developed algorithm is based on the resemblance of the behavior of a monkey searching for food by climbing up trees. The general assumption is that the monkey explores trees and learns which branches lead to better food resources. The found food resources, which are represented as desired solutions; are arranged in the order of the final solution value from objective function, and the branches of the tree of solutions are represented as perturbations between two neighboring feasible solutions. By saving the optimum solutions along with some other better solutions, a crossover is performed to replace inferior monkey's food resource with new and better locations. Then the algorithm continues to generate new populations of solutions until the stopping criteria reaches a specific number of trees.

The presented algorithm obtains as a solution a set of several security devices, this solution of the set is considered optimal. In addition, the Monkey Search algorithm can find the most optimal baggage strategy solution for the 10 different security levels and its total annual cost strategy using several variables and equations presented in Chapter 2. This type of problem has been studied in literature and different approaches have been proposed, *(Jacobson et al. 2004)* used simulated annealing algorithm for designing optimal aviation baggage screening

strategies, *(Mclay et al. 2005; Mclay et al. 2007)* used integer programming models for the deployment of baggage screening devices. Finally, *(Jacobson et al. 2004)* used integer programming for modeling and analyzing multiple station baggage screening security system performance.

The presented work considers several assumptions in order to simplify the problem. One of these assumptions considers that all the constants used in this model are estimates based on data from the Federal Aviation Administration such as: probability of a threat, expected annual operating cost of a device, cost of a system false alarm and cost of a system false clear. In addition, the cost of false alarm system and false clear system could vary depending on each scenario. The annual operating cost is just an average; each device has different costs associated with its operation. To finish, the average number of bags screened in a year is an approximation so the annual bags incoming to all airports may not be modeled in this way.

In summary, a new single objective evolutionary algorithm was developed to solve the baggage screening strategy problem considering the main objective of minimization of the total annual cost strategy. The results obtained shows that algorithm performs well in the 10 different security levels. In order to expand these work enhancements to the model include allowing the user to decide what cost should be minimized such as Total Annual Strategy Cost, False Alarm cost only, False Clear Cost only, Purchase Cost only and Operating Cost only. Another improvement is to add more devices attributes. Currently, the annual Operating Cost is a constant, but if operating cost will be added into the list of devices attributes, it would be a more realistic model. Finally, a new meta-heuristic optimization method will considered as a future research.

REFERENCES

Abshouri, A. (2011). New Firefly Algorithm Based on Multi Swarm And Learning Automated in Dynamic Enviroments. *IEEE Transactions on Evolutionary Computation*.

Baykasoglu, A., Dereli, T., & Sabuncu, I. (2004). Facility Layout Problems with Budget Constraints. *IEEE Transaction on Evolutionary Computation*.

Bean, J. (1994). Genetic Algorithm and Random Keys For Sequency and Optimization. *ORSA Journal on Computing*.

Behrang, M., Assareh, E., Assari, R., & Ghanbarzadeh, A. (2011). Using Bees Algorithm and Artificial Neural Network to Forecast World Carbon Dioxide Emission. *Taylor & Francis Group, LLC*.

Berrick, A. (2005). Better Planning Needed to Optimize Deployment of Checked Baggage Screening Systems. *Homeland Security and Justice*.

Bojic, I., & Kusek, M. (2010). Fireflies Synchronization in Small Overlay Networks. *Faculty of Electrical Engineering and Computing*.

Butler, V., & Poole, R. (2002). Re-Thinking Checked-Baggage Screening. *Reason Public Policy Institute*.

Candalino, T. (2001). Selecting A Baggage Screening Strategy to Meet Screening Requirements and Minimize Cost. *Texas Tech University*.

Candalino, T., Kobza, J., & Jacobson, S. (2004). Designing Optimal Aviation Baggage Screening Strategies Using Simulated Annealing. *Elsevier Science Ltd.* .

Carroll, D. (1996). Genetic Algorithms and Optimization Chemical Oxigen-Iodine Lasers. *IEEE Transaction On Evolutionary Computation*.

Chatterjec, S., Camera, C., & Lynch, L. (1995). Genetic Algorithm and Traveling Salesman Problems. *IEEE Transactions On Evolutionary Computation*.

Chen, P. (2009). Particle Swarm Optimization for power Dispach with Pumped Hydro. *IEEE transactions on Evolutionary Computation*.

Cheng, R., & Gen, M. (2000). *Genetic Algorithms and Engineering Optimization.* John Wiley & Sons, Inc.

Cortes, P., Garcia, J., Munuzuri, J., & Guadix, J. (2010). A Viral System Massive Infection Algorithm. *international Journal of Bio-inspired Computation*, 71-74.

Cortes., P., Garcia, J., Munuzuri, J., & Onieva, L. (2007). Viral Systems: A New Bio-Inspired Optimization Approach. *IEEE Transactions on Evolutionary Computation*.

Dhivja, M., Sundarambal, M., & Nithissh, L. (2011). Energy Efficient Computation of Data Fusion in Wireless Sensor Networks Using Cuckoo Based Particle Approach. *International J. Communications, Network and System Sciences*, 249-255.

Dobhansky, T. (1959). Blyth, Darwin, and Natural Selection. *American Naturalist*, 204-206.

Dorigo, M. (1992). Optimization, Learning and Natural Algorithms. *Ph.D. thesis , Dipartimento di Elettronica*.

Dorigo, M., & Gambardella, M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 29-41.

Holland, J. (1992). Genetic Algorithms: Computer Programs That "Envolve" in Ways That Ressemble Natural Selection Can Solve Complex Problems Even Their Creators Do Not Fully Understand. *Scientific American*.

Huang, Y., Qiu, Z., & Liu, Q. (2008). A Monkey-King Algorithm for Supply Chain Network Design Under Uncertainty. *World Congress on Intelligent Control and Automation*.

Ituarte-Villarreal, C., & Espiritu, J. (2011). Optimization of Wind Turbine Placement Using A Viral Based Optimization Algorithm. *Elsevier B.V.*

Jacobson, S., Mclay, L., Kobza, J., & Bowman, J. (2004). Modeling Analyzing Multiple Station Baggage Screening Security System Performance. *Springer Science + Businees Media, Inc.*

Kanoh, H., Matsumoto, M., Hasegawa, K., Kato, N., & Nishihara, S. (1997). Solving Constraint-Satisfaction Problems by a Genetic Algorithm Adopting Viral Infection. *Elsevier B.V.*

Karaboga, D. (2005). An Idea Based On Honey Bee Swarm For numerical optimization. *Tecnical Report-TR06*.

Kennedy, J., & Eberthart, R. (1995). Particle Swarm Optimization. *IEEE Transaction on Evolutionary Computation*, 1942-1948.

Kubota, N., Shzmojima, K., & Fukuda, T. (1996). Traveling Sales Problem. *IEEE Transaction on Evolutionary Computation*.

Kumar, A., & Chakarverty, S. (2011). Design Optimization for Reliable Embedded System Using Cuckoo Search. *IEEE International Conference on Electro/Information Technology*.

Kwannetr, U., Leeton, U., & Kulworawanichpong, T. (2010). Optimal Power Flow Using Artificial Bees Algorithm. *Power System Research Unit, Suranaree University Technology*.

Laukasik, S., & Zak, S. (2010). Firefly Algorithm for Continous Costranined Optimization Tasks. *Proceedings of the International Conference on Computer and Computational Intelligence* .

Lyon, D. (2006). Airport Screening, Surveillance, and Social Sorting: Canadian Responses to 9/11 in Context. *Federal Aviation Administration Conference*.

Mclay, L., Virta, J., Kobza, J., & Jacobson, S. (2003). Interger Programming Models for Deployment of Airport Baggage Screening Security Devices. *Springer Science + Business Media, Inc.*

Mead, K. (2002). *Challenges Facing the TSA in implementing the Aviation and Transportation Security Act.* Washington : Report Number CC-2002-088, Office of Inspector General, Departement of Transportation.

Mead, K. (2003a). *Transportation Security Administration Programs and Cost Controls.* Washington, DC.: Report Number CC-2003-066, Office of Inspector General, Department of Transportation.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant Colony Optimization for Resource-Constrained Project Scheduling. *IEEE Transaction on Evolutionary Computation*.

Misra, R., & Mandal, C. (2006). Complexity of the Optimal Data Aggregation in Wireless Sensor Network. *IEEE Transaction on Evolutionary Computation*.

Mucherino, A., & Seref, O. (2008). Monkey Search: A Novel Metaheuristic Search For Global Optimization. *Center for Applied Optimization, University of Florida.*

Mucherino, A., Liberti, L., Lavor, C., & Maculan, N. (2009). Comparison Between an Exact and a MetaHeuristic Algorithm for the Molecular Distance Geometry Problem. *GECCO*.

Nikolaev, A., Jacobson, S., & Mclay, L. (2007). A Sequential Stochastic Security System Design Problem for Aviation Security. *Transportation Science*.

Nikolay, Y., & Hitoshi, I. (2006). *Adaptive Learning of Polynomial Networks; Genetic Programming.* New York: Springer.

Othman, N., Musirin, I., Rahim, A., & Othman, Z. (2010). Bees Algorithm Technique for Loss Minization in Power Transmission Network Using Static Var Compesator. *IEEE Transaction on Evolutionary Computation*.

Pan, Y., Kumar, P., Kim, J., Memik, G., Zhang, Y., & Choudhary, A. (2009). Firefly: Illuminating Future Network on Chip with Nanophotonics. *Northwestern University*.

Pham, D., Koc, E., Lee, Y., & Phrueksanant, J. (2008). Using the Bees Algorithm to Schedule Jobs for a Machine. *Manufacturing Engineering Centre, Cardiff University, Cardiff CF24*.

Rao, E., & Dickey, R. (1999). Security Technology. *IEEE Iternational Carnahan Conference on Aviation Security*, 158-167.

Robinson, J., & Rahmat-Samii, Y. (2004). Particle Swarm Optimization in Electromagnetics. *IEEE transactions on Evolutionary Computation*.

Ross, T. (1995). Fuzzy Logic WIth Engineering Applications. *Mcgraw Hill*.

Rybicki, E. (2007). A Feeling for the Molechism. *Departmenet of Molecular and Cell Biology*.

Sayadia, K., Ramezaniana, R., & Ghaffari-Nasaba, N. (2010). A Discrete Firefly Meta-Hueristic with Local Search for Makespan Minimization in Pertubations Flow Shop Scheduling Problems. *IEEE Transactions on Evolutionary Computation*.

Selvakumar, I., & Thanushkodi, K. (2007). A New Particle Swarm Optimization Solution to Nonconvex Economic Dispatch Problems. *IEEE Transactions on Power Systems*.

Tuba, M., Subotic, M., & Stanarevic, N. (2010). Modified Cuckoo Search Algorithm for Unconstrained Optimization Problems. *Proceedings of the European Computing Conference*.

Virta, J., Sheldon , H., Jacobson, D., & John, E. (2001). Analyzing the Cost of Screening Selected and Non-Selected Baggage. *University of Illinois*.

Wang, J., Yu, Y., Zeng, Y., & Luan, W. (2009). Discrete Monkey Algorithm and Its Application in Transmission Network Expasion Planning. *IEEE Transaction on Evolutionary Computation*.

Yang, X. (2009). Firefly Algorithms for Multimodal Optimization. *Springer-Verlag* , 169-178.

Yang, X. (2010). Firefly Algorithm, Stochastic Test Functions and Design Optimization. *International Journal of Bio-Inspired Computation*, 78-84.

Yang, X., & Deb, S. (2009). Cuckoo Search Via L'evy Flights. *Elsevier B.V.*

Yang, X., Gandomi, H., & Alavi, H. (2011). Cuckoo Search Algorithm: a Metaheuristic Approach to Solve Structural Optimization Problems. *Springer-Verlag*.

Yoshida, H., Kawata, Y., Fukuyama, Y., & Nakanishi, Y. (1999). A Particle Swarm Optimization For Reactive Power and Voltage Control Considering Voltage Stability. *IEEE International Conference on Intelligent System Applications to Power Systems*.

Zhang, J., Yan, Y., & Lin, Y. (2009). Research on Fault Diagnosis Based on SVM and Monkey-King Genetic Algorithm. *World Congress on Intelligent Control and Automation*.

APPENDIX A
MATLAB CODE OF MONKEY SEARCH ALGORITHM

```
%Number of initial solutions
mejoresmejoresmejores=[];
mejorsubirpx=[];
for changos=1:100

M=1;

%Number of maximum  levels

MSL=5;

%Cost Matrix [cost, FA, FC, Throughtput]

MC=[ 900000      0.4 0.075   1200
330000           0.25        0.095   600
250000           0.2 0.085   100
850000           0.15        0.075   100
850000           0.3 0.065   254
3000             0.5 0.055   1000
2000             0.1 0.0545  2
500              0.5 0.0535  1000
30000            0   0.0825  10
5000             0.5 0.075   1000
965000           0.25        0.075   50
850000           0.15        0.095   251
965000           0.25        0.085   260
200000           0.2 0.075   62
80000            0.25        0.065   80
70000            0.15        0.0755  80
90000            0.5 0.0835  1000
560000           0.2 0.0525  24
80000            0.4 0.075   600
800000           0.15        0.05    50
50               0.2 0.095   1
8300             0.2 0.085   180
16600            0.3 0.075   360
1500000          0   0.065   12
965000           0.15        0.0755  120
850000           0.15        0.0945  100
330000           0.18        0.0435  20
450000           0.25        0.0825  100
2000000          0.25        0.053   62
80000            0.12        0.095   80
57000            0.1 0.095   57
57000            0.25        0.085   80
70000            0.15        0.075   80
```

```
45000           0.15        0.0965 66
200000          0   0.055   1
600000          0.2 0.06    80
60000           0.5 0.0635 1000
200             0.1 0.0825 12
450000          0.25        0.075   550];

% Size matrix

[ii,jj]=size(MC);

%initial population
lev=[];
MLIJ=[];
for i=1:M
   MLI=[];

   MSLR=ceil((MSL-1)*rand());
   for k=1:MSLR
      ML=ceil((ii*(rand())));
      MLI=[MLI,ML];
      falt=MSL-MSLR;
      complete=zeros(1,falt);

   end
   MLI=[MLI complete];
   MLIJ=[MLIJ; MLI];
   lev=[lev; MSLR];
end

MLIJ;
%evaluation
%MLIJ=[34,21,21];
%lev=[3];


mejoresmejores=[];
for iteraciones=1:1000
g=1;
PT=5*(10^-10);
bags=5000;
S=365*24*(bags/10);
falsclear=1400000000;
falsalarm=30;
CO=125000;
```

```
totalstratcostk=[];

FCcosth=0;
FCcostm=1;
bags=5000;

 Screenslevel1=ceil(bags/MC(MLIJ(g,1),4));
 costlevel1=Screenslevel1*(MC(MLIJ(g,1),1));
switch lev(g,1)

   case 1

 Screenslevel1=ceil(bags/MC(MLIJ(g,1),4));
 costlevel1=Screenslevel1*(MC(MLIJ(g,1),1));
 FCcostlevel1=round(PT*S*falsclear*((MC(MLIJ(g,1),3))*(1-MC(MLIJ(g,1),3))));
 FAcostlevel1=round(round(falsalarm*S*(1-PT)*((MC(MLIJ(g,1),2)))));
 Totalstratcost=((costlevel1)/10)+CO*(Screenslevel1)+FCcostlevel1+FAcostlevel1;

   case 2

      bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
      screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
      costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

      FC=(PT*S*falsclear)*((MC(MLIJ(g,1),3)+MC(MLIJ(g,2),3))*((1-MC(MLIJ(g,1),3))*(1-
MC(MLIJ(g,2),3))));
      FAcostlevel=round(falsalarm*S*(1-PT)*(MC(MLIJ(g,1),2)*MC(MLIJ(g,2),2)));

Totalstratcost=((costlevel1+costlevel2)/10)+CO*(Screenslevel1+screenlevel2)+FC+FAcostlevel;


   case 3
      bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
      screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
      costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

      bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
      screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
      costlevel3=screenlevel3*(MC(MLIJ(g,3),1));

      FC=(PT*S*falsclear)*((MC(MLIJ(g,1),3)+MC(MLIJ(g,2),3)+MC(MLIJ(g,3),3))*((1-
MC(MLIJ(g,1),3))*(1-MC(MLIJ(g,2),3))*(1-MC(MLIJ(g,3),3))));
      FAcostlevel=round(falsalarm*S*(1-
PT)*(MC(MLIJ(g,1),2)*MC(MLIJ(g,2),2)*MC(MLIJ(g,3),2)));
```

```
Totalstratcost=((costlevel1+costlevel2+costlevel3)/10)+CO*(Screenslevel1+screenlevel2+screen
level3)+FC+FAcostlevel;

   case 4
      bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
      screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
      costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

      bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
      screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
      costlevel3=screenlevel3*(MC(MLIJ(g,3),1));

      bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
      screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
      costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

        FCsum=0;
      FCmul=1;
      for kk=1:lev(g,1)
      FCsum=(FCsum+MC(MLIJ(g,kk),3));
      FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
      end
      FC=(PT*S*falsclear)*(FCsum*FCmul);

      FAmul=1;
      for ll=1:lev(g,1);
      FAmul=(FAmul*MC(MLIJ(g,ll),2));
      end
      FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);



Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4)/10)+CO*(Screenslevel1+screenle
vel2+screenlevel3+screenlevel4)+FC+FAcostlevel;

   case 5

      bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
      screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
      costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

      bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
      screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
      costlevel3=screenlevel3*(MC(MLIJ(g,3),1));
```

```
bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

bags5= bags4*(MC(MLIJ(g,4),2))*(1-PT)+(1-MC(MLIJ(g,4),3)*PT);
screenlevel5=ceil(bags5/MC(MLIJ(g,5),4));
costlevel5=screenlevel5*(MC(MLIJ(g,5),1));

FCsum=0;
FCmul=1;
for kk=1:lev(g,1)
FCsum=(FCsum+MC(MLIJ(g,kk),3));
FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
end
FC=(PT*S*falsclear)*(FCsum*FCmul);

FAmul=1;
for ll=1:lev(g,1)
FAmul=(FAmul*MC(MLIJ(g,ll),2));
end
FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);
```

```
Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4+costlevel5)/10)+CO*(Screenslevel
1+screenlevel2+screenlevel3+screenlevel4+screenlevel5)+FC+FAcostlevel;
```

```
case 6

bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
costlevel3=screenlevel3*(MC(MLIJ(g,3),1));

bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

bags5= bags4*(MC(MLIJ(g,4),2))*(1-PT)+(1-MC(MLIJ(g,4),3)*PT);
screenlevel5=ceil(bags5/MC(MLIJ(g,5),4));
costlevel5=screenlevel5*(MC(MLIJ(g,5),1));
```

```
bags6= bags5*(MC(MLIJ(g,5),2))*(1-PT)+(1-MC(MLIJ(g,5),3)*PT);
screenlevel6=ceil(bags6/MC(MLIJ(g,6),4));
costlevel6=screenlevel6*(MC(MLIJ(g,6),1));

FCsum=0;
FCmul=1;
for kk=1:lev(g,1)
FCsum=(FCsum+MC(MLIJ(g,kk),3));
FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
end
FC=(PT*S*falsclear)*(FCsum*FCmul);

FAmul=1;
for ll=1:lev(g,1)
FAmul=(FAmul*MC(MLIJ(g,ll),2));
end
FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);



Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4+costlevel5+costlevel6)/10)+CO*(
Screenslevel1+screenlevel2+screenlevel3+screenlevel4+screenlevel5+screenlevel6)+FC+FAcost
level;

   case 7

bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
costlevel3=screenlevel3*(MC(MLIJ(g,3),1));

bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

bags5= bags4*(MC(MLIJ(g,4),2))*(1-PT)+(1-MC(MLIJ(g,4),3)*PT);
screenlevel5=ceil(bags5/MC(MLIJ(g,5),4));
costlevel5=screenlevel5*(MC(MLIJ(g,5),1));

bags6= bags5*(MC(MLIJ(g,5),2))*(1-PT)+(1-MC(MLIJ(g,5),3)*PT);
screenlevel6=ceil(bags6/MC(MLIJ(g,6),4));
costlevel6=screenlevel6*(MC(MLIJ(g,6),1));
```

```
bags7= bags6*(MC(MLIJ(g,6),2))*(1-PT)+(1-MC(MLIJ(g,6),3)*PT);
screenlevel7=ceil(bags7/MC(MLIJ(g,7),4));
costlevel7=screenlevel7*(MC(MLIJ(g,7),1));

FCsum=0;
FCmul=1;
for kk=1:lev(g,1)
FCsum=(FCsum+MC(MLIJ(g,kk),3));
FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
end
FC=(PT*S*falsclear)*(FCsum*FCmul);

FAmul=1;
for ll=1:lev(g,1);
FAmul=(FAmul*MC(MLIJ(g,ll),2));
end
FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);



Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4+costlevel5+costlevel6+costlevel7)
/10)+CO*(Screenslevel1+screenlevel2+screenlevel3+screenlevel4+screenlevel5+screenlevel6+s
creenlevel7)+FC+FAcostlevel;

    case 8

bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
costlevel3=screenlevel3*(MC(MLIJ(g,3),1));

bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

bags5= bags4*(MC(MLIJ(g,4),2))*(1-PT)+(1-MC(MLIJ(g,4),3)*PT);
screenlevel5=ceil(bags5/MC(MLIJ(g,5),4));
costlevel5=screenlevel5*(MC(MLIJ(g,5),1));

bags6= bags5*(MC(MLIJ(g,5),2))*(1-PT)+(1-MC(MLIJ(g,5),3)*PT);
screenlevel6=ceil(bags6/MC(MLIJ(g,6),4));
costlevel6=screenlevel6*(MC(MLIJ(g,6),1));
```

```
bags7= bags6*(MC(MLIJ(g,6),2))*(1-PT)+(1-MC(MLIJ(g,6),3)*PT);
screenlevel7=ceil(bags7/MC(MLIJ(g,7),4));
costlevel7=screenlevel7*(MC(MLIJ(g,7),1));

bags8= bags7*(MC(MLIJ(g,7),2))*(1-PT)+(1-MC(MLIJ(g,7),3)*PT);
screenlevel8=ceil(bags8/MC(MLIJ(g,8),4));
costlevel8=screenlevel8*(MC(MLIJ(g,8),1));

  FCsum=0;
FCmul=1;
for kk=1:lev(g,1)
FCsum=(FCsum+MC(MLIJ(g,kk),3));
FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
end
FC=(PT*S*falsclear)*(FCsum*FCmul);

FAmul=1;
for ll=1:lev(g,1)
FAmul=(FAmul*MC(MLIJ(g,ll),2));
end
FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);



Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4+costlevel5+costlevel6+costlevel7+
costlevel8)/10)+CO*(Screenslevel1+screenlevel2+screenlevel3+screenlevel4+screenlevel5+scre
enlevel6+screenlevel7+screenlevel8)+FC+FAcostlevel;

    case 9

bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
costlevel3=screenlevel3*(MC(MLIJ(g,3),1));

bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

bags5= bags4*(MC(MLIJ(g,4),2))*(1-PT)+(1-MC(MLIJ(g,4),3)*PT);
screenlevel5=ceil(bags5/MC(MLIJ(g,5),4));
costlevel5=screenlevel5*(MC(MLIJ(g,5),1));
```

```
bags6= bags5*(MC(MLIJ(g,5),2))*(1-PT)+(1-MC(MLIJ(g,5),3)*PT);
screenlevel6=ceil(bags6/MC(MLIJ(g,6),4));
costlevel6=screenlevel6*(MC(MLIJ(g,6),1));

bags7= bags6*(MC(MLIJ(g,6),2))*(1-PT)+(1-MC(MLIJ(g,6),3)*PT);
screenlevel7=ceil(bags7/MC(MLIJ(g,7),4));
costlevel7=screenlevel7*(MC(MLIJ(g,7),1));

bags8= bags7*(MC(MLIJ(g,7),2))*(1-PT)+(1-MC(MLIJ(g,7),3)*PT);
screenlevel8=ceil(bags8/MC(MLIJ(g,8),4));
costlevel8=screenlevel8*(MC(MLIJ(g,8),1));

bags9= bags8*(MC(MLIJ(g,8),2))*(1-PT)+(1-MC(MLIJ(g,8),3)*PT);
screenlevel9=ceil(bags9/MC(MLIJ(g,9),4));
costlevel9=screenlevel9*(MC(MLIJ(g,9),1));

 FCsum=0;
FCmul=1;
for kk=1:lev(g,1)
FCsum=(FCsum+MC(MLIJ(g,kk),3));
FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
end
FC=(PT*S*falsclear)*(FCsum*FCmul);

FAmul=1;
for ll=1:lev(g,1)
FAmul=(FAmul*MC(MLIJ(g,ll),2));
end
FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);



Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4+costlevel5+costlevel6+costlevel7+
costlevel8+costlevel9)/10)+CO*(Screenslevel1+screenlevel2+screenlevel3+screenlevel4+screen
level5+screenlevel6+screenlevel7+screenlevel8+screenlevel9)+FC+FAcostlevel;

   case 10

    bags2= bags*(MC(MLIJ(g,1),2))*(1-PT)+(1-MC(MLIJ(g,1),3)*PT);
    screenlevel2=ceil(bags2/MC(MLIJ(g,2),4));
    costlevel2=screenlevel2*(MC(MLIJ(g,2),1));

    bags3= bags2*(MC(MLIJ(g,2),2))*(1-PT)+(1-MC(MLIJ(g,2),3)*PT);
    screenlevel3=ceil(bags3/MC(MLIJ(g,3),4));
    costlevel3=screenlevel3*(MC(MLIJ(g,3),1));
```

```
bags4= bags3*(MC(MLIJ(g,3),2))*(1-PT)+(1-MC(MLIJ(g,3),3)*PT);
screenlevel4=ceil(bags4/MC(MLIJ(g,4),4));
costlevel4=screenlevel4*(MC(MLIJ(g,4),1));

bags5= bags4*(MC(MLIJ(g,4),2))*(1-PT)+(1-MC(MLIJ(g,4),3)*PT);
screenlevel5=ceil(bags5/MC(MLIJ(g,5),4));
costlevel5=screenlevel5*(MC(MLIJ(g,5),1));

bags6= bags5*(MC(MLIJ(g,5),2))*(1-PT)+(1-MC(MLIJ(g,5),3)*PT);
screenlevel6=ceil(bags6/MC(MLIJ(g,6),4));
costlevel6=screenlevel6*(MC(MLIJ(g,6),1));

bags7= bags6*(MC(MLIJ(g,6),2))*(1-PT)+(1-MC(MLIJ(g,6),3)*PT);
screenlevel7=ceil(bags7/MC(MLIJ(g,7),4));
costlevel7=screenlevel7*(MC(MLIJ(g,7),1));

bags8= bags7*(MC(MLIJ(g,7),2))*(1-PT)+(1-MC(MLIJ(g,7),3)*PT);
screenlevel8=ceil(bags8/MC(MLIJ(g,8),4));
costlevel8=screenlevel8*(MC(MLIJ(g,8),1));

bags9= bags8*(MC(MLIJ(g,8),2))*(1-PT)+(1-MC(MLIJ(g,8),3)*PT);
screenlevel9=ceil(bags9/MC(MLIJ(g,9),4));
costlevel9=screenlevel9*(MC(MLIJ(g,9),1));

bags10= bags9*(MC(MLIJ(g,9),2))*(1-PT)+(1-MC(MLIJ(g,9),3)*PT);
screenlevel10=ceil(bags10/MC(MLIJ(g,10),4));
costlevel10=screenlevel10*(MC(MLIJ(g,10),1));

FCsum=0;
FCmul=1;
for kk=1:lev(g,1)
FCsum=(FCsum+MC(MLIJ(g,kk),3));
FCmul=(FCmul*(1-MC(MLIJ(g,kk),3)));
end
FC=(PT*S*falsclear)*(FCsum*FCmul);

FAmul=1;
for ll=1:lev(g,1)
FAmul=(FAmul*MC(MLIJ(g,ll),2));
end
FAcostlevel=round(falsalarm*S*(1-PT)*FAmul);


Totalstratcost=((costlevel1+costlevel2+costlevel3+costlevel4+costlevel5+costlevel6+costlevel7+
costlevel8+costlevel9+costlevel10)/10)+CO*(Screenslevel1+screenlevel2+screenlevel3+screenl
```

```
evel4+screenlevel5+screenlevel6+screenlevel7+screenlevel8+screenlevel9+screenlevel10)+FC+
FAcostlevel;

end

totalstratcostk=[totalstratcostk;Totalstratcost];

totalstratcostk=[MLIJ totalstratcostk];

inicialmemoria=[totalstratcostk];
MLIJg=MLIJ;
```

CURRICULUM VITA

Edgar Jimenez was born in Chihuahua, Chih., Mexico. The third of four offspring of Cirilo Jimenez and Bertha Estrada. He received his bachelor degree in Industrial Engineering in the University of Texas at El Paso in spring 2009. Then he started his professional career as a Quality Engineer in an electric manufacturing company called EMERSON in Juarez, Chihuahua. Later, he entered to the University of Texas at El Paso in fall 2010. While pursuing a master's degree in industrial engineering, he worked with Dr. Jose Espiritu as his Research Assistant. He had the opportunity to present his research about quality control in flexible pavement in the 2010 IIE Annual Conference and Expo in Reno, Nevada. Currently, he is working in Dal-Tile Corporation as a Process Engineer in El Paso, Tx.