Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Presenta:

Ing. Jesús Enríquez Gaytán.

Asesores:

Dr. Felipe Gómez Castañeda.
Dr. José Antonio Moreno Cadenas.

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

- Contenido
  i. Objetivo.
  ii. Antecedentes.
  iii. Aplicación en software.
  iv. Aplicación en Hardware.
  v. Conclusiones.
  vi. Trabajo Futuro.

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Objetivo.

El objetico principal de este trabajo es, por una parte, el estudiar algunas técnicas meta-heurísticas en optimización de funciones, para obtener las mascarillas de una red neuronal celular (CNN) en procesamiento de imágenes; mientras que por otra parte, la implementación de dichas técnicas para trabajos de segmentación de imágenes y aproximadores de función de señales en un dispositivo FPGA.

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Antecedentes.

- Algoritmo de Optimización.
- Algoritmos Meta-heurísticos de optimización.
- Arquitectura de la red neuronal celular – CNN.
  - Diseño de mascarillas para procesamiento de imágenes.
- Dispositivo FPGA.

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Algoritmo de Optimización.

Problema de optimización numérica.

función objetivo f(x).

$x = [x1, x2, \ldots , xn]$ $x \in R^n$; Vector x de n variables.

Sujeta a:

$g(x) \leq 0$, i=1,...m # restricciones de desigualdades.

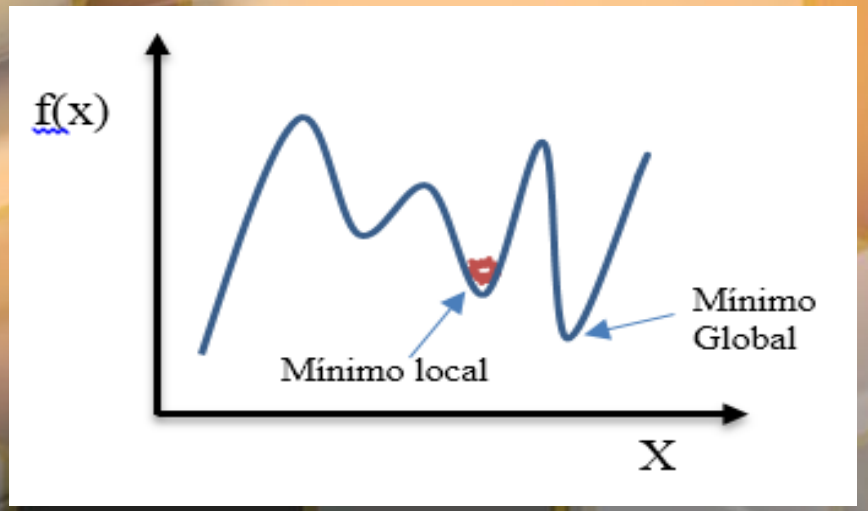
$h(x) \leq 0$, j=1,...p # restricciones de igualdad.

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

- Algoritmo Meta-heurísticos de Optimización.
- Inteligencia colectiva (Swarm Intelligence).
  - Algoritmo de nube de partículas (PSO – Particle Swarm Optimization).
  - Algoritmo de colonia artificial de abejas (ABC – Artificial Bee Colony).



Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Algoritmo Meta-heurísticos de Optimización.

- Algoritmo de colonia artificial de abejas (ABC – Artificial Bee Colony).
  - Dervis Karaboga en el 2005.
  - Tipos de Abejas Obreras, Observadoras y exploradoras.
  - Algoritmo iterativo.

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Enviar abejas exploradoras. $X_i^j = X_{min}^j + rand(0,1)(X_{max}^j - X_{max}^j)$

Repetir

- Enviar las abejas empleadas y calcular el fitness. $V_i^j = X_i^j + rand(0,1)(X_i^j - X_k^j)$
- Calcular la probabilidad de cada solución de ser explotada.
- Explotar soluciones por medio de las abejas observadoras.
- Determinar exploración de la soluciones (abandonar soluciones).
- Enviar abejas exploradoras a explorar.
- Almacenar la mejor solución.

$$\text{fitness}(x_p) = \begin{cases} \frac{1}{1+f(x_p)} & \text{if } f(x_p) \geq 0 \\ 1+\text{abs}(f(x_p)) & \text{if } f(x_p) < 0 \end{cases}$$

$$P_p = \frac{\text{fitness}(x_p)}{\sum_{p=1}^{np} \text{fitness}(x_p)}$$

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Algoritmo Meta-heurísticos de Optimización.

- Algoritmo de nube de partículas (PSO – Particle Swarm Optimization).
  - Desarrollado psic. Jammes Kennedy y por el ing. Russell Eberhart en 1995.
  - Algoritmo iterativo.

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Generar aleatoriamente una nube inicial de soluciones.

- Calcular la aptitud de la nube inicial.

Repetir

- Seleccionar al líder (o lideres del cúmulo).
- Para cada partícula actualizar la posición (vuelo).
  - $V_i^{t+1} = w^t V_i^t + C_1 r_1^t \left(P_i - X_i^t\right) + C_2 r_2^t (G_i - X_i^t)$
  - $X_i^{t+1} = X_i^t + V_i^{t+1}$
- Evaluar cada partícula
- Actualizar el P_i de cada partícula

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Arquitectura de la red neuronal celular – CNN.

$$\frac{dv_{xij}(t)}{dt} = -v_{xij}(t) + \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

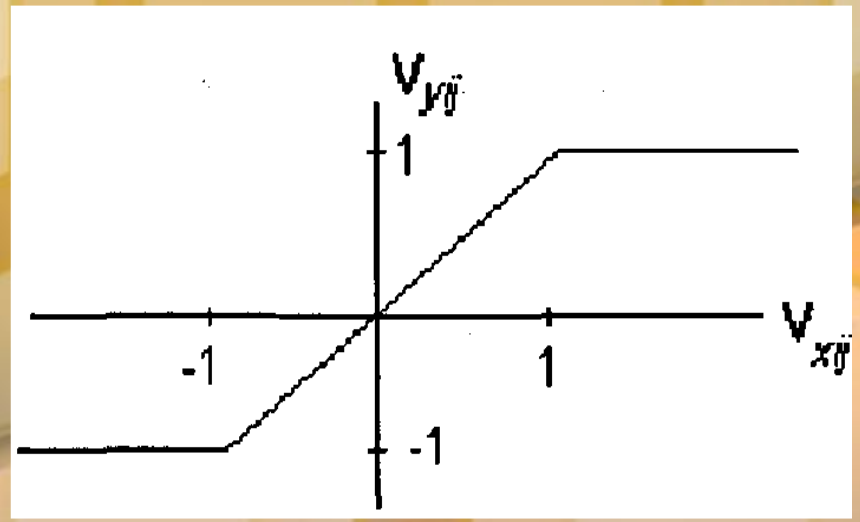
$$1 \le i \le M; 1 \le j \le N.$$

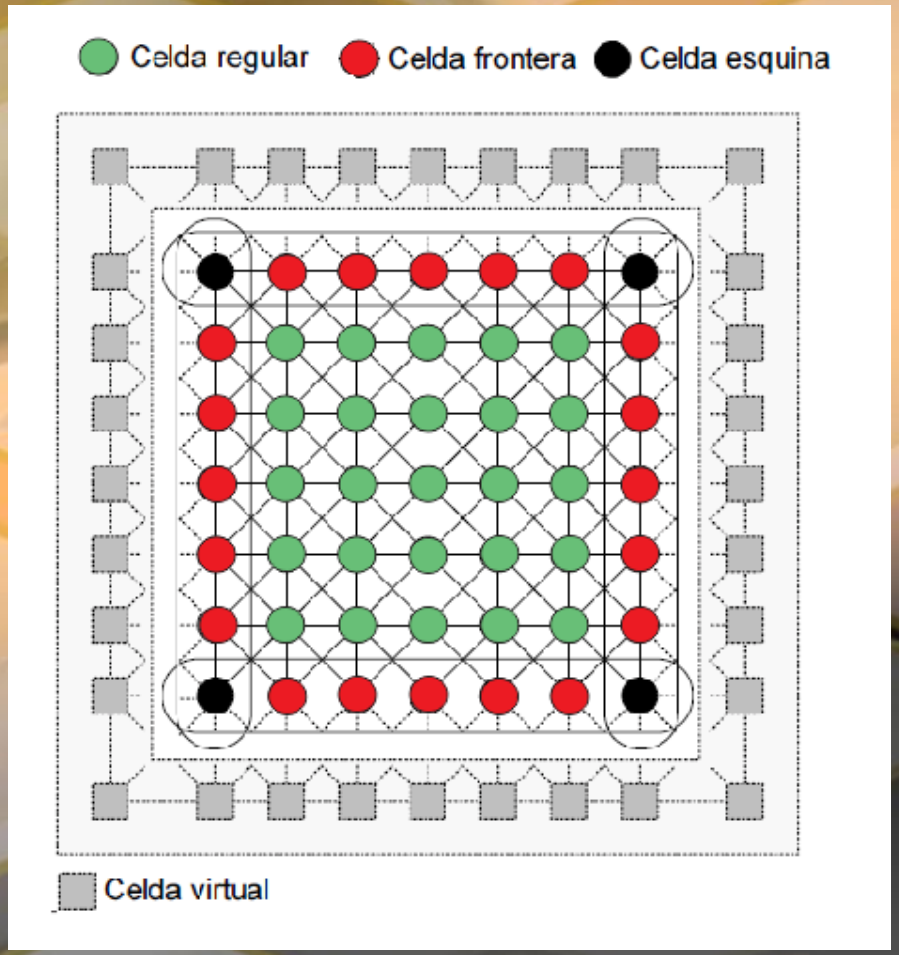
$$A = \begin{bmatrix} a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} \\ a_{i,j-1} & a_{i,j} & a_{i,j+1} \\ a_{i+1,j-1} & a_{i+1,j} & a_{i+1,j+1} \end{bmatrix}, \mathrm{B} = \begin{bmatrix} b_{i-1,j-1} & b_{i-1,j} & b_{i-1,j+1} \\ b_{i,j-1} & b_{i,j} & b_{i,j+1} \\ b_{i+1,j-1} & b_{i+1,j} & b_{i+1,j+1} \end{bmatrix} \text{ y } I_{ij}$$

$$v_{yij} = f\big(x_{ij}(t)\big) = \frac{1}{2}\left|x_{ij}(t)+1\right| - \frac{1}{2}\left|x_{ij}(t)-1\right| \qquad \begin{cases} 1, si\ V_{yIJ} \ge 1 \\ V_{yIJ}, si -1 < V_{yIJ} < 1 \\ -1, si\ V_{yIJ} \le -1 \end{cases}$$

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular





Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	1	-1	1	-1	-1	1	1	-1	1	1	1	-1
-1	1	-1	1	-1	1	-1	-1	1	1	-1	1	1	1	-1
-1	1	-1	1	-1	1	-1	-1	1	1	-1	1	1	1	-1
-1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	-1	1	-1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	-1	1	-1	-1	1	-1	-1
-1	1	-1	1	-1	1	-1	-1	1	1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	1	1	-1	1	1	-1	1	1	1	-1
-1	-1	1	-1	-1	1	1	-1	1	1	-1	1	1	1	-1
-1	-1	1	-1	-1	1	1	-1	1	1	-1	1	1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Imagen de 18 x 15.

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Diseño de mascarillas de la red neuronal celular.

Fue propuesto por K. Nakai y A. Ushida, el principio del diseño se basa en forzar la salida de cada neuronas en respuesta a un patrón de entrada considerado solamente el estado inicial y el estado estable deseado para la celda.

El estado en equilibrio de la ecuación de estado.

$$v_{xij} = \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

Son estados estacionarios

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Diseño de mascarillas de la red neuronal celular.

Considerando un patrón de entrada V_{uij} y una salida deseada V_{yij}, las siguientes relaciones deben ser satisfechas como desigualdades.

A) V_{yij}=1, debe cumplir $V_{xij} \geq 1$.

$$1 \leq \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

B) V_{yij}=-1, debe cumplir $V_{xij} \leq -1$.

$$1 \geq \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

C) $V_{yij}(\infty)$=1, debe cumplir $V_{xij} < 1$.

$$0 < \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

D) $V_{yij}(\infty)$=-1, debe cumplir $V_{xij} > -1$.

$$0 > \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Diseño de mascarillas de la red neuronal celular.

Función objetivo, será aquella que mas se aproximé al centro del área solución de las desigualdades.

$$F(T) = \sum f_k(T)^2$$

Donde

$$f_k(T) = \begin{cases} f_k(T) - \emptyset & f_k(T) - \emptyset \geq 0 \\ p + m * (f_k(T) - \emptyset) & f_k(T) - \emptyset < 0 \end{cases}$$

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Diseño de mascarillas de la red neuronal celular.

Ejemplo: Función objetivo aplicando ABC.

- Partiendo de las restricciones: $x+y-2<0$, $x-y>0$, $x>0$ y $y>0$ y siguiendo los pasos para la función objetivo con un $\Phi = 1$, es posible obtener la siguiente función objetivo:

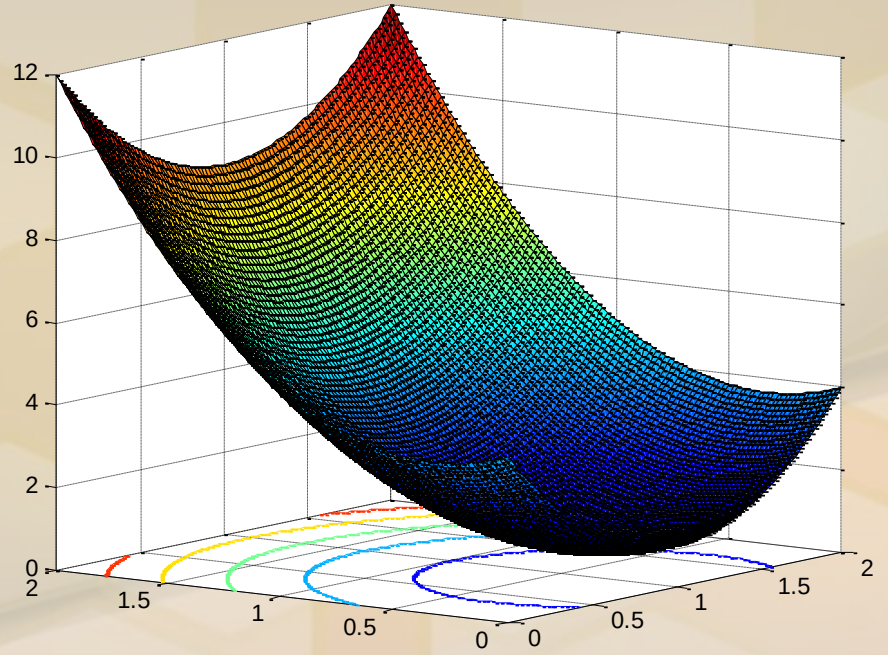
$$F(\mathbf{T}) = [f(x - y - 1)]\wedge 2 + [f(- x - y + 2 - 1)] \wedge 2 + [f(x - 1)] \wedge 2 + [f(y - 1)]\wedge 2$$

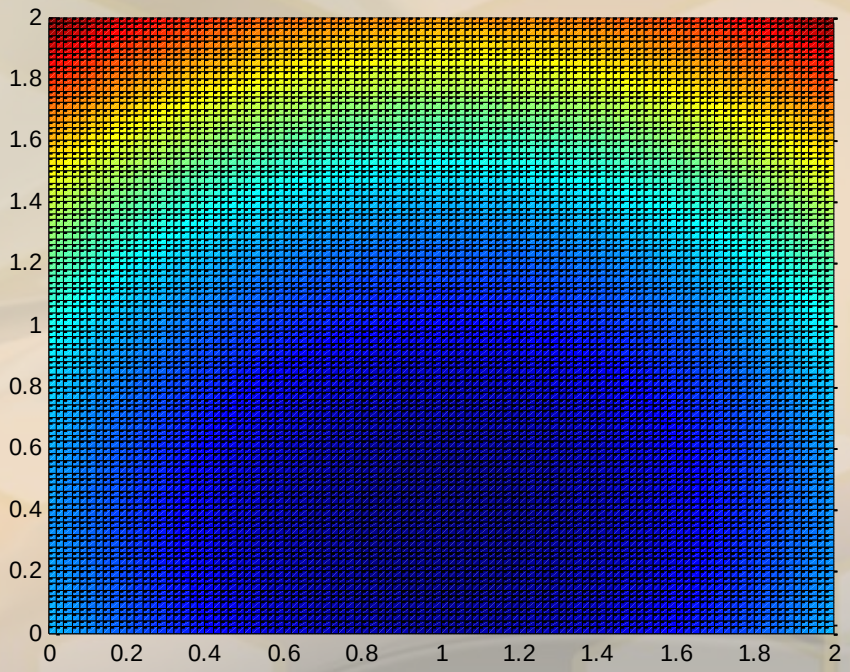
Mínimo de la función = 0.666.

X =1 y Y=0.3393

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular



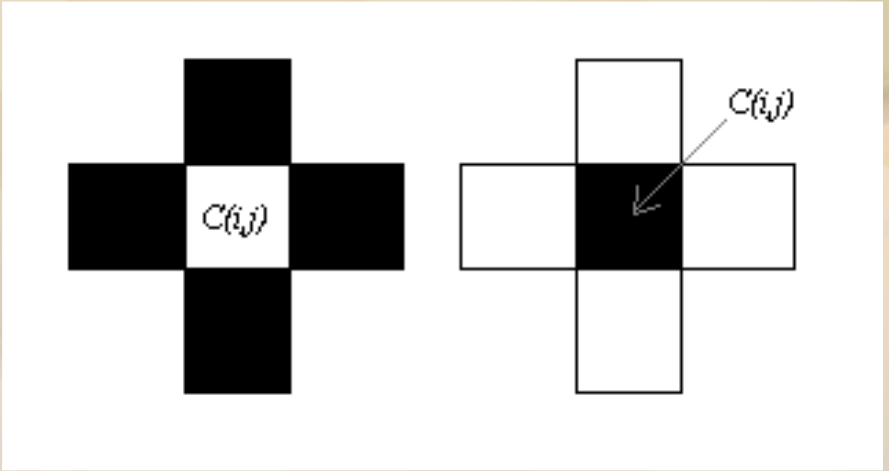


Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Diseño de mascarillas de la red neuronal celular.

Removedor de ruido



$$A = \begin{bmatrix} 0 & a & 0 \\ a & a0 & a \\ 0 & a & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, I$$

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Diseño de mascarillas de la red neuronal celular.

Ejeplo.-

Suponiendo que

Pixel de entrada $V_{uij}\ es\ Blanco\ (-1)$ pero queremos que e Pixel de Salida V_{yij} sea negreo(+1); Entonces se cumplen las condiciones de desigualdades A y C.

A) Para un valor V_{yij}=1 debe cuplir $V_{xij} \geq 1$.

$$1 \leq \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

C) $V_{yij}(\infty)$=1, debe cumplir $V_{xij} < 1$.

$$0 < \sum_{C(k,l)\in Nr(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l)\in Nr(i,j)} B(i,j;k,l)v_{ukl} + I_{ij}$$

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

$$A = \begin{bmatrix} 0 & a & 0 \\ a & a0 & a \\ 0 & a & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, I$$

A) $0 < 1 + ao(-1) + a(1) + a(1) + a(1) + a(1) + bo(-1) + I$

$-1 < -ao + 4 * a - bo + I$

C) $1 \leq ao(1) + a(1) + a(1) + a(1) + a(1) + bo(-1) + I$

$1 \leq ao + 4 * a - bo + I$

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

CASO	Píxel de Entrada v_{uij}	Píxel de Salida v_{yij}	Neuronas del vecindario C_{kl} $k \neq i, l \neq j$		Desigualdades
			No. de Negros	No. de Blancos	
(1)	Negro (+1)	Blanco(-1)	0	4	-ao - 4a + bo+ I ≤ -1 B)
					ao - 4a + bo + I < 1 D)
			0	3	-ao - 3a + bo + I ≤ -1 B)
					ao - 3a + bo + I < 1 D)
(2)	Negro (+1)	Negro (+1)	1	3	ao – 2a + bo + I ≥ 1 A)
			1	2	ao – a + bo + I ≥ 1 A)
			2	2	ao + bo + I ≥ 1 A)
			2	1	ao + a + bo + I ≥ 1 A)
			3	1	ao + 2a + bo + I ≥ 1 A)
			3	0	ao + 3a + bo + I ≥ 1 A)
			4	0	ao + 4a + bo + I ≥ 1 A)
(3)	Blanco(-1)	Blanco(-1)	0	4	-ao – 4a - bo + I ≤ -1 B)
			0	3	-ao – 3a - bo + I ≤ -1 B)
			1	3	-ao – 2a - bo + I ≤ -1 B)
			1	2	-ao – a - bo + I ≤ -1 B)
			2	2	-ao - bo + I ≤ -1 B)
			2	1	-ao + a - bo + I ≤ -1 B)
			3	1	-ao +2a - bo + I ≤ -1 B)
(4)	Blanco(-1)	Negro (+1)	3	0	ao + 3a - bo + I ≥ 1 A)
					-ao + 3a - bo + I > -1 C)
			4	0	ao + 4a - bo + I ≥ 1 A)
					-ao + 4a - bo + I > -1 C)
Condición para Estabilidad					ao > 1

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

A0=2.1032
B0= 3.9921
A=2.0381
I= -0.017462

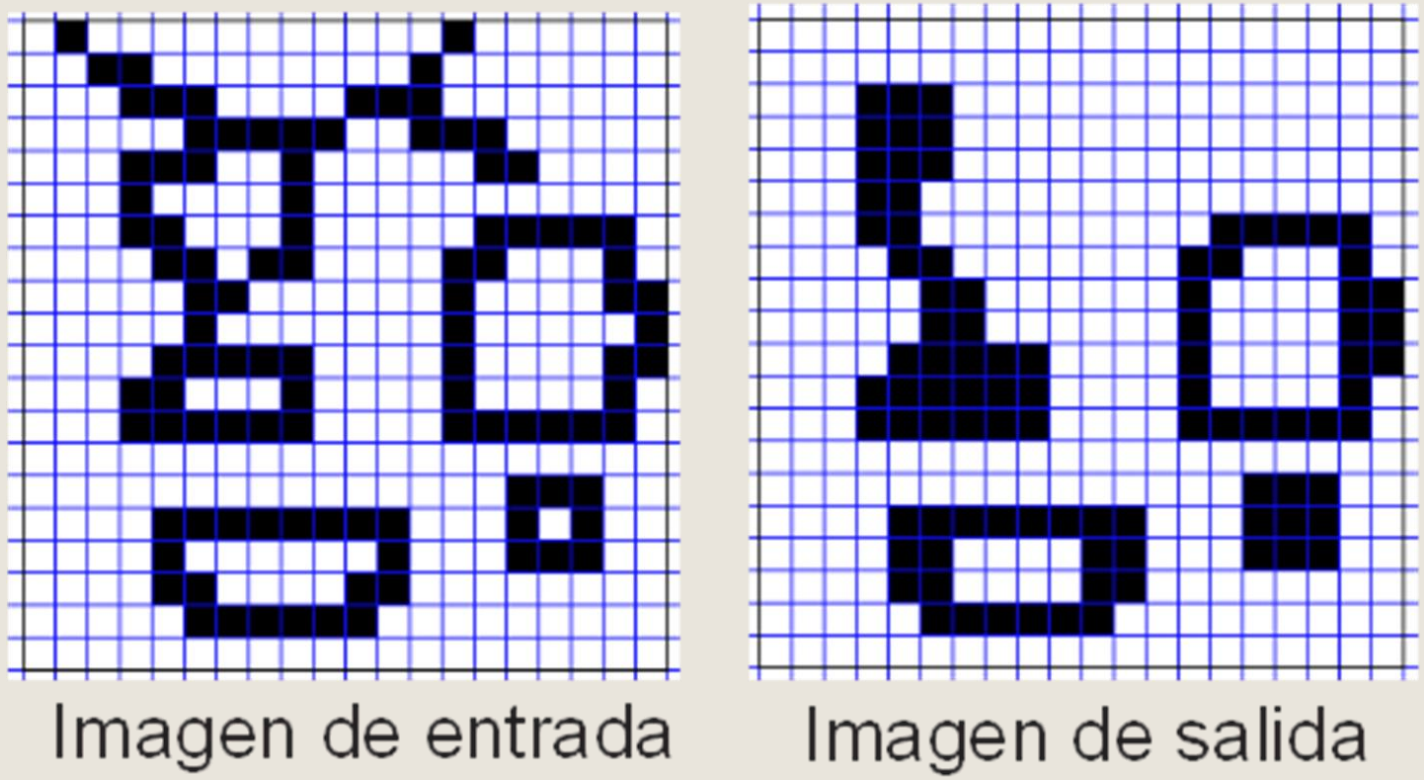


Imagen de entrada

Imagen de salida

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Aplicación de mascarillas de la red neuronal celular.

Detector de bordes.

$$A = \begin{bmatrix} 0 & -2 & 0 \\ -2 & 4.05 & -2 \\ 0 & -2 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4.931 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{I=-1}$$

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Aplicación de mascarillas de la red neuronal celular.

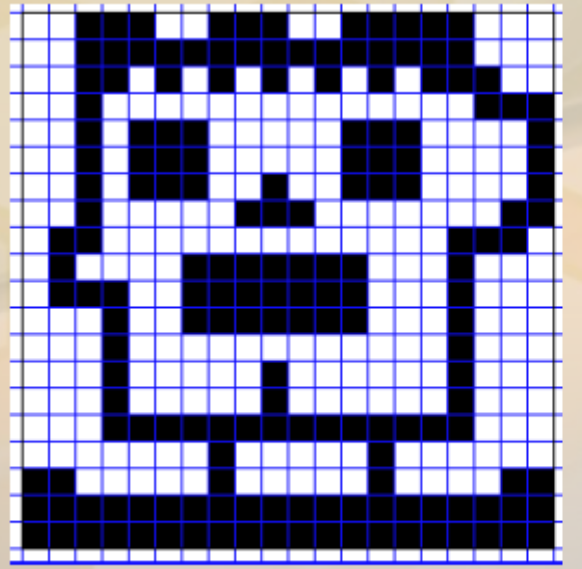


Imagen de entrada

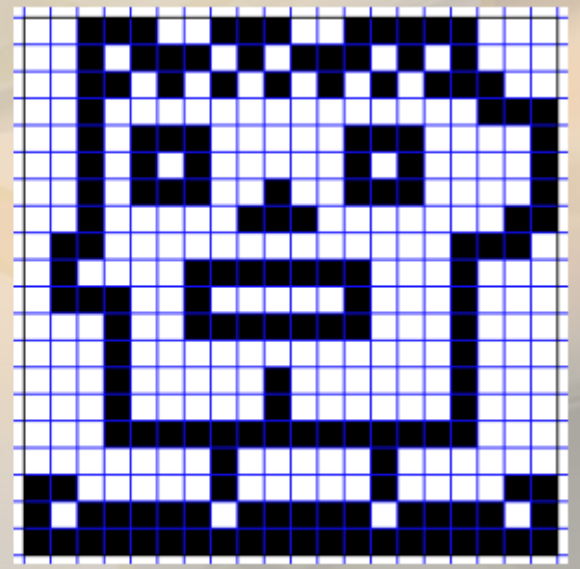


Imagen de salida

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Aplicación de mascarillas de la red neuronal celular.

Rellenado de huecos.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1.5 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad I=.5$$

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Aplicación de mascarillas de la red neuronal celular.

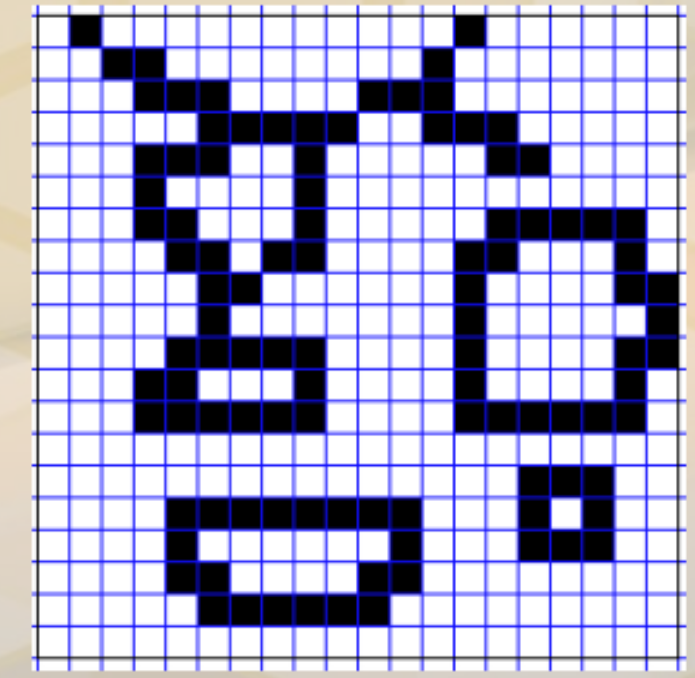


Imagen Original entrada

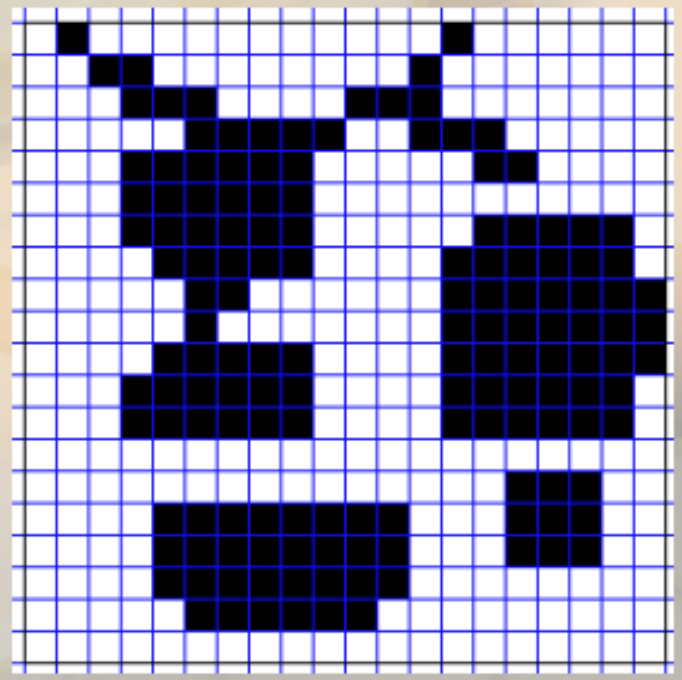


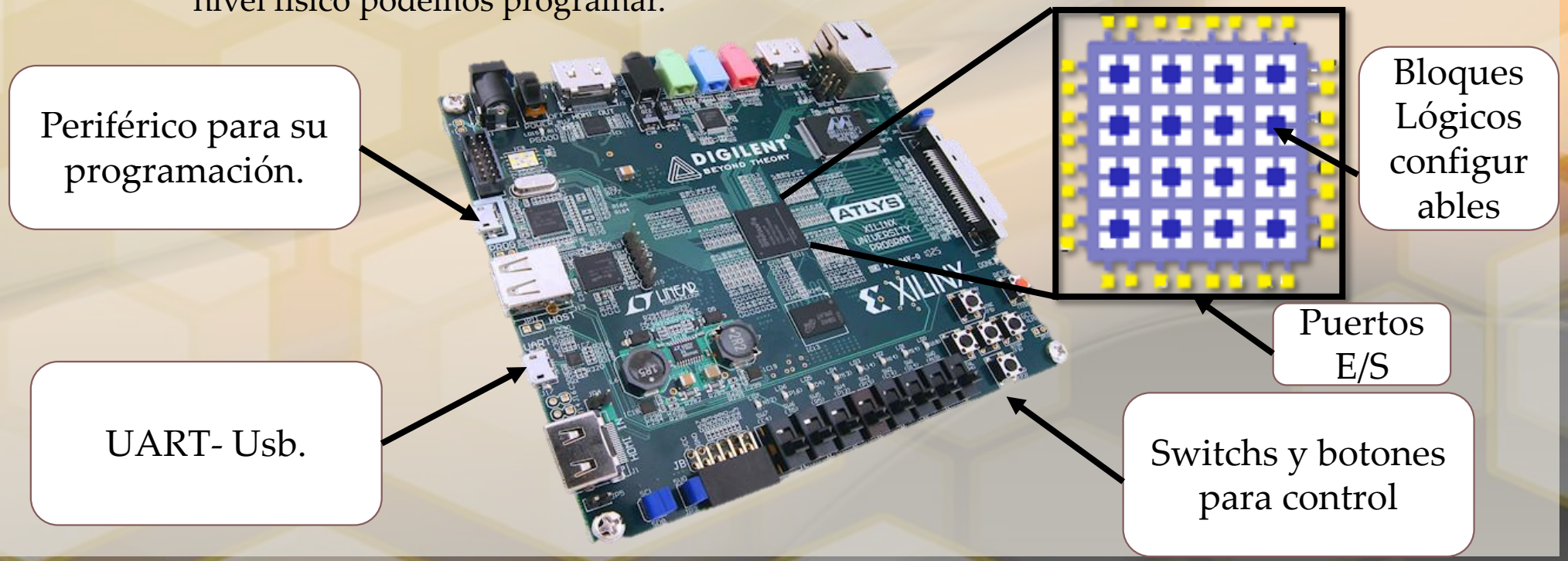
Imagen de Salida

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

I. Antecedentes.

Dispositivo FPGA.

Un FPGA es un dispositivo lógico programable, es decir un chip cuyas puertas lógicas a nivel físico podemos programar.



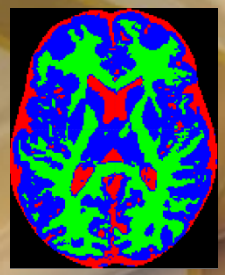
Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Cinvestav

III. Aplicación en software.

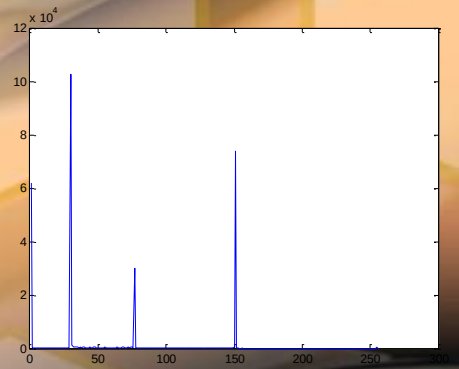
i. Segmentación de imágenes por PSO.

Lectura de la imagen.

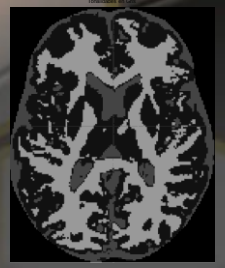


Conversión a escalad de grises.

- Generación de histograma de la imagen.



Segmentación mediante PSO.



Binarizacion de la imagen

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

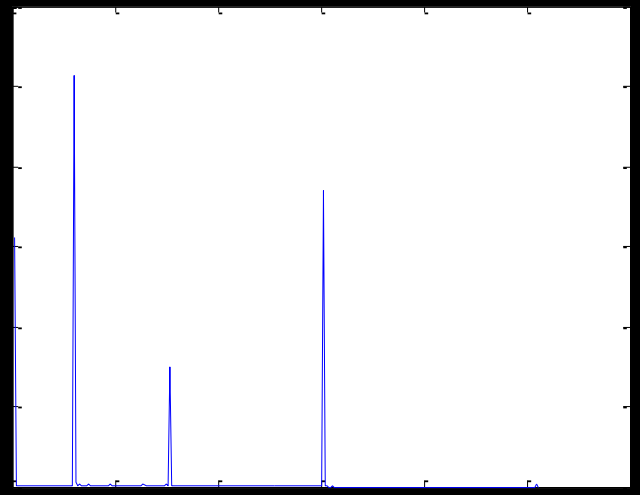
$$F(b) = \frac{1}{n}\sum_{i=1}^{n}\delta(I_i - b)$$

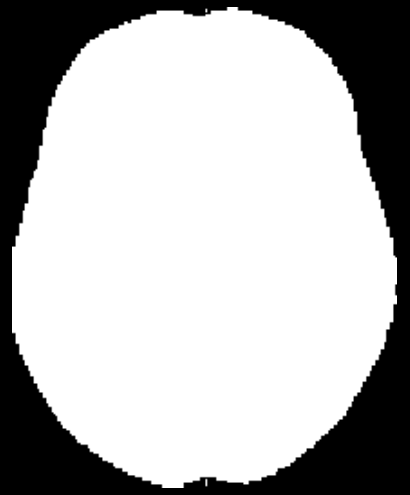
$$\text{with}: \delta(k) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

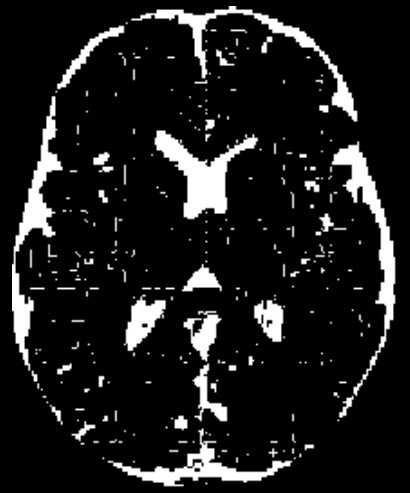
$$\text{Y\_bin} = \begin{cases} 1 & \text{if } \quad \text{Th} > \text{Y} \\ 0 & \text{otherwise} \end{cases}$$

```
Update algorithm
for k=1:1:nombre of iterations
 for i=1:1:M
    if pbest(i)<fitness(i)
       pbest(i)=fitness(i);
       pbest(i-1)=X(i);
    end
  TH=max(fitness);
    if TH>=gbest
       gbest=TH;
    v(i)=round(w(k)*V(i)+c1*rand*(pbest(i)-x(i))+c2*rand*(gbest-x(i)));
    x(i)=v(i)+x(i);
    end
 end
 Th(k)=gbest;
End
```

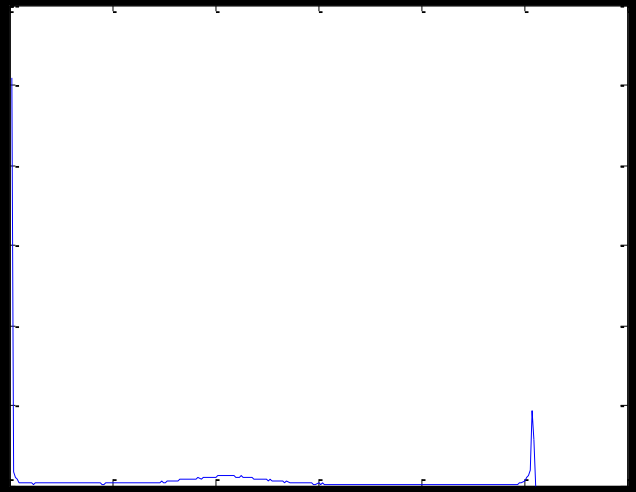
Cinvestav







Cinvestav



Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

III. Aplicación en software.

ii. Ejemplo numérico resuelto con ABC.

Ejemplo tomado de la literatura, en cual se evalúan 13 variables en una función objetivo, sujeta a 9 restricciones.

Minimizar la función $f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$

Sujeta a

$$
\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0 \\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0 \\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0 \\
g_4(x) &= -8x_1 + x_{10} \le 0 \\
g_5(x) &= -8x_2 + x_{11} \le 0 \\
g_6(x) &= -8x_3 + x_{12} \le 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \le 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \le 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \le 0
\end{aligned}
$$

$0 \le x_i \le 1 \quad (i = 1, \ldots, 9), \quad 0 \le x_i \le 100 \quad (i = 10,11,12)$

$0 \le x_{13} \le 1.$

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Numero de abejas 40, limite de ciclos antes de abandonar la fuente de alimento = 400

X=[0.99, 1, 1, 1, 1, 1, 1, 1, 1, 3.04, 2.96, 2.95, 1]

Mínimo global = [-14.944]

```
Ýter= 99993 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 99994 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 99995 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 99996 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 99997 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 99998 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 99999 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
Ýter= 100000 ObjVal= -14.9447 X10= 3.0438 X11= 2.9654 X12= 2.9591
```

x_1	1
x_2	1
x_3	1
x_4	1
x_5	1
x_6	1
x_7	1
x_8	1
x_9	1
x_{10}	3
x_{11}	3
x_{12}	3
x_{13}	1
g_1	0
g_2	0
g_3	0
g_4	−5
g_5	−5
g_6	−5
g_7	0
g_8	0
g_9	0
f	**−15**

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

III. Aplicación en software.

iii. Aproximado de función.

Análisis de componente Independientes (ICA).

Teniendo dos señales.

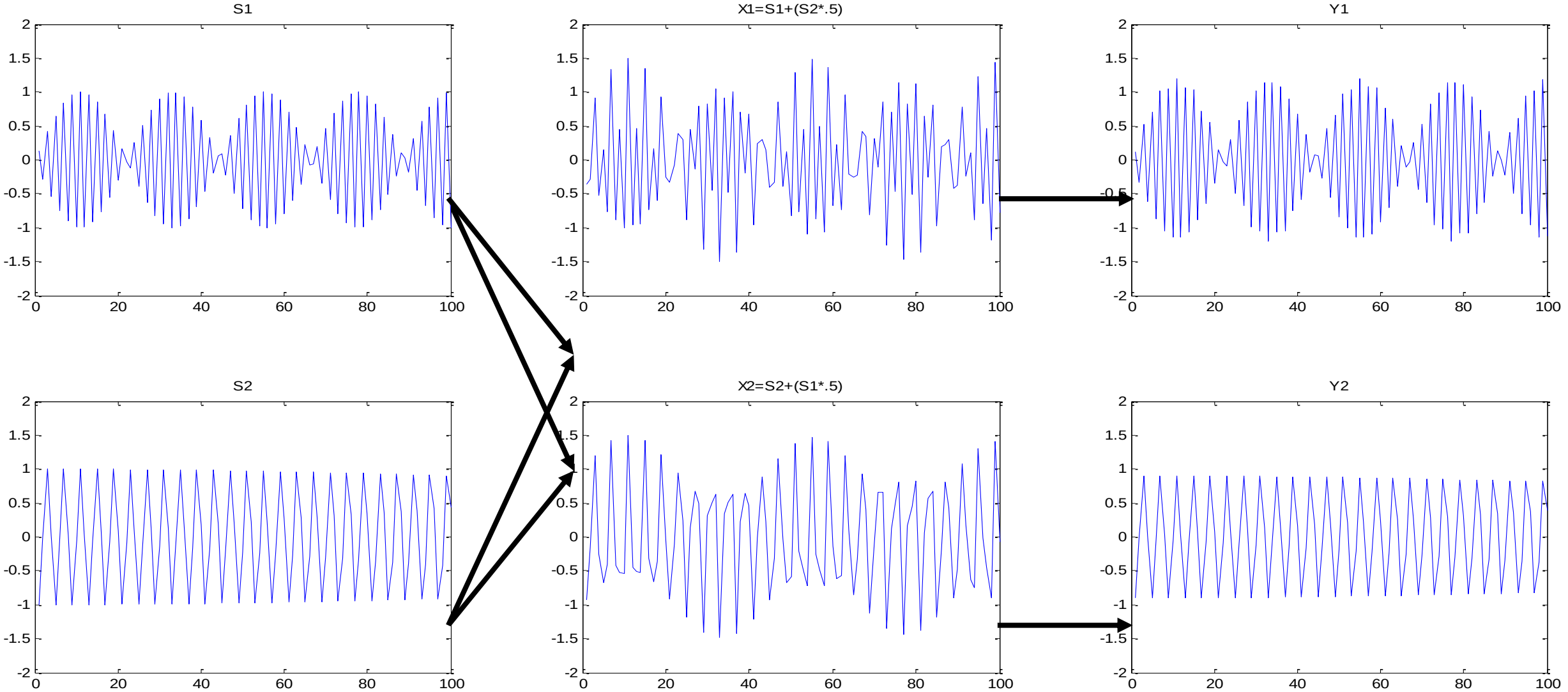
$$x_1 = a_{11} * S_1 + a_{12} * S_2$$
$$x_2 = a_{21} * S_1 + a_{22} * S_2$$

Donde "a" serán parámetros físico que como distancia del micrófono o distorsión de la señal. A esto se le llama problema de la fiesta de cocteles.

El problema de interés a solucionar seria al ecuación matricial de ICA, en el cual normal mente se usa blancamiento y infomax, en nuestro caso el procedimiento será guiado o supervisado y no a ciegas.

$X = A * S$ ecuación ICA

$$\mathrm{ECM} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2.$$




```
Iteraciones = 500  Abejas = 30
X inicial= 1   0   0   1
Elapsed time is 0.991181 seconds.
A= 1.5035    -0.70264    -0.59954       1.2007
```

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

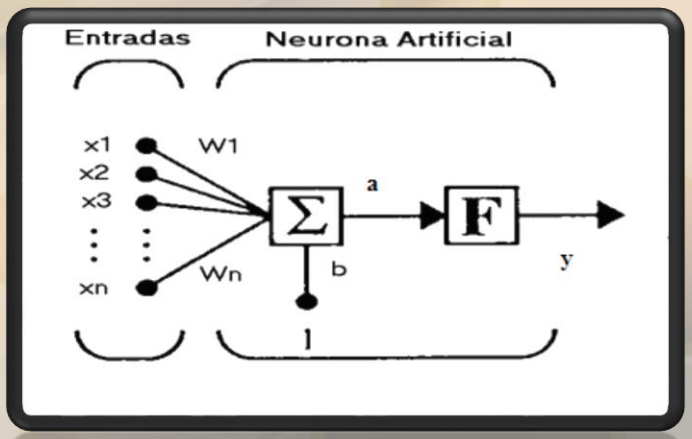
III. Aplicación en software.

vi. Entrenamiento de red neuronal artificial multicapa como aproximado de funciones.

- Neurona artificial: unidad de procesamiento de la información, es un dispositivo simple de cálculo que ante un vector de entradas proporciona una única salida.
- Elementos:
  - Conjunto de entradas, *xj*
  - Pesos sinápticos, *wi*
  - Función de activación:

    $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n = a$
  - Función de transferencia:

    $y = F(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n)$
  - Bias o polarización: entrada constate de magnitud 1, y peso b que se introduce en el sumador



Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

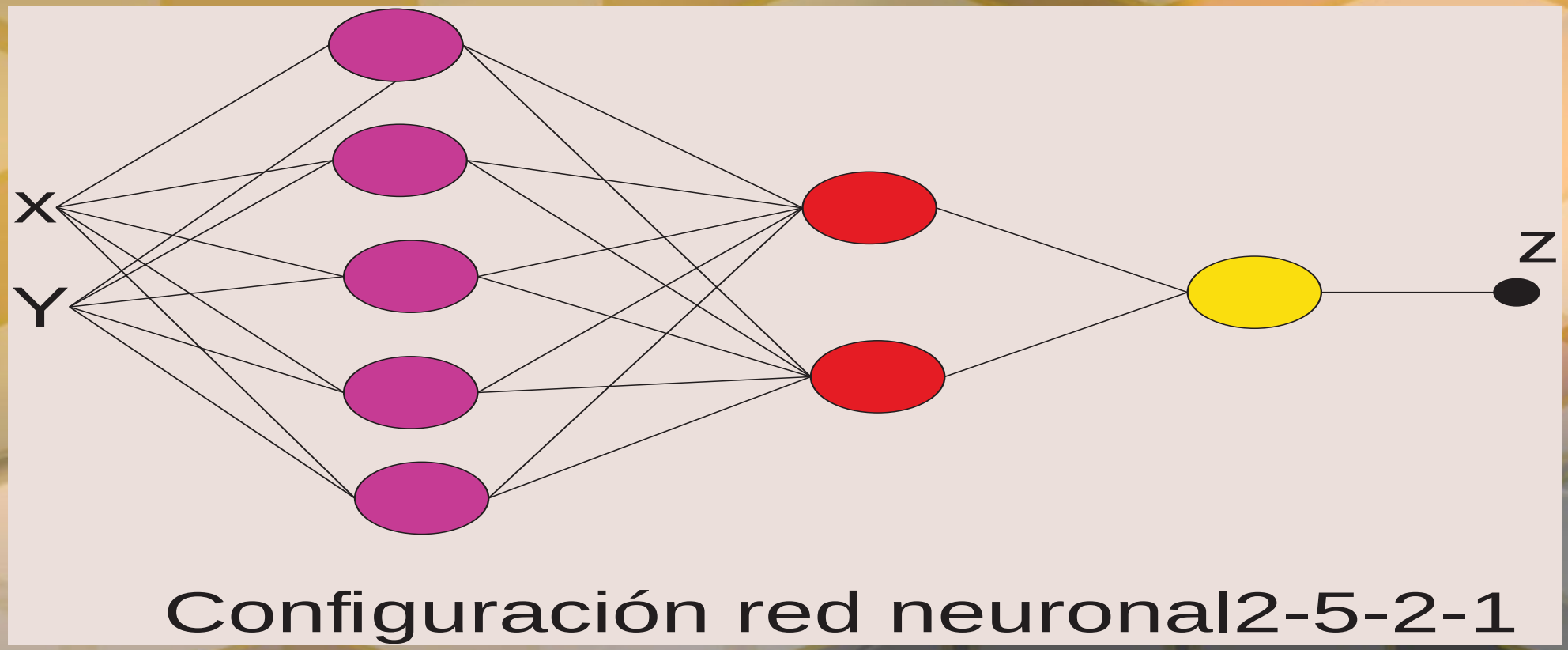
III. Aplicación en software.

vi. Entrenamiento de red neuronal artificial multicapa como aproximado de funciones.

- Función de trasferencia. Tangente Sigmoidal Hiperbólica

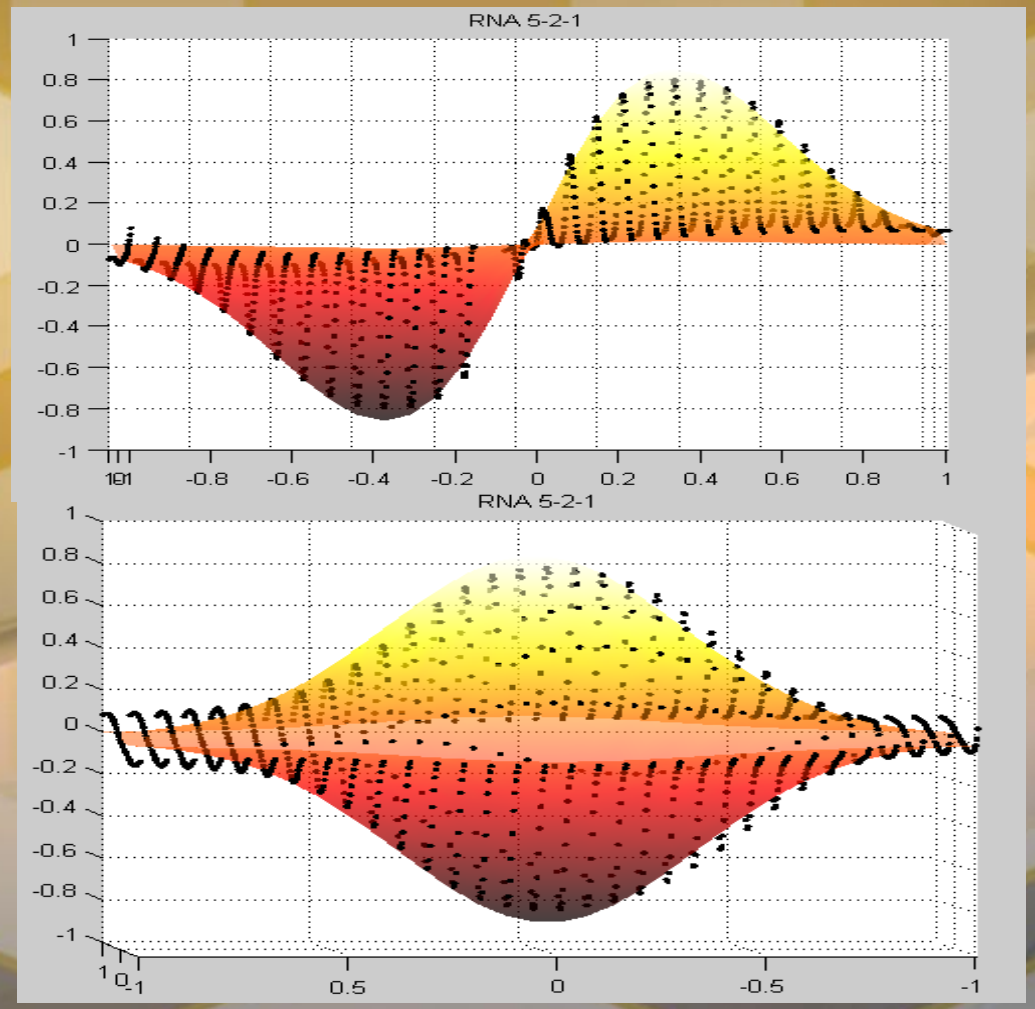
Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular



Configuración red neuronal2-5-2-1

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular




```
Ýter= 14994 ObjVal= 0.063632 r = 1
Ýter= 14995 ObjVal= 0.063632 r = 1
Ýter= 14996 ObjVal= 0.063618 r = 1
Ýter= 14997 ObjVal= 0.063618 r = 1
Ýter= 14998 ObjVal= 0.063618 r = 1
Ýter= 14999 ObjVal= 0.063618 r = 1
Ýter= 15000 ObjVal= 0.063618 r = 1
Minutos= 14.043
```

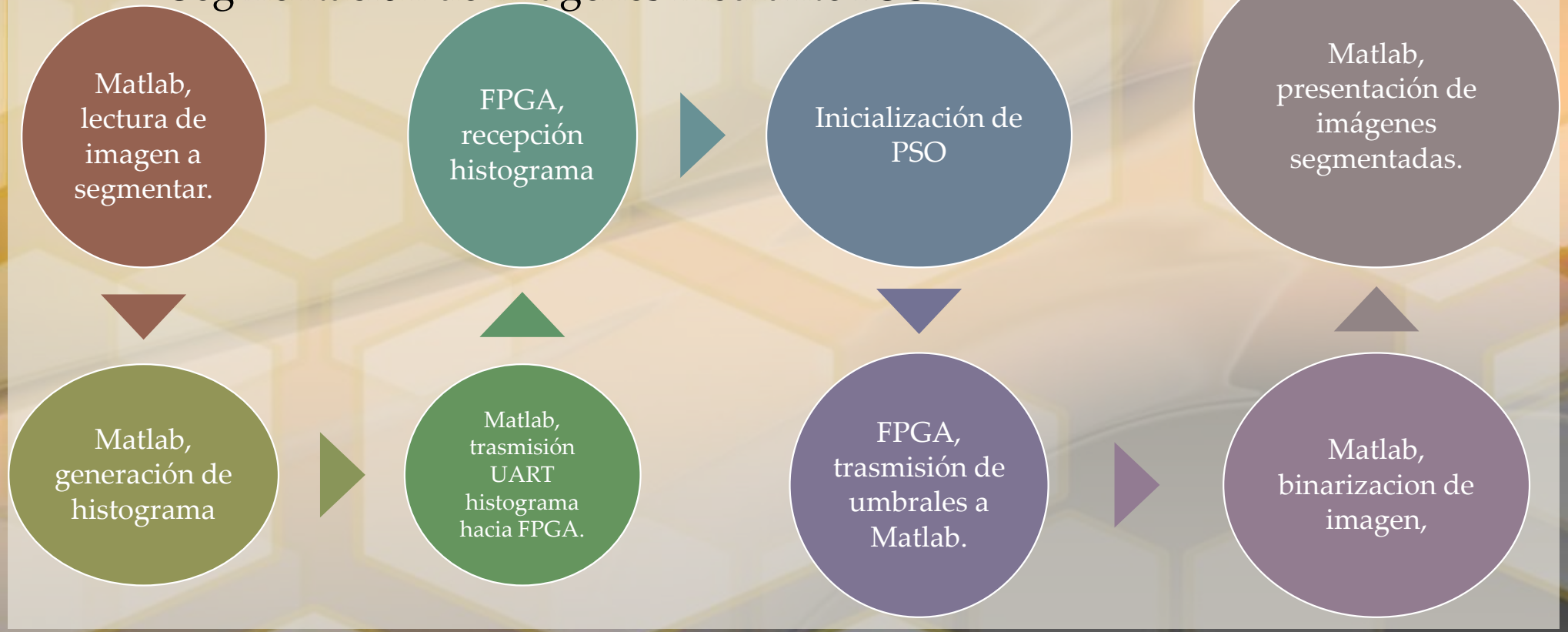
Función a aproximar.

$$Z = 4xe^{-4(x^2-y2\ )}$$

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

iv. Aplicación en Hardware.

Segmentación de imágenes mediante PSO.



Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

```
-------------- Logica del estado Siguiente ----------
process (Estado_Actual,Estado_Siguiente,Up,Cont,ciclo,ciclo_2)
--
begin

case (Estado_Actual) is
      when fin_transmision =>
         if Up='1' then
            Estado_Siguiente<=Rand_X;
         else
            Estado_Siguiente<=fin_transmision;

         end if;
      when Rand_X =>
         if Cont='1'  then
            Estado_Siguiente<=   sumatoria;

         else
            Estado_Siguiente<=Rand_X;
         end if;

      when sumatoria=>

         if Cont='0' and ciclo='0' then
         Estado_Siguiente<=sumatoria;

         else
         Estado_Siguiente<=Funcion;
         end if;
       when Funcion =>
         if   ciclo='0' and Cont='0' then
            Estado_Siguiente<=sumatoria;

         elsif ciclo='0' and Cont='1'  then
            Estado_Siguiente<=Funcion;
         elsif ciclo_2='1' and ciclo='1' and Cont='0'  then
            Estado_Siguiente<=Xi_Max;
         else
            Estado_Siguiente<=Apt_mej;
         end if;
      when Apt_mej =>
            if Cont='0' then
            Estado_Siguiente<=Max_pso;
            else
            Estado_Siguiente<=Apt_mej;
            end if;
       when Max_pso=>
            if Cont='1' then
               Estado_Siguiente<=Vel_inicial;
            else
               Estado_Siguiente<= Max_pso;
            end if;
       when Vel_inicial=>
```

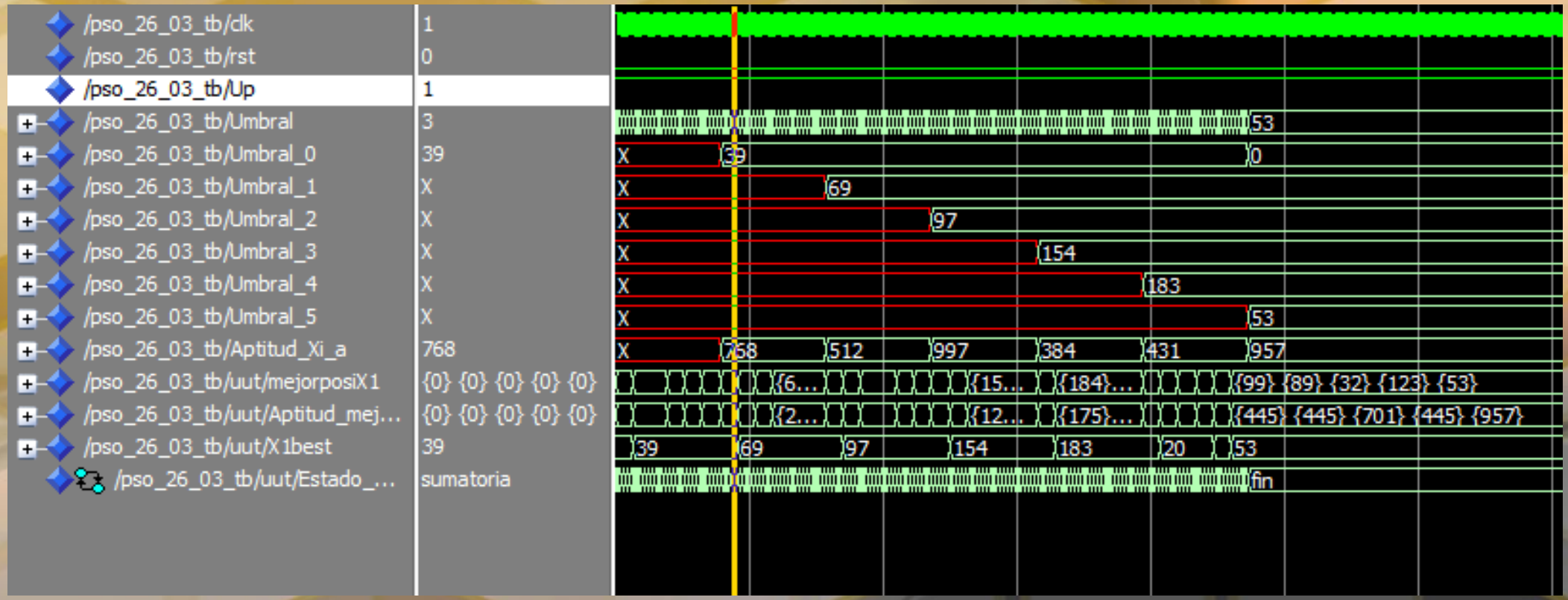
Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

```
        if Cont='1' then
           Estado_Siguiente<=Xi_lim;
        else
           Estado_Siguiente<=Vel_inicial;
        end if;
    when Xi_lim=>
        if Cont='0' and ciclo='0' and ciclo_2='1' then
           Estado_Siguiente<=sumatoria;
        else
           Estado_Siguiente<=Xi_lim;
        end if;
    when Xi_Max=>
        if Cont='1' then
           Estado_Siguiente<=Global;
        else
           Estado_Siguiente<=Xi_Max;
        end if;
    when Global=>
       if Cont='1' then
          Estado_Siguiente<=Vel_completa;
       else
          Estado_Siguiente<=Global;
       end if;
    when Vel_completa=>
       if Cont='1' then
          Estado_Siguiente<=iteracion;
       else
```

```
          Estado_Siguiente<=Vel_completa;
       end if;
    when iteracion =>
       if Cont='1' and ciclo='1' then
          Estado_Siguiente<=Xi_lim;
       elsif Cont='0' and ciclo='0' then
          Estado_Siguiente<=rango;
       else
       Estado_Siguiente<=iteracion;
       end if;
    when rango =>
       if Cont='1' and ciclo='1' and ciclo_2 ='1' then
          Estado_Siguiente <=fin ;
       elsif Cont='0' and ciclo='0' and ciclo_2 ='1' then
          Estado_Siguiente <= rango;
       else
          Estado_Siguiente <=fin_transmision ;
       end if;
    when fin=>
       Estado_Siguiente <= fin;

  end case;
end process;
--
```

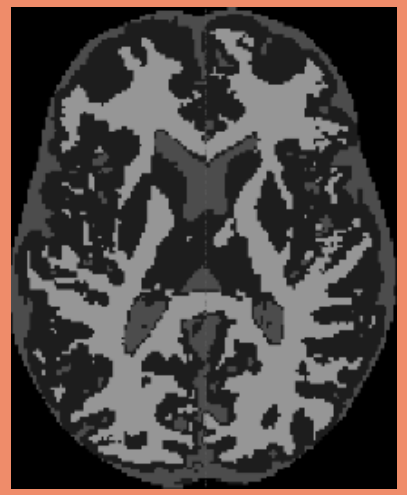
Cinvestav

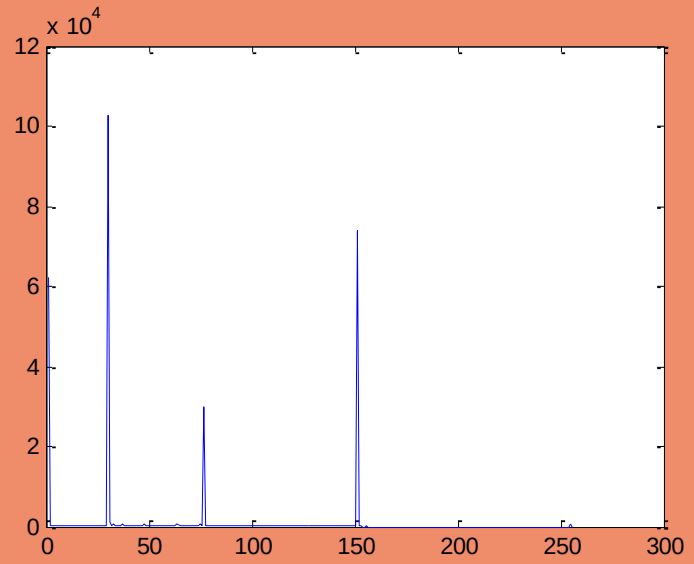
Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular



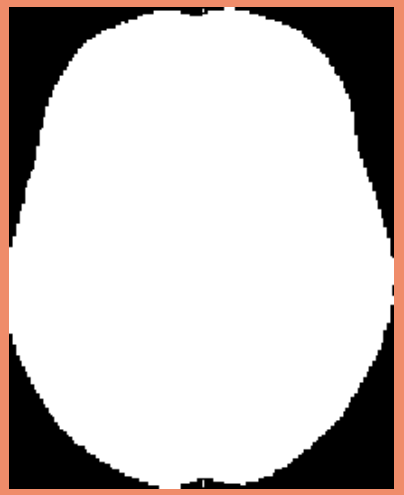
Cinvestav

Tonalidades en Gris





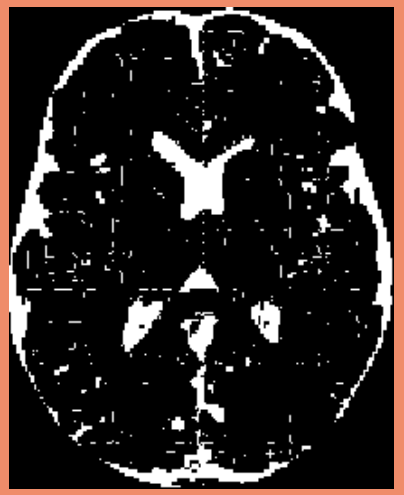
Binario



Binario

Binario

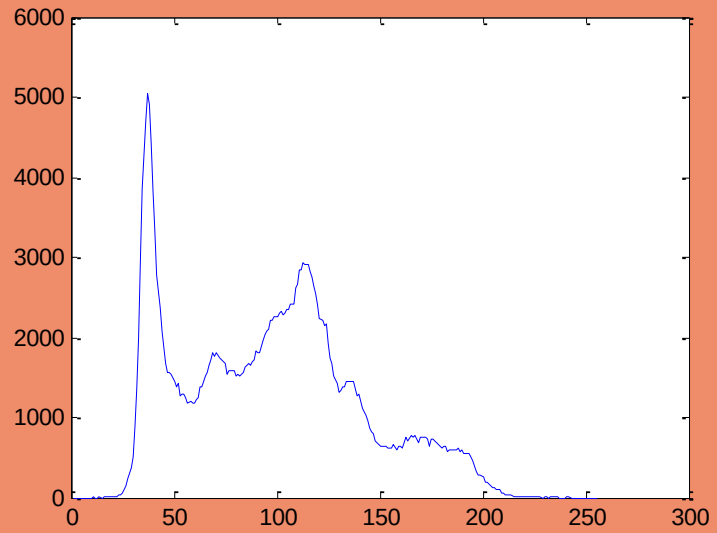
Binario



Grupo de Sistemas VLSI

Cinvestav

Tonalidades en Gris



Binario

Binario

Binario

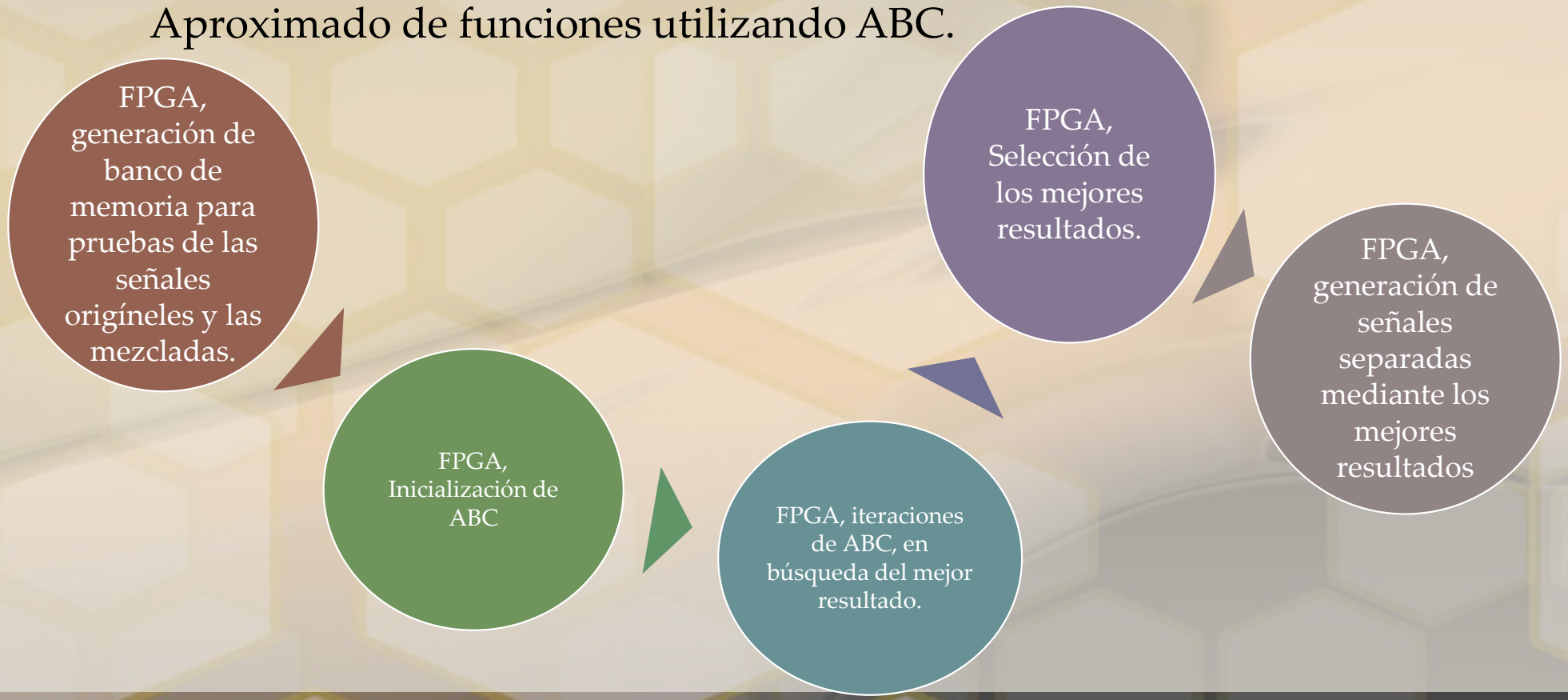
Binario

Grupo de Sistemas VLSI

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

iv. Aplicación en Hardware.

Aproximado de funciones utilizando ABC.



Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Cinvestav

iv. Aplicación en Hardware.

Aproximado de funciones utilizando ABC. Banco de trabajo Señales deseadas.

```
constant S1: s1_a:= (
"0000000100100001","1111110111000011","0000011101001100","1111101110110101","0000010100110011","1111100111111101","0000011010110001","1111100011000001",
"0000011110100110","1111100000011000","0000011111111111","1111100000010000","0000011110110101","1111100010101010","0000011011001110","1111100111011010",
"0000010101011100","1111101110000111","0000001101111101","1111110110001111","0000000101010110","1111111111001001","1111111100010100","0000001000000111",
"1111110011100101","0000010000011100","1111101011110101","0000010111011101","1111100101101100","0000011100100110","1111100001101001","0000011111011110",
"1111100000000001","0000011111110101","1111100000111100","0000011101101010","1111100100010101","0000011001000111","1111101001111011","0000010010100101",
"1111110001010010","0000001010100011","1111111001110011","0000000001101100","0000000010110100","1111111000101100","0000001011100111","1111110000010010",
"0000010011011111","1111101001000111","0000011001110011","1111100011110010","0000011110000100","1111100000101100","0000011111111011","1111100000000101",
"0000011111010000","1111100010000010","0000011100000101","1111100110010111","0000010110101011","1111011100101111","0000001111011101","1111110100101001",
"0000000111000001","1111111101011101","1111111110000001","0000000110011101","1111110101001011","0000001110111101","1111011010011100","0000010110010001",
"1111100110101101","0000011011110011","1111100010001111","0000011111000111","1111100000001000","0000011111111101","1111100000100100","0000011110010000",
"1111100011100001","0000011010001000","1111101000101110","0000010011111011","1111101111110010","0000001100001001","1111111000001001","0000000011011001",
"0000000001001000","1111111010010111","0000001010000001","1111110001110010","0000010010000111","1111101010010110","0000011000110000","1111100100100111",
"0000011101011100","1111100001000101","0000011111110001","1111100000000000");

type s2_a is array (0 to 99) of std_logic_vector(15 downto 0);
constant S2: s2_a:= (
"1111100000000000","1111111111101101","0000011111111111","0000000000100100","1111100000000000","1111111111001001","0000011111111111","0000000001001000",
"1111100000000001","1111111110100101","0000011111111101","0000000001101100","1111100000000011","1111111110000001","0000011111111011","0000000010010000",
"1111100000000101","1111111101011101","0000011111111000","0000000010110101","1111100000001000","1111111100111000","0000011111110101","0000000011011001",
"1111100000001100","1111111100010100","0000011111110001","0000000011111101","1111100000010000","1111111011110000","0000011111101100","0000000100100001",
"1111100000010101","1111111011001100","0000011111100111","0000000101000100","1111100000011011","1111111010101001","0000011111100001","0000000101101000",
"1111100000100001","1111111010000101","0000011111011011","0000000110001100","1111100000101000","1111111001100001","0000011111010011","0000000110101111",
"1111100000101111","1111111001111110","0000011111001100","0000000111010011","1111100000111000","1111111000011011","0000011111000011","0000000111110110",
"1111100001000000","1111111011111000","0000011110111010","0000001000011001","1111100001001010","1111110111010101","0000011110110000","0000001000111100",
"1111100001010100","1111110110110010","0000011110100110","0000001001011111","1111100001011110","1111110110001111","0000011110011011","0000001010000001",
"1111100001101001","1111110101101101","0000011110010000","0000001010100011","1111100001110101","1111110101001010","0000011110000100","0000001011000110",
"1111100010000010","1111110100101000","0000011101110111","0000001011100111","1111100010001111","1111110100000111","0000011101101010","0000001100001001",
"1111100010011100","1111110011100101","0000011101011100","0000001100101011","1111100010101011","1111110011000100","0000011101001101","0000001101001100",
"1111100010111001","1111110010100011","0000011100111110","0000001101101101");
```

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

iv. Aplicación en Hardware.

Aproximado de funciones utilizando ABC. Banco de trabajo Señales mezcladas.

```
type mix1_a is array (0 to 99) of std_logic_vector(15 downto 0); --mix1=s1+(s2*.5)
constant M_1: mix1_a:= (
"1111110100100001","1111110110111010","0000011101001011","1111101111000111","0000000100110100","1111100111100010","0000101010110000","1111100011100101",
"0000001110100111","1111011111101011","0000101111111110","1111100001000111","0000001110110111","1111100001101011","0000101011001100","1111101000100011",
"0000000101011111","1111011001101110","0000011101111001","1111110111101010","1111110101011011","1111111101100110","0000001100001111","0000001001110100",
"1111100011101100","0000001110100111","1111111011101110","0000011001011100","1111010101110101","0000011010011111","1111110001100000","0000100001101110",
"1111010000001100","0000011101011011","1111110000110000","0000100000001100","1111010100100011","0000010110011100","1111111001101100","0000010101011001",
"1111100001100010","0000000111100110","0000001001100001","0000000100110010","1111110011001001","1111110101011101","0000011011010001","1111110011101010",
"0000000011110111","1111100101100111","0000101001011001","1111100111011011","0000001110100000","1111011100111001","0000101111011101","1111100100000000",
"0000001111110000","1111011101111110","0000010101100010","1111101010100011","0000000111010000","1111101000011001","0000011110110110","1111111001000111",
"1111110111101011","1111111000110110","0000001101010100","0000001011001101","1111100101111010","0000001010000101","1111111100011010","0000011011010010",
"1111010111100010","0000010110101010","1111110001010111","0000100100011001","1111010001000011","0000011010100011","1111101111100110","0000100011110011",
"1111010100100010","0000010100011101","1111110111101010","0000011001101111","1111100000111010","0000000110001101","0000000110111110","0000001001011101",
"1111110010010110","1111110100001010","0000011000101111","1111111000001000","0000000011011100","1111100011111000","0000100111010111","1111101011001110",
"0000001110111001","1111011010010111","0000101110010000","1111100110110111");

type mix2_a is array (0 to 99) of std_logic_vector(15 downto 0); -- mix2=s2+(s1*.5);
constant M_2: mix2_a:= (
"1111100010010000","1111111011001111","0000100110100101","1111110111111110","1111101010011010","1111110011001000","0000101101010111","1111110010101001",
"1111101111010100","1111101110110001","0000101111111101","1111110001110101","1111101111011110","1111101111010110","0000101101100010","1111110101111110",
"1111101010110100","1111110100100000","0000100110110111","1111111101111100","1111100010110100","1111111100011101","0000011101111111","0000000111011101",
"1111011001111111","0000000100100011","0000010101101100","0000001111101011","1111010011000111","0000001010000100","0000010000100001","0000010100010000",
"1111010000010110","0000001011000111","0000010000000101","0000010011111001","1111010010100110","0000000111001101","0000010100011111","0000001110111011",
"1111011001001010","1111111111010111","0000011100010100","0000000111000010","1111100010000010","1111110101111000","0000100101000111","1111111111011000",
"1111101010011111","1111101101100010","0000101100000101","1111111001001100","1111101111111010","1111101000110001","0000101111000001","1111110111111001",
"1111110000101000","1111101000111001","0000101100111101","1111111011100101","1111011100011111","1111101101101100","0000100110011111","0000000011010000",
"1111100100110100","1111110101100000","0000011101100111","0000001100101110","1111011100000100","1111111101101110","0000010101000001","0000010101001010",
"1111010101000000","0000000011100111","0000001111010111","0000011010000111","1111010001111010","0000000101001001","0000001110010110","0000011010001110",
"1111010011110010","0000000001101101","0000010010001110","0000010101100101","1111011010001000","1111111010001011","0000011001101110","0000001101110110",
"1111100011000001","1111110000110001","0000100010011100","0000000101100100","1111101011101110","1111101000001111","0000101001100110","1111111111100000",
"1111110001100111","1111100011000110","0000101100110111","1111111101101101");
```

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

iv. Aplicación en Hardware.

Aproximado de funciones utilizando ABC.

```
---------------------------  Multiplicacion de señal mista por  matrix A  ---------------------------
when Mult_mixA=>
   i_a0<=i_b;
   i_a1<=i_a0+1;

                  x1_1<=X1(a1);
                  x1_2<=X1(a2);
                  x1_3<=X1(a3);
                  x1_4<=X1(a4);

   M_1b1<= M_1(i_a0)*X1(a1)+M_2(i_a0)*X1(a2);
   M_1b2<= M_1(i_a1)*X1(a1)+M_2(i_a1)*X1(a2);

   M_2b1<= M_1(i_a0)*X1(a3)+M_2(i_a0)*X1(a4);
   M_2b2<= M_1(i_a1)*X1(a3)+M_2(i_a1)*X1(a4);
--
   M_1a(i_a0)<=M_1b1(26 downto 11);
   M_1a(i_a1)<=M_1b2(26 downto 11);

   M_2a(i_a0)<=M_2b1(26 downto 11);
   M_2a(i_a1)<=M_2b2(26 downto 11);

   if clk_2>6 then

         if i_b>=98 then  --- Control de indexe de avance al terminar el muestreo se sale
            control<="0010" ; -- y pasa a la parte ECM con el mismo índice de j
         else
            i_b<=i_b+2;
            clk_2<=0;
         end if;
   else
         clk_2<=clk_2+1;
    end if;
```

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Cinvestav

iv. Aplicación en Hardware.

Aproximado de funciones utilizando ABC.

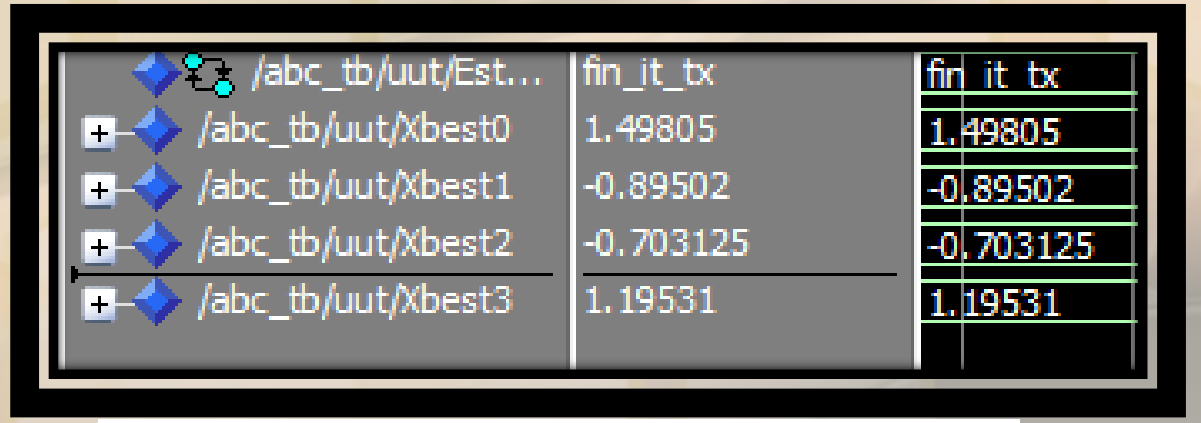
```
when sumatoria=>
            if iter>=99 then
               control<="0011";
               M_S_2(index)<= M_S;
            else
               M1_S1:=M_1a(iter)-S1(iter);
               M2_S2:=M_2a(iter)-S2(iter);

               M1_S1_a:=(M1_S1)*(M1_S1);
               M2_S2_a:=(M2_S2)*(M2_S2);

               M1_S1_b:=(M1_S1_b)+unsigned(M1_S1_a);-- borrar despues de usar
               M2_S2_b:=(M2_S2_b)+unsigned(M2_S2_a); -- borrar despues de usar
               M_S<=M1_S1_b+M2_S2_b;
               iter<=iter+1;
```

```
when control_inter =>
                     M1_S1_b:="00000000000000000000000000000000";-- borrar despues de usar
                     M2_S2_b:="00000000000000000000000000000000"; -- borrar despues de usar
                     M_1b1<= X"00000000";
                     M_1b2<= X"00000000";
                     M_2b1<= X"00000000";
                     M_2b2<= X"00000000";
                        M_1a<=(others=>X"0000");
                        M_2a<=(others=>X"0000");
                     i_b<=0;
                     i_a1<=0;
                     i_a0<=0;
                     a1<=a1+4; a2<=a2+4;
                     a3<=a3+4; a4<=a4+4;
                     clk_2<=0;
                     index<=index+1;
```

Cinvestav

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

- No. De iteraciones.- 500
- No. De Abejas.- 30.
- Fuente de alimento inicial .- [1,0,0,1]




```
Iteraciones = 500  Abejas = 30
X inicial= 1   0   0   1
Elapsed time is 0.991181 seconds.
A= 1.5035    -0.70264    -0.59954      1.2007
```

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

vi. Conclusiones.

- Los algoritmos meta-heurísticos presentan resultados eficientes y aceptables a comparación de algoritmos de optimización clásicos.

- Estos métodos demostraron obtener valores de los parámetros con una determinada exactitud, mediante las aplicaciones en software como en hardware, con la ventaja de disminuir drásticamente el uso de recursos en ambos.

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

vii. Trabajo Futuro.

- Utilizar algoritmos meta-heurísticos, en redes neuronales para aumentar la exactitud de su funcionamiento haciendo crecer las capas ocultas de una red neuronal según los resultados el entrenamiento mediante algoritmos meta-heurísticos.
- Se continuara, trabajando en procesamiento de imágenes para optimiza de borde en imágenes para el reconocimiento de objetivos, entre otros.

Algoritmo de Optimización Meta-Heurístico Aplicado a una Red Neuronal Celular

Gracias por su atención.