 UNIVERSIDAD PANAMERICANA FACULTAD DE INGENIERÍA	Materia: Inteligencia Artificial Reporte del Proyecto Final Profesor: Dr. Félix Orlando Martínez Ríos Fecha: 25 de mayo de 2020

“Optimization effectiveness of metaheuristic algorithms and the most suitable one for reconstruction of microwave images”

Nombres y IDs: **Andrea Hinojosa Albornoz (0218911)**
Manuel Kanahuati Ceballos (0187323)
Braulio Rodríguez Pérez (0204510)

ID	Nombre	Carrera	a-eng-UEC	a-eng-ACSEP	a-eng-IR	c-env-A	c-env-D	c-manuf-AP	c-manuf-EM	k-modern-	k-modern-S
0218911	Andrea Hinojosa Albornoz	I.T.I.S.I									
0187323	Manuel Kanahuati Ceballos	I.T.I.S.I									
0204510	Braulio Rodríguez	I.T.I.S.I									

Abstract - Optimization is highly used nowadays for different problems and researches. With the exponential growth of data amount, various algorithms (like GA, PSO or SFL) are used to optimize a problem with different functions. GA, PSO or SFL algorithms will be tested with various functions and be compared in order to get the best algorithm for each function. All the algorithms will be implemented in R. The best algorithm will be selected as the most suitable solution in microwave imaging reconstruction, which is a high dimensional optimization problem.

Key Terms - Genetic Algorithm (GA), Particle Swarm Optimization (PSO), SFL, Optimization methods, microwave imaging.

I. Introduction:

Optimizing an objective function refers to finding the best solution under some criteria such as cost. The best solution refers to a set of values that maximizes or minimizes the value of our function. Continuous optimization can be applied only to real and derived functions. Obtaining an exact solution through an analytical procedure is not practical for functions that can be very complicated, so computational calculation is used using algorithms focused on this, based in successive iterations and degrees of precision in order to find the best solution for the studied function.

There are real problems that require a solution that is not always easy, including: finding the shortest path between different points, finding the minimum cost of a route, the optimal assignment of tasks or jobs to carry out a production chain, among others, and using exact algorithms to find the solutions are inefficient or impossible to apply.

Metaheuristic algorithms are mainly used to solve function optimization problems. They are iterative and they guide a heuristic, intelligently combining different concepts to correctly explore the search space. These algorithms to achieve a good search development use intensification and diversification. The first is to focus the search in the space where you have already found good solutions and from there build new solutions. Diversification prevents the algorithm from staying in a single search space where it can get stuck. It is important that the algorithm builds on these concepts and strikes a good balance between them.

One way to classify metaheuristic algorithms is between population-based and non-population-based algorithms. In the case of non-population-based methods, a single solution is used as a starting point and the path to finding the final solution is described. An example is the simulated annealing algorithm, in which a single solution is generated at the beginning and another one is created based on it; from these two solutions the values in the objective function are evaluated. At the end of the iterations you are left with the solution that best optimized the function. Population-based methods use a series of initial solutions (subjects from a population) and simultaneously optimize them to find solutions from different starting points. During the process of a population-based algorithm, the subjects of a population are manipulated and combined until a new improved one is generated. These methods offer higher quality solutions than non-population based methods because they optimize by searching from different starting points. An example is the genetic algorithms in which you start not only with a solution but with a set of individuals (pairs) and from these subjects solutions from the population, you make a mutation process between them to repopulate (a new population evolved).

II. Problem:

In this paper we will try to find the most effective metaheuristic algorithm for optimization so it can be used for high-dimensional microwave image reconstruction. We will test three algorithms GA, PSO and SFL using 13 different fitness functions and comparing their optimization performance so we can conclude which is the best optimizing algorithm. This algorithm will be the one selected to optimize the objective function used in high-dimensional microwave image reconstruction.

To achieve the results each algorithm will be tested as follows. Each algorithm will run 30 times for each objective function. We will obtain the mean and standard deviation value of the 30 optimum values of each function with the three algorithms. In conclusion we will compare each of the 13 functions between the three algorithms using the mean value each algorithm got after the 30 observations.

All tests will be done using a population of 40 subjects and the range values the variables can take are different for each objective function (this is described in Section VI).

Advances in medical imaging have been interested on microwave imaging. Medical imaging is the technique used to obtain visual representations of the interior of a human body. Microwave imaging allows to make non-invasive evaluation on a specific object, like a diagnosis of human physiologies. The reconstruction of this images represents a complex scattering problem. The scattering problem is to recover qualities of an object measured under the illumination of a wave. This is a problem when we need good details on the composition of an object. This problem is solved iteratively, where the error between the measured and the scattered data will be minimized at the end of each iteration. This helps the measured data be more like the real object. The objective function the most suitable algorithm would minimize is the normalized root mean square error between the measured and computed values of the scattered field:

$$OF = \sqrt{\frac{1}{V}\sum_{1}^{V}\frac{\left|\vec{E}*_{meas}(\rho,\phi,z) - \vec{E}*_{comp}(\rho,\phi,z)\right|^2}{\left|\vec{E}*_{meas}(\rho,\phi,z)\right|^2}}$$

$V \rightarrow$ *total number of receivers used in the imaging system*

$\overrightarrow{E*_{meas}} \rightarrow$ *are the measured values in location* ρ, ϕ, z

$\overrightarrow{E*_{comp}} \rightarrow$ *are the computed values in location* ρ, ϕ, z

Population-based algorithms are being used to reconstruct microwave images. Where the global optimization problem is the imaging problem, each subject has its own fitness values for the defined objective function. The algorithm will reconstruct the image by searching for the optimal solution. The dimensionality of the problem is often very high in order to reconstruct detailed images and this affects the computational cost (this being very high).

This paper is organized as follows. Section III includes the description of each metaheuristic algorithm used (GA, PSO and SFL). Section IV includes the R library used to implement the three algorithms and the description of the parameters each function uses. Section V includes the source code used for each algorithm followed by Section VI where all 13 fitness functions (which are the ones that will help us find the most effective metaheuristic algorithm for optimization) are described and their source code (for R implementation) are also found there. Section VII includes all the results obtained by the tests where we will conclude which algorithm would be more suitable for the microwave image reconstruction.

III. Metaheuristic algorithm description:

Genetic Algorithm (GA)

Genetic algorithms (GAs) are stochastic search algorithms inspired by the basic principles of biological evolution and natural selection. Genetic Algorithms simulate the evolution of living organisms, where the fittest individuals dominate over the weaker ones, by mimicking the biological mechanisms of evolution, such as selection, crossover and mutation.

GA is a population algorithm. This population is build up , randomly, with artificial chromosomes. Each chromosome represents a possible solution and are evaluated by a fitness function. After being evaluated, chromosomes are passed through more steps, so they can evolve.

A typical design for a classical GA might be as follows:

1)Initialization: Randomly generate a initial population of n chromosomes.

2)Fitness calculation: Chromosome are evaluated with a given function.

3)Selection: The algorithm selects the better solutions.

4)Crossover: There are many different kinds of crossover, the most common type is single point crossover. In single point crossover, chooses a locus at which you swap the remaining alleles from one parent to the other.

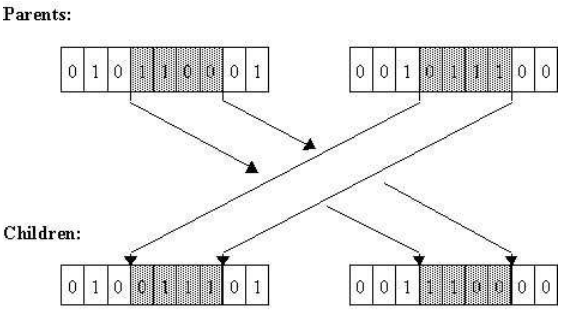


Figure 1:: Crossover example

5)Mutation: After selection and crossover, we get new population full of individuals. In order to

ensure that the individuals are not all exactly the same, you allow for a small chance of mutation.

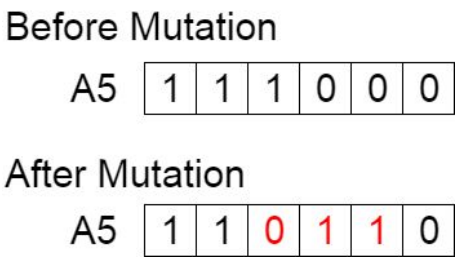


Figure 2:: Mutation example

6)Calculates the fitness of each individual and updates the population.

7) If the new population satisfies the criteria, it stops. If not, start from selection step with the new population.

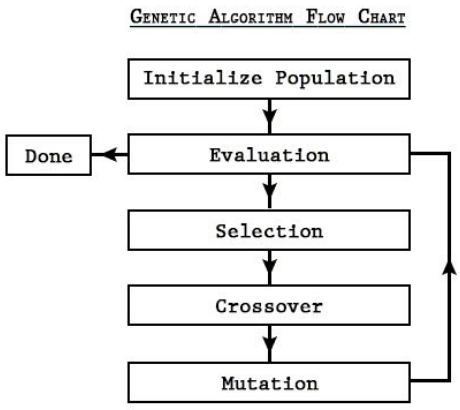


FIGURE 2

Figure 3:: Flow chart of GA

Particle Swarm Optimization (PSO)

Refers to Particle Swarm Optimization. Which works as a continuous optimization technique for functions. This metaheuristic algorithm is based on social activities presented by insects, birds, and fish. It is specifically based on the natural process of group communication this kind of animals present. This communication is used to share individual information when swarms move or migrate. When one member of the group finds an optimum place of movement the whole group will instantly follow it. The particles of the swarm will always tend to go to the most optimum place they have found as individuals and as a swarm, this is caused because the position of each particle is updated using its historical behavior as an individual and of the whole group. This algorithm was proposed by Kennedy and Eberhart in 1995.

It is a population algorithm where each individual has a position and a speed associated (both being vectors of size d). Where d is the dimension of the problem. Besides, each individual has an inertia value that keeps them in a direction of movement which depends on:

a) The personal best: The particle is attracted to the best location it has ever had.

b) The global best: The particle is attracted to the best location found by the set of particles.

The inertia weight proposed by Shi and Eberhart is used to increase PSO performance

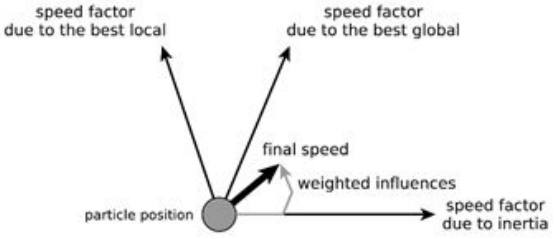


Figure 1: Values related to each individual in the population

The force with which each particle is pushed in each direction depends on two predefined parameters (its attraction to its best personnel and its attraction to the best global). Updating the locations of the particles is done as follows:

A) First, the inertia weight is updated:

$$w = w_{max} - \frac{w_{max} - w_{min}}{Iter_{max}} \times Curent\ Iteration$$

w_{max} *and* $w_{min} \rightarrow$ *maximum inertia and minimun inertia*

$Iter_{max} \rightarrow$ *The maximum number of iterations*

Current Iteration

B) The speed of each particle is updated with the following equation:

$$V_i^{k+1} = wV_i^k + c_1 r_1 \left((P_{best})_i^k - X_i^k\right) + c_2 r_2 \left((G_{best})_i^k - X_i^k\right)$$

$V_i^k \rightarrow speed\ of\ particle\ i\ in\ iteration\ k$

$c_1 \rightarrow\ attraction\ constant\ \ to\ personal\ best$

$c_2 \rightarrow\ attraction\ constant\ \ to\ swarm´s\ best$

$r_1\ and\ r_2 \rightarrow random\ numbers\ between\ 0\ and\ 1$

$X_i^k \rightarrow position\ of\ particle\ i\ in\ iteration\ k$

$(Pbest)_i^k \rightarrow best\ postition\ of\ particle\ i\ in\ iteration\ k$

$(Gbest)_i^k \rightarrow best\ postition\ the\ swarm\ has\ had\ until\ now$

Once the velocity is updated, the position of the particle is now updated with the following equation:

$$X_i^{k+1} = X_i^k + V_i^{k+1}$$

From equation A) we can conclude that on every iteration the value of the inertia decreases, so the speed of every particle will also decrease causing that the change of value (position) of every particle, as the algorithm goes, will be smaller.

The steps to find the optimal solution are as follows:

I. Initialization: Initializes the first population of particles and their speeds.
II. The suitability of the particles is calculated and the best position is found as best global and best local.
III. Update speed: In each iteration, speed depends on two things, the best global and the best personal.
IV. Update the position of the particles. After calculating the new velocity, the position of the particle changes.
V. If the iteration is less than the maximum defined iteration, go back to step II.
VI. Check the termination criteria, if the termination criterion is met, return the best overall as the optimal solution for a given problem.

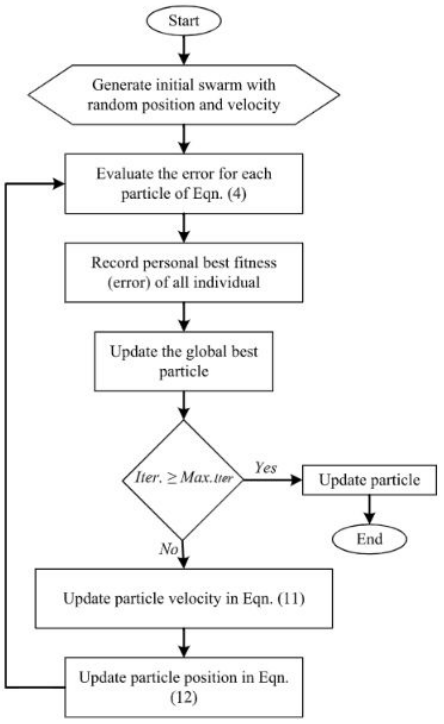


Figure 2: Flow diagram of the PSO algorithm

Shuffled Frog Leaping Algortithm (SFL)

SFLA is a recently popular meta-heuristic based on the memetic evolution of a group of frogs when seeking for the location that has the maximum amount of available food. Proposed in 2003, SFLA is the combination of the merits of memetic algorithm (MA) and PSO.

In SFLA, the population consists of a group of frogs (solutions) that is partitioned into subsets (memeplexes). Different memeplexes, each performs a local search, are considered as different cultures (memes) of frogs.

The SFLA is described as follows: assume that the initial population is formed by P randomly generated frogs. For L-dimensional problems (L variables), a frog i is represented as Xi=(xi1, xi2,., xiL). Afterwards, the frogs are sorted in a descending order according to their fitness. Then, the entire population is divided into m memeplexes, each containing n frogs (i.e.P = mun). In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog m goes to the mth memeplex, and frog m+1 goes back to the first memeplex, etc.

Within each memeplex, the frogs with the best and the worst fitness are denoted by Xb and Xw. The frog with the global best fitness is identified as Xg.

$$S = rand(\,) \times x_b - x_w$$

$$X_{new} = X_w + S\,, \; -S_{max} \le S \le S_{max}$$

Where rand() is a random number between 0 and 1. S_{max} is the maximum change in a frog's position.

The steps to find the optimal solution are as follows:

I. Initialization: initialize population randomly.
II. Separate population into numMemeplex memeplexes.
III. Update worst candidate solution using best candidate solution on each memeplex as much as frogLeaping Iteration.
IV. Shuffled back each memeplexes into population.
V. Sort population based on fitness.
VI. If a termination criterion (a maximum number of iterations or a sufficiently good fitness) is met, exit the loop, else back to separate population into memeplexes.

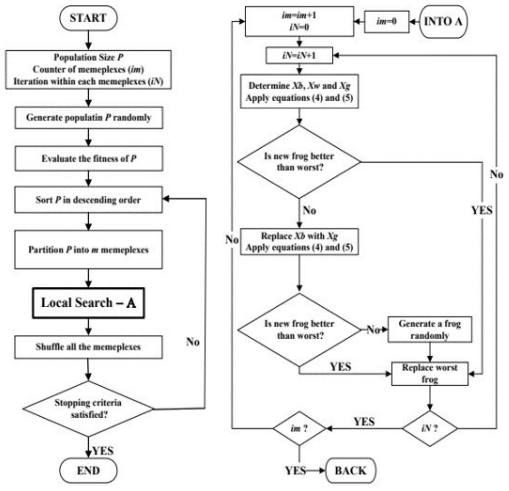


Figure 3: Flow diagram of the SFL algorithm

IV. Used libraries:

We used the "metaheuristicOpt" library, which contains the GA, PSO and SFL functions. This package contains an implementation of metaheuristic algorithms for continuous optimization.

In the following section we will describe how to implement each metaheuristic algorithm used in this paper with the "metaheuristicOpt" library in R.

GA function:

In order to use GA function, we only have to charge the metaheuristicOpt package.

Usage:

ga(FUN, optimType = "MIN", numVar, numPopulation = 40, maxIter = 500, rangeVar, Pm = 0.1, Pc = 0.8)

Arguments

FUN → objective function or cost function.

optimType → String value that represent the type of optimization, "MIN" for minimization and "MAX" for maximization.

numVar → Positive integer to determine the number variables.

numPopulation → Positive integer to determine the number populations. The default value is 40.

maxIter → Positive integer to determine the maximum number of iterations. The default value is 500.

rangeVar → Matrix (2 × n) containing the range of variables, where n is the number of variables, and first and second rows are the lower bound (minimum) and upper bound (maximum) values, respectively.

Pm → Positive integer to determine mutation probability. The default value is 0.1.

Pc → Positive integer to determine crossover probability. The default value is 0.8.

PSO function:

In order to use PSO function, we only have to charge the metaheuristicOpt package.

Usage:

PSO(FUN, optimType = "MIN", numVar, numPopulation = 40, maxIter = 500, rangeVar, Vmax = 2, ci = 1.49445, cg = 1.49445, w = 0.729)

Arguments:

FUN → Objective function

optimType → string representing the optimization type MIN or MAX. The default value is MIN

numVar → Positive integer to determine the number variables.

numPopulation → number of subjects, default is 40.

maxIter → number of iterations of the process

rangeVar → range of values of each variable

Vmax → maximum speed of a particle

ci → attraction constant to the personal best

cg → attraction constant to the best global

w → inertia weight constant

SFL function:

This is the internal function that implements Shuffled Frog Leaping Algorithm. It is used to

solve continuous optimization tasks. Users do not need to call it directly, but just use metaOpt.

Usage:
SFL(FUN, optimType = "MIN", numVar, numPopulation = 40, maxIter = 500, rangeVar, numMemeplex = as.integer(numPopulation/3), frogLeapingIteration = as.integer(10))

Arguments:

FUN → Objective function or cost function.

optimType → String value that represent the type of optimization. There are two option for his arguments MIN and MAX. Default value is MIN.

numVar → Positive integer to determine the number variables.

numPopulation → Positive integer to determine the number populations. The default value is 40.

maxIter → Positive integer to determine the maximum number of iterations. The default value is 500.

rangeVar → Matrix containing the range of variables.

numMemeplex → Positive integer between 0 and numVar to determine number of memeplexes.

frogLeapingIteration → Positive integer to determine number of iterations for each memeplex.

V. Source codes:

GA Source Code:

```
#install.packages("metaheuristicOpt")
#library(metaheuristicOpt)

#Fitness Function

FO <- function(xx)
{
 ii <- c(1:length(xx))
 sum <- sum(xx^2/4000)
 prod <- prod(cos(xx/sqrt(ii)))

 y <- sum - prod + 1
 return(y)
}
Pm <- 0.1
Pc <- 0.8
numVar <- 30
rangeVar <- matrix(c(-10, 10), nrow=2)
minimos<-c()

for(i in 1:30){
 resultGA <- GA(  FO, optimType="MIN",
numVar, numPopulation=40,
            maxIter=1000, rangeVar, Pm, Pc)
 ## calculate the optimum value using sphere
function
 optimum.value <- sphere(resultGA)
 minimos[i]<-optimum.value
 acum<- acum+i
}
## calculate the
mean(minimos)
sd(minimos)
```

PSO Source Code:

```
#install.packages("metaheuristicOpt")
library(metaheuristicOpt)


#Fitness Function
FO <- function(xx)
{
 ii <- c(1:length(xx))
```

```
 sum <- sum(xx^2/4000)
 prod <- prod(cos(xx/sqrt(ii)))

 y <- sum - prod + 1
 return(y)
}



#Parameter definition
Vmax <- 2
ci <- 1.5
cg <- 1.5
w <- 0.7
numVar <- 30
rangeVar <- matrix(c(-600,600), nrow=2)
minimos<-c()

for (i in 1:30) {
 # Optimum vector searched by PSO function
 resultPSO <- PSO(FO, optimType="MIN",
numVar, numPopulation=40,
              maxIter=1000, rangeVar, Vmax, ci, cg,
w)

 # Optimum value obtained evaluating the best
particle in the fitness function
 optimum.value <- FO(resultPSO)
 minimos[i] <- optimum.value
}

#Mean value of the 30 observations
mean(minimos)
#S.D value of the 30 observations
sd(minimos)
```

SFL Source Code:

```
#install.packages("metaheuristicOpt")
library(metaheuristicOpt)

# define objective function
rothyp <- function(x)
{
 d <- length(x)

 xmat <- matrix(rep(x,times=d), d, d,
byrow=TRUE)
 xmatlow <- xmat
 xmatlow[upper.tri(xmatlow)] <- 0

 inner <- rowSums(xmatlow^2)
 outer <- sum(inner)

 y <- outer
 return(y)
}

## Define parameter
numVar <- 30
rangeVar <- matrix(c(-100,100), nrow=2)
minimos<-c()

for (i in 1:30) {
 ## calculate the optimum solution shuffled frog
leaping algorithm
 resultSFL <- SFL(rothyp, optimType="MIN",
numVar, numPopulation=40,
              maxIter=1000, rangeVar)

 ## calculate the optimum value using rotated
function
 optimum.value <- rothyp(resultSFL)
 minimos[i] <- optimum.value
}

#Mean value of the 30 observations
mean(minimos)
#S.D value of the 30 observations
sd(minimos)
```

VI. Fitness functions' codes: (FO)

In this section we included the specifications each objective function has (length, search space and its minimum value).

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f1$	Sphere	$f(x)=\sum_{i=1}^{d} x_i^2$	30	$x_i \epsilon [-100,100]$	$f(x)=0$
$f2$	Schwefel 2.22	$f(x)=\sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	30	$x_i \epsilon [-10,10]$	$f(x)=0$
$f3$	Rotated Hyperellipsoid	$f(x)=\sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j\right)^2$	30	$x_i \epsilon [-100,100]$	$f(x)=0$
$f4$	Schwefel 2.21	$f(x)=\max_{i=1,\ldots,d} \lvert x_i \rvert$	30	$x_i \epsilon [-10,10]$	$f(x)=0$
$f5$	Rosenbrock	$f(x)=\sum_{i=1}^{d} \left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	30	$x_i \epsilon [-30,30]$	$f(x)=0$
$f6$	Step 2	$f(x)=\sum_{i=1}^{d} [x_i+0.5]^2$	30	$x_i \epsilon [-100,100]$	∞ and $f(x)=0$
$f7$	Quartic with noise	$f(x)=\sum_{i=1}^{d} ix_i^2+random[0,1]$	30	$x_i \epsilon [-1.28,1.28]$	$f(x)=0+random[0,1]$

f	Function	Mathematical Equation	Dimensions	Search space	Minimum
$f8$	Schwefel	$f(x)=418.9829d-\sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	$x_i \epsilon [-500,500]$	$f(x)=0$
$f9$	Rastrigin	$f(x)=\sum_{i=1}^{d} \left[x_i^2-10\cos(2\pi x_i)+10\right]$	30	$x_i \epsilon [-5.12,5.12]$	$f(x)=0$
$f10$	Ackley 1	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right)+20+\exp(1)$	30	$x_i \epsilon [-32,32]$	$f(x)=0$
$f11$	Griewank	$f(x)=\sum_{i=1}^{d} \frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$x_i \epsilon [-600,600]$	$f(x)=0$
$f12$	Generalized Penalized 1	$f(x)=\frac{\pi}{d}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{d-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{d} u(x_i,a,k,m)$, where $y_i=1+\frac{1}{4}(x_i+1)$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k(-x_i-a)^m & x_i<-a\end{cases}$, $a=10,k=100,m=4$	30	$x_i \epsilon [-50,50]$	$f(x)=0$
$f13$	Generalized Penalized 2	$f(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{d-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_d-1)^2\left[1+\sin^2(2\pi x_d)\right]\right\}+\sum_{i=1}^{d} u(x_i,a,k,m)$, where $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k(-x_i-a)^m & x_i<-a\end{cases}$, $a=5,k=100,m=4$	30	$x_i \epsilon [-50,50]$	$f(x)=0$

In this section we included the source code used for every fitness function that helped us test the effectiveness of each metaheuristic algorithm used in this paper (GA, PSO and SFL).

Sphere (*f1*):

```
sphere <- function(x){ return(sum(x^2)) }
```

Schwefel 2.22 (*f2*):

```
schewefels2.22 <-
function(x){return(sum(abs(x)+prod(abs(x))))}
```

Rotated (*f3*):

```
rothyp <- function(x)
{
        d <- length(x)

        xmat <- matrix(rep(x,times=d),
d, d, byrow=TRUE)
        xmatlow <- xmat
        xmatlow[upper.tri(xmatlow)] <-
0

        inner <- rowSums(xmatlow^2)
        outer <- sum(inner)

        y <- outer
        return(y)
}
```

Schwefel 2.21 (*f4*):

```
schewefels2.21 <- function(x)
{
 maximum <- 0
 for(i in x){
  if(abs(i)>maximum) {
   maximum <- abs(i)
  }
 }
 return(maximum)
}
```

Rosenbrock (*f5*):

```
rosen <- function(xx)
{
 d <- length(xx)
 xi <- xx[1:(d-1)]
 xnext <- xx[2:d]

 sum <- sum(100*(xnext-xi^2)^2 +
(xi-1)^2)

 y <- sum
 return(y)
}
```

Step2 (*f6*):

```
step2 <- function(x){
        acum<-0
        for(i in x){
        acum <- acum + (i+0.5)^2
        }
        return(acum)
         }
```

Quartic with noise (*f7*):

```
quartic <- function(x){
 dim <- length(x)
 result <- sum(c(1:dim)*(x^4))+runif(1)
 return(result)
}
```

Schwefel (*f8*):

```
schwef <- function(xx)
{
 d <- length(xx)

 sum <- sum(xx*sin(sqrt(abs(xx))))

 y <- 418.9829*d - sum
 return(y)
}
```

Rastrigin (*f9*):

```
rastrigin <- function(x)
10*length(x)+sum(x^2-10*cos(2*pi*x))
```

Ackley 1 (*f10*):

```
ackley1 <- function(xx, a=20, b=0.2, c=2*pi)
{
 d <- length(xx)

 sum1 <- sum(xx^2)
 sum2 <- sum(cos(c*xx))

 term1 <- -a * exp(-b*sqrt(sum1/d))
 term2 <- -exp(sum2/d)

 y <- term1 + term2 + a + exp(1)
 return(y)
}
```

Griewank (*f11*):

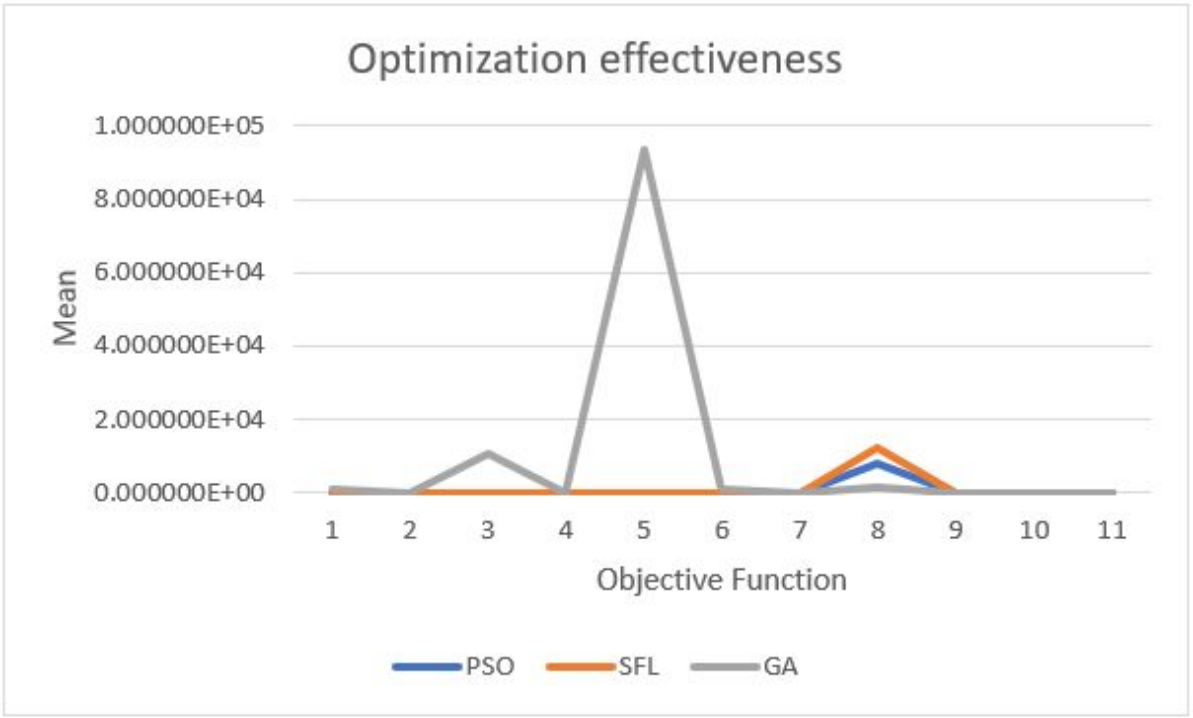
```
griewank <- function(xx)
{
  ii <- c(1:length(xx))
 sum <- sum(xx^2/4000)
 prod <- prod(cos(xx/sqrt(ii)))

 y <- sum - prod + 1
 return(y)
}
```

VII. Results:

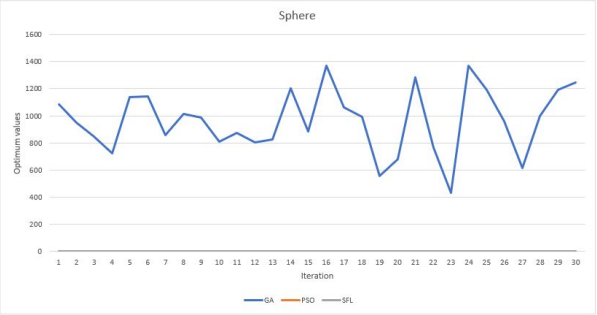
The following table show the optimization results of each function by the three algorithms:

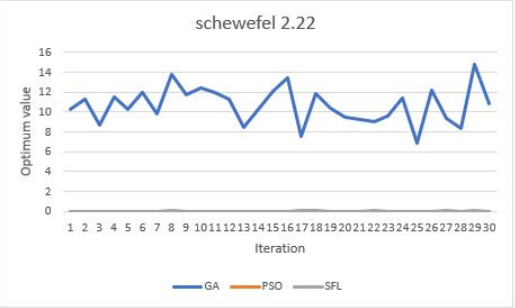
				PSO		SFL		GA	
	DIMENSION	Search Space		*PROMEDIO*	*D.E*	*PROMEDIO*	*D.E*	*PROMEDIO*	*D.E*
Sphere	30	-100,100	***f1***	1.991166E-25	5.13191E-25	3.58E-06	9.83E-06	*9.58E+02*	*2.35E+02*
Schwefel 2.22	30	-10,10	***f2***	1.981416E-12	3.46426E-12	0.007442375	0.0152966	*1.07E+01*	*1.84E+00*
Rotated	30	-100,100	***f3***	7.308752E-23	3.16693E-22	0.1147619	0.3508896	*1.09E+04*	*2.78E+03*
Schwefel 2.21	30	-10,10	***f4***	1.083170E-05	1.04678E-05	2.071518	0.4690952	*2.17E+00*	*2.99E-01*
Rosenbrock	30	-30,30	***f5***	17.15886	11.91581	123.7789	77.87103	*9.34E+04*	*3.62E+04*
Step 2	30	-100,100	***f6***	1.765408E-25	2.54156E-25	1.48E-06	1.76E-06	*9.37E+02*	*2.37E+02*
Quartic	30	-1.28-1.28	***f7***	5.9754530000	7.363943000	0.4867878	0.2932479	*7.18E-01*	*2.85E-01*
Schwefel	30	-500,500	***f8***	7809.003	1812.833	12483.8	1.773126	*1.28E+03*	*2.53E+02*
Rastrigin	30	-5.12,5.12	***f9***	58.17891	20.25393	23.61295	6.517665	*5.07E+01*	*7.58E+00*
Ackley 1	30	-32,32	***f10***	0.1378112	0.3633634	1.250976	0.7159771	*7.88E+00*	*5.65E-01*
Griewank	30	-600,600	***f11***	0.0312028	0.02902455	0.04382675	0.03682335	*9.37E+00*	*1.73E+00*

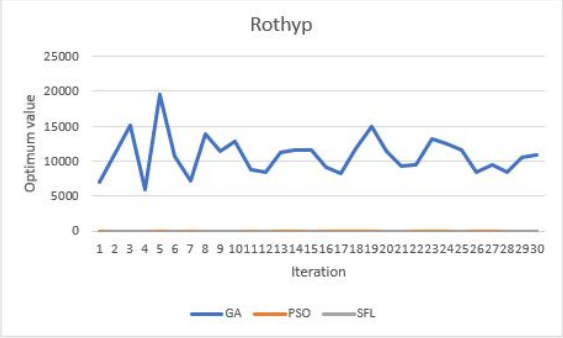
The following graph show the visual comparison of the values obtained by the three algorithms in each function:

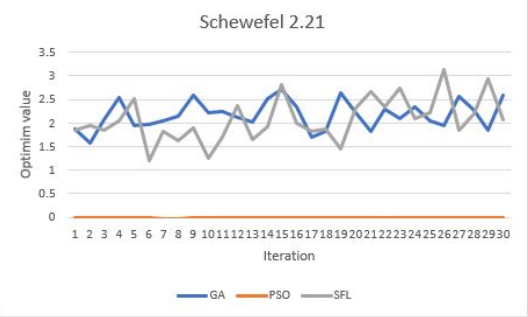


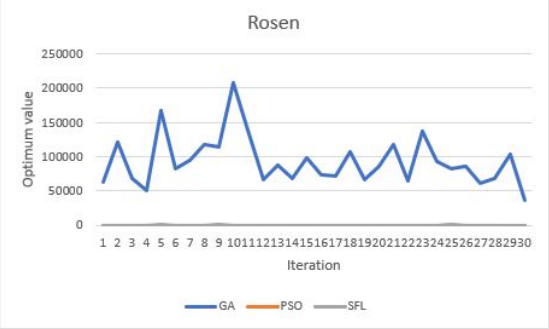
The following graphs show the comparison of the three algorithms by function:

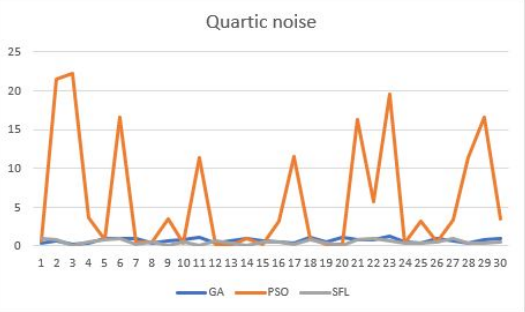


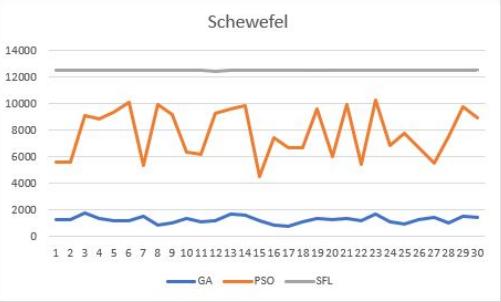


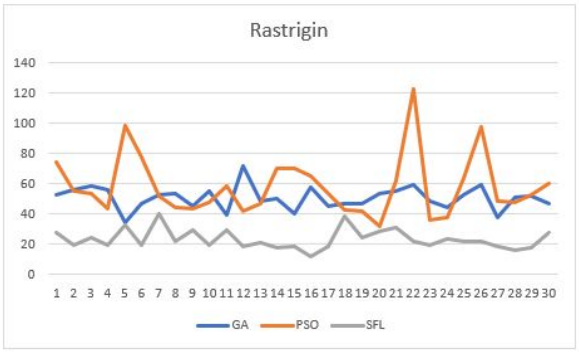


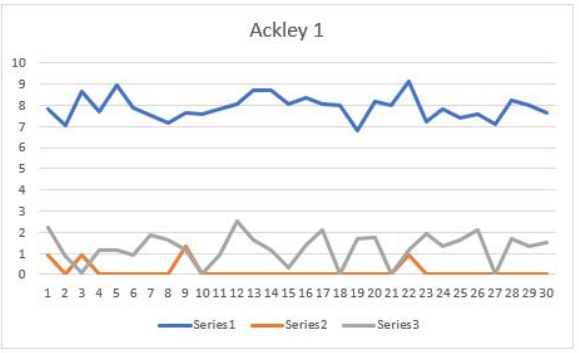












As we can notice the algorithm that better works on minimization problems is PSO. Out of 11 functions tested, this algorithm created the lowest value in eight functions. Another thing we can notice is that the SFL algorithm was better than the PSO in multimodal functions, also the GA algorithm was the worst of the three algorithms on minimization problems, only on Schwefel (***f8)*** works better than the others.

As we can see in the graphs, the values of the GA algorithm are bigger compared to the other two algorithms, which is why in the graphs the result of the SFL and PSO algorithms are practically not seen. The same problem we have on the Quartic Noise (***f7)*,** but this time the algorithm that has the worst results is the PSO.

VIII. Conclusions:

Metaheuristic algorithms allow us to find the optimal solution we are looking for, through an intelligent search that ensures the exploration of the entire solution space. This is achieved through iterations of the process where each iteration will improve the values thanks to always keeping in mind the latest best solution found. At the same time, these algorithms must be based on diversification to avoid stagnation in some solution, which may or may not be the optimum solution.

After analysing the algorithms performance, we come to the conclusion that PSO had the best benchmarks. GA is the worst algorithm, being overcome in 10 of the 11 functions tested. This means that GA function is bad implemented when we work with more variables number and larger ranges. In functions like Rastrigin, Schwefel or Quartic we have a better solution in GA because we use less range to evaluate the function.

Population-based algorithms are being used to reconstruct microwave images. Where the global optimization problem is the imaging problem, each subject has its own fitness values for the defined objective function. The algorithm will reconstruct the image by searching for the optimal solution. PSO is our best option to solve the microwave imaging construction problem. It will reconstruct in a good percentage the image of the microwave.

XI. References:

Anita Thengade, Rucha Dondal. (2012, 7-8 april). Genetic Algorithm-Survey Paper. https://www.researchgate.net/publication/268010749_Genetic_Algorithm_-_Survey_Paper

Cai, W. Song, C. Zhang, H. Zhao, H. (2007) The Limited Mutation Particle Swarm Optimizer. Recuperado el 3 de mayo de 2020 de https://link.springer.com/chapter/10.1007/978-3-540-74769-7_29

Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Engineering Optimization, 38(2), 129–154.

Farahat, M. Taalat, M. (2019) Moth-Flame algorithm for accurate simulation of a non-uniform electric field in the presence of dielectric barrier. Recuperado el 30 de abril de 2020 de https://www.researchgate.net/publication/329897481_Moth-Flame_algorithm_for_accurate_simulation_of_a_non-uniform_electric_field_in_the_presence_of_dielectric_barrier

Huang, T. (2007) A Microparticle Swarm Optimizer for the Reconstruction of Microwave Imageshttps://www.researchgate.net/publication/3019046_A_Microparticle_Swarm_Optimizer_for_the_Reconstruction_of_Microwave_Images

ZhaoLiping, WangWeiwei, Han Yi, XuYefeng, ChenYixian . (2012, enero 1). Application of Shuffled Frog Leaping Algorithm to an Uncapacitated SLLS Problem. Recuperado 30 de abril de 2020, de https://www.sciencedirect.com/science/article/pii/S2212671612000364