Ciencias Matemáticas, Vol. 33, No. 1, Pag. 46-55, 2019

Algoritmos Heurísticos Híbridos para diseño de S-Caja
Hybrids Heuristics Algorithms for S-box Design

Antonio Bolufé Rohler, Dania Tamayo Vera, Rodnel Peña Alvarado

Resumen Las S-cajas son un componente fundamental dentro de los algoritmos de clave simétrica. S-cajas con buenas propiedades criptográficas garantizan la seguridad ante los diversos criptoanálisis existentes. De especial interés resultan las propiedades de orden de transparencia y no-linealidad, que miden la resistencia ante el ataque diferencial de potencia y el criptoanálisis lineal, respectivamente. El diseño de S-cajas con un alto valor de no-linealidad y un bajo orden de transparencia es un problema NP-duro. La aplicación de algoritmos metaheurísticos para el diseño de S-cajas data de finales de los años 90. Sin embargo, los enfoques que se han presentado a lo largo de todos estos años son muy similares y pocos sofisticados desde el punto de vista del diseño de las metaheurísticas. En esta tesis se presenta un nuevo enfoque para el diseño de S-cajas: la aplicación de algoritmos heurísticos híbridos y la utilización de técnicas de aprendizaje de máquina para encontrar el mejor punto de transición entre las técnicas híbridas. Los resultados obtenidos en esta tesis son comparables con los mejores resultados alcanzados en trabajos previos.

Abstract The S-boxes are a fundamental component within the symmetric key algorithms. S-boxes with good cryptographic properties guarantee security before the various existing cryptanalysis. Of particular interest are the properties of order of transparency and non-linearity, which measure resistance to differential power attack and linear cryptanalysis, respectively. The design of S-boxes with a high non-linearity value and a low order of transparency is a NP-hard problem. The application of metaheuristic algorithms for the design of S-boxes dates from the late 90s. However, the approaches that have been presented over all these years are very similar and not very sophisticated from the point of view of the design of the metaheuristics. This thesis presents a new approach to the design of S-boxes: the application of hybrid heuristic algorithms and the use of machine learning techniques to find the best transition point between hybrid techniques. The results obtained in this thesis are comparable with the best results achieved in previous works.

Palabras Clave
Algoritmos Heurísticos, Metaheurístico, Optimización, Criptografía, S-cajas, Criptoanálisis.

[1] *Instituto de Criptografía, universidad de la Habana, La Habana, Cuba, teresa.bernarda@matcom.uh.cu*
[1] *Instituto de Criptografía, universidad de la Habana, La Habana, Cuba, rodnel.pena@matcom.uh.c´*

1. Introducción

Una heurística es un algoritmo de optimización que garantiza encontrar una solución buena en un tiempo razonable. Estos algoritmos son cada vez más utilizados al resolver complejos problemas de optimización. Muchas heurísticas son específicas y dependientes del problema. un algoritmo metaheurístico es un marco de trabajo algorítmico, de alto nivel e independiente del problema, que ofrece un conjunto de estrategias o reglas generales para diseñar algoritmos heurísticos. Las metaheurísticas permiten combatir de manera efectiva un problema inherente a los métodos heurísticos y en general a muchos métodos de búsqueda: el estancamiento en óptimos locales. La aplicación de estos algoritmos a la optimización de funciones multi-modales ha sido objeto de estudio en los últimos tiempos. La motivación viene dada porque la mayoría de los problemas de la vida real tienen un carácter multimodal. Las funciones multi-modales se caracterizan por tener múltiples campos de atracción. un campo de atracción es una región del espacio de búsqueda en cuyos puntos al seguir el gradiente de la función se converge a un mismo óptimo local. En este tipo de funciones el proceso de optimización puede ser dividido en dos etapas: exploración y explotación. Podemos definir exploración como la etapa del proceso de optimización encargada de detectar los mejores campos de atracción y la explotación como la etapa en la cual se converge al correspondiente óptimo local de los campos seleccionados. Líderes y Seguidores (*Leader and Followers*) es una metaheurística que se enfoca en evitar que soluciones obtenidas mediante explotación sean comparadas con nuevas soluciones exploratorias. Para ello se utilizan dos poblaciones: una población de líderes (*leaders*) que ha acumulado explotación y que guía la búsqueda y una solución de seguidores (*follo-*

wers) encargadas de explorar nuevas regiones del espacio. En un algoritmo híbrido la elevada capacidad de exploración de Líderes y Seguidores se combina con algoritmos de búsqueda local que permiten, una vez concluida la exploración, converger rápidamente a los óptimos locales. En la época actual el intercambio de información por medios electrónicos es algo cotidiano. Para garantizar la seguridad y privacidad de los datos existen diversos tipos de algoritmos de encriptación, entre los que se destacan por su fortaleza los algoritmos de clave simétrica como el Advanced Encryption Standard (AES). Las S-cajas (o cajas de substitución) constituyen un componente principal en el procedimiento de cifrado, estas aportan la no-linealidad y la confusión necesarias para que el algoritmo sea considerado seguro. Aunque se conocen S-cajas con buenas propiedades criptográficas, la utilización de múltiples S-cajas durante el proceso de cifrado es la transformación más usual dentro del grupo de variantes propuestas al algoritmo AES [**?**]. Debido a esto se hace necesario encontrar S-cajas con un conjunto de criterios que permitan garantizar la seguridad de las mismas ante las diversas técnicas de criptoanálisis existentes: el criptoanálisis diferencial y lineal, el algebraico y el ataque diferencial de potencia, entre otros. En la literatura es posible encontrar diversos trabajos donde se aplican algoritmos heurísticos y metaheurísticos a la optimización de las propiedades de las S-cajas. John Clark, uno de los principales investigadores del tema, expresa que hasta el momento se han aplicado (meta)heurísticas muy simples como Recocido Simulado y Algoritmos Genéticos, y plantea la necesidad de aplicar algoritmos más novedosos y sofisticados al problema en cuestión. También expresa que las funciones objetivos utilizadas han sido directamente las funciones de costo asociadas a las propiedades que se quiere mejorar, y considera que estas no tienen por qué ser necesariamente las mejores funciones para guiar una búsqueda heurística.

2. Marco Teórico

En la actualidad existen varios tipos de algoritmos criptográficos: los de clave simétrica y los de clave asimétrica. Esta investigación se enmarca en la criptografía de clave simétrica, específicamente en los cifrados en bloques. Quizás el más famoso de estos algoritmos sea el DES (*Data Encryption Standard*) [**?**]. En los años 90 este había llegado al fin de su vida útil siendo sustituido por el AES (*Advanced Encryption Standard*). Su estructura consta de cuatro transformaciones básicas, las cajas de sustitución (S-cajas) (SubBytes), las matrices MDS (MixColumns), el desplazamiento de las filas (ShiftRows) y la suma Xor con las subclaves (AddRoundKey). Las S-cajas constituyen un componente imprescindible en el diseño de cifrados de bloques, estas aportan la no-linealidad y la confusión para que el algoritmo sea seguro. Debido a esto se hace necesaria la definición de un conjunto de criterios que permiten determinar la seguridad de las S-cajas: el criptoanálisis diferencial y lineal, el algebraico y el ataque diferencial de potencia (DPA)[**?**] [**?**], entre otros. La no linealidad es una de las propiedades más importantes. Un elevado valor de la no linealidad de una S-caja reduce la efectividad de los ataques por criptoanálisis lineal. Otra de las propiedades importantes es el orden de transparencia. Mientras menor sea el orden de transparencia de una S-caja, mayor es su resistencia ante ataques DPA. una S-caja es una función booleana vectorial de n bits de entrada a m bits de salida $\phi : F_2^n \longrightarrow F_2^m$

Definición 2.1 (Función Booleana) *Una función booleana $f(x) : F_2^n \longrightarrow F_2$ es la función que le hace corresponder a n bits de entrada un bit de salida.*

Definición 2.2 (Función Afín) *Una función booleana ϕ es llamada función afín si es de la forma*

$$\phi(x) = b_1x_1 \oplus b_2x_2 \oplus ..... \oplus b_nx_n \oplus c = b * x \oplus c \qquad (1)$$

Donde $b_1, b_2, ..., b_n, c \in F_2$ y $b, x \in E_2^n$. Si $c = 0$ entonces ϕ es llamada función lineal.

Al conjunto de todas las funciones afines se le denota por A_n y al de todas las funciones lineales por L_n.

Definición 2.3 (Peso de Hamming) *El* peso de Hamming *de una función booleana se refiere al número de* 1 *en su tabla veritativa correspondiente a la salida y se denota por $Hw(\phi)$.*

$$Hw(\phi) = |f| = \sum_{x \in F_2^n} \phi(x) \qquad (2)$$

Definición 2.4 (Distancia de Hamming) *La* distancia de Hamming *entre dos funciones booleanas es definida como el número de posiciones diferentes entre sus tablas de verdad y se expresa como*

$$d(\phi, \varphi) = Hw(T_\phi \oplus T_\varphi) \qquad (3)$$

Definición 2.5 (Funciones Componentes) *Sea $\phi : F_2^n \longrightarrow F_2^m, \phi = (\phi_1, ..., \phi_m)$, se le llaman funciones componentes o funciones coordenadas de ϕ a las funciones booleanas de n entradas ϕ_i $\forall i \in 1, 2, ..., m$ que cumplen que $\phi_i(x) = y_i$.*

Definición 2.6 (Forma Normal) *Se denomina forma algebraica normal (FAN) de la función binaria $f(x)$ a la expresión del tipo:*

$$\phi(x_1, ...x_n) = \sum_{k=0}^{n} \sum_{i_1....i_k} a_{i_1....i_k} x_{i_1} ....x_{i_k} \qquad (4)$$

Definición 2.7 (Grado Algebraico de una Función Booleana) *El grado algebraico de una función booleana ϕ se define como el grado del polinomio 2.6 que es la forma normal de ϕ. Se denota como $deg(\phi)$.*

Definición 2.8 (Grado Algebraico) *El grado algebraico de ϕ, $deg(\phi)$ se define como el mínimo de los grados algebraicos de todas las combinaciones lineales no nulas de las funciones componentes de ϕ.*

Definición 2.9 (Transformada de Walsh) *Se denota por W_ϕ a la transformada de Walsh de la función ϕ,y es definida como:*

$$W_\phi(w) = \sum_{x \in F_2^n} \hat{\phi}(x)\hat{L_w}(x) \quad (5)$$

donde $\hat{\phi}(x)$ es la forma polar de la función booleana $\phi : \hat{\phi}(x) = (-1)^{\phi}(x)$ El valor máximo absoluto de la transformada de Walsh Hadamard de una función ϕ es denotado por:

$$W_{MAX}(\phi) = \max_{w \in F_2^n} \|W_\phi(w)\| \quad (6)$$

2.1 No Linealidad

Definición 2.10 (No Linealidad de una función booleana [?]) *La no linealidad de una función booleana ϕ de n variables se define como el mínimo de la distancia de hamming 2.4 entre esta y cualquier función lineal. Esta es definida como:*

$$NL_\phi = \frac{1}{2}(2^n - W_{MAX}(\phi)) \quad (7)$$

Definición 2.11 (No linealidad de una S-Cajas) *La no linealidad de una S-caja es el mínimo de los valores de no linealidad de las funciones booleanas que la componen.*

2.2 Orden de Transparencia

Los ataques por consumo de potencias (DPA) tienen como idea revelar la clave de un dispositivo criptográfico por medio del análisis de consumo de potencia mientras el dispositivo criptográfico cifra o descifra diferentes bloques de datos[**?**]. El orden de transparencia es una propiedad de las S-cajas que brinda resistencia ante los ataques DPA. El orden de transparencia de una función $\phi : F_2^n \longrightarrow F_2^m$ de define como [**?**]:

$$Aux(B,a) = \sum_{v \in F_{2m}} (-1)^{v*B} WD_a F(0,v) \quad (8)$$

$$T_F = \max_{B \in F_2^m} \{|m - 2Hw(B)| - \frac{1}{(2^{2n} - 2^n)} \sum_{a \in F_{2n}} |Aux(B,a)|\} \quad (9)$$

$$WD_a\phi(0,v) = corr(\phi,\varphi) = \sum_{x \in F_{2n}} (-1)^{\phi(x)+\varphi(x)} \quad (10)$$

$$\phi(x) = v\phi(x), \varphi(x) = v\phi(x+a). \quad (11)$$

Teorema 2.1 *Sean n y m dos enteros positivos, el orden de transparencia T_ϕ de toda función $\phi : F_2^n \longrightarrow F_2^m$ satisface la siguiente relación:*

$0 \leq T_\phi \leq m$

Si toda función coordenada ϕ es bent $T_\phi = m$, y $T_\phi = 0$ si y solo sí ϕ es una función afín.

Definición 2.12 (Función Bent) *Una función booleana ϕ es llamada una función bent si muestra el máximo valor de no linealidad alcanzable y se definen como:*

$$\phi_{bent} \Leftrightarrow \forall w \in F_2^n \|W_\phi(w)\| = 2^{\frac{n}{2}} \quad (12)$$

2.3 NP-Completo

Encontrar funciones booleanas y S-cajas que logren un balance adecuado entre propiedades antagónicas resulta complejo por el balance que debe lograrse para no exhibir vulnerabilidades explotables. El espacio de las funciones booleanas es exponencial respecto al tamaño de entrada $n : 2^{2^n}$. En aras de hacer los algoritmos criptográficos más resistentes, el tamaño de las funciones booleanas está en continuo incremento, y cuando n crece, el espacio de búsqueda crece de forma exponencial [**?**]. Clark demuestra en [**?**] que la obtención de funciones booleanas con buenas propiedades criptográficas, pertenece a la clase NP-Completo. Es por ello que una de las principales líneas de investigación en este campo ha sido la aplicación de metaheurísticas para encontrar funciones booleanas y S-cajas con propiedades deseables.

2.4 Algoritmos Heurísticos y Metaheurísticos

Los algoritmos de optimización pueden ser divididos en dos categorías: algoritmos exactos y algoritmos heurísticos. Los métodos exactos están diseñados para encontrar la solución óptima en un tiempo finito. Los problemas NP-duros, este "tiempo finito"puede aumentar de manera exponencial respecto al número de dimensiones. Los algoritmos heurísticos no garantizan encontrar el óptimo, pero permiten encontrar soluciones buenas en un "tiempo razonable". Muchas heurísticas son específicas y dependientes del problema. Por el contrario, una metaheurística es un marco de trabajo algorítmico, de alto nivel e independiente del problema.

2.5 Exploración y Explotación

La optimización de funciones multi-modales requiere que una metaheurística efectúe exploración y explotación. La exploración consiste en encontrar las regiones más prometedoras del espacio de búsqueda, la explotación debe garantizar la convergencia al óptimo (local) de dicha región. *Thresheld Convergence* (TC) es una técnica diseñada para promover una exploración no sesgada del espacio de búsqueda [**?**], de manera que se facilite la detección de los mejores *campos de atracción* antes de efectuar la explotación de los mismos [**?**] [**?**] [**?**] [**?**]. Un campo de atracción es una región del espacio de búsqueda desde cuyos puntos cualquier algoritmo de búsqueda local converge siempre al mismo óptimo local. La integración de *Thresheld Convergence* ha permitido mejorar los resultados de diversas metaheurísticas al optimizar funciones multi-modales.

2.6 Líderes y Seguidores(Leader and Followers)

Definición 2.13 (Calidad de un campo de atracción) *Sea C_a un campo atracción de la función f y x^* su óptimo local correspondiente, la calidad de C_a se define como la calidad de su óptimo local: $f(x^*)$.*

Definición 2.14 (Comparación sesgada) *Sean C_{a_1} y C_{a_2} campos de atracción de la función f y x_1 y x_2 soluciones tales que $x_1 \in C_{a_1}$ y $x_2 \in C_{a_2}$. Una comparación sesgada ocurre $f(x_1)\, f(x_2)$ y $f(x_{1^*})\, f(x_{2^*})$, donde x_{1^*} y x_{2^*} son los óptimos locales de C_{a_1} y C_{a_2}, respectivamente.*

La metaheurística de *Líderes y Seguidores* (LaF, por sus siglas en inglés) hace énfasis en evitar comparaciones sesgadas. El rasgo distintivo de esta metaheurística es el uso de dos poblaciones: una población de "líderes"(*Lp*) que contiene las soluciones de los mejores campos de atracción y las cuales guían la búsqueda, y una población de "seguidores"(*Sp*) en las cuales se efectúan comparaciones no sesgadas. La población de líderes, almacena las mejores soluciones encontrada, pero estas no se comparan con las nuevas soluciones. Cuando *Sp* alcanza la media de la *Lp*, las dos poblaciones se mezclan mediante una selección por torneo y los mejores individuos pasan a formar una nueva población de líderes. Esto permite a los seguidores convertirse en líderes y que el proceso de búsqueda mejore.

Algorithm 1: Líderes y Seguidores (LaF)

```
Entrada : maxEvals
L ←Inicializar aleatoriamente los líderes
S ←Inicializar aleatoriamente los seguidores
repeat
    for i ←1,n do
        lider ←Escoger un lider de L
        seguidor ←Escoger un seguidor de S
        Xnueva ←crearIndividuo(lider , seguidor )
        if f(Xnueva) < f(seguidor) then
            seguidor ←Xnueva
        end
    end
    if mediana(f(S)) < mediana(f(L)) then
        L ←mezclarPoblaciones(L , S )
        S ←Inicializar aleatoriamente los seguidores
    end
until evals < maxEvals
Retornar lider_b
```

3. Hibridación

El propósito fundamental de la hibridación entre distintas metaheurísticas consiste en complementar las debilidades de un algoritmo con las fortalezas de otro. Combinar las metaheurísticas con técnicas de aprendizaje de máquinas constituye otra forma de mejorar la eficiencia y la efectividad de los algoritmos heurísticos. El aprendizaje de máquina es el proceso de explorar grandes volúmenes de datos para extraer conocimiento [**?**].

- Aprendizaje supervisado: cuando se cuenta con información que permite clasificar/describir los datos observados.
- Aprendizaje no supervisado: cuando solo se conocen los datos.

Este artículo se enfoca en la hibridación utilizando técnicas de aprendizaje supervisado, usando técnicas de clasificación. Los clasificadores examinan los atributos de una instancia dada para asignarla a una categoría o clase predefinida. La extracción de conocimiento se puede efectuar antes de comenzar la búsqueda, lo cual se conoce como *estrategia de conocimiento off-line* [**?**], o dinámicamente durante la búsqueda, que se conoce como *estrategia de conocimiento on-line* [**?**]. El clasificador se puede utilizar para modificar un componente de la metaheurística, o para ajustar parámetros y tomar decisiones durante la ejecución del algoritmo [**?**]. En esta investigación se hará uso de estrategia de conocimiento off-line para la toma de decisiones durante la ejecución del algoritmo.

3.1 Hibridación entre Heurísticas

La metaheurística Líderes y Seguidores está compuesta por dos operadores: el operador de cruzamiento, que permite generar una nueva solución a partir de un seguidor y un líder, y el operador de selección, que mezcla las poblaciones de seguidores y de líderes en cada reinicio. Para adaptar LaF a problemas combinatorios se reemplazó el operador de cruzamiento continuo [**?**] por el operador de cruzamiento por orden (*order crossover*). La hibridación de realizó con la metaheurística simple Búsqueda Local. Esta metaheurística en cada iteración este algoritmo evalúa todas las soluciones vecinas e itera a la mejor de ellas, siempre y cuando sea mejor que la solución actual. El criterio de vecindad que se utilizó fue el intercambio entre todos los pares de elementos distintos. El criterio propuesto será igual a las combinaciones de 2 en 256. El algoritmo híbrido propuesto consta de dos fases distintas: en la primera fase se lleva a cabo la exploración y en la segunda fase se explotan las mejores regiones encontradas. En la primera fase se ejecuta LaF, teniendo como condición de parada un número máximo de evaluaciones y se retorna la población de líderes. En la segunda fase se aplica Búsqueda Local a cada uno de los líderes (empezando por los mejores). A continuación, un seudocódigo del híbrido LaF-BL, donde σ_t es el parámetro que controla el punto de transición entre los procesos de exploración y explotación.

Se establece como condición de parada un número máximo de evaluaciones de la función objetivo. se fijan tres momentos distintos para hacer la transición: 30 %, 50 % y 70 % de las evaluaciones (σ_t = 0,3, 0,5 y 0,7). Los correspondientes híbridos se denotarán respectivamente como LaF-BL-30, LaF-BL-50 y LaF-BL-70.

3.2 Resultados

Se realiza un ajuste de parámetros para determinar el tamaño óptimo de población en LaF. En la literatura no se reporta el tamaño óptimo de población para problemas combinatorios. Para ajustar este parámetro se realizaron 30 corridas del algoritmo. Los tamaños de población: 100,200,300,400 y 500; se fijó 500.000 como el número máximo de evaluaciones

Algorithm 2: Algoritmo Híbrido (LaF-BL)

```
Entrada : maxEvals, pTransicion
evalsLaF = maxEvals * pTransicion
evalsBL = maxEvals * (1-pTransicion)
L = LaF(evalsLaF)
foreach líder ∈ L do
    while evals < evalsBL do
        BL(líder, evalsBL)
    end
end
```

de la función objetivo. Para ambas propiedades los mejores resultados se tuvieron con un tamaño de población de 300. A continuación, se presentan los resultados alcanzados por distintos algoritmos para las propiedades de no-linealidad y el orden de transparencia. Los algoritmos descritos en secciones previas (LaF, BL, LaF-BL-30, LaF-BL-50 y LaF-BL-70) se ejecutaron un algoritmo genético y un algoritmo aleatorio. El algoritmo genético implementado utiliza un operador estándar de selección por torneo. Operador de mutación se utilizó el intercambio de dos componentes, la probabilidad de mutar se fijó en p = 0,8.

3.3 Función Objetivo Compuesta

El orden de transparencia y la no-linealidad son antagónicas [**?**] en el sentido de que al mejorar los valores de una empeoran los de la otra. Por ello, el problema de optimizar S-cajas con buenas propiedades criptográficas es inherentemente un problema de optimización multi-objetivo.

Definición 3.1 (Problema de Optimización Multi-objetivo) *Un problema de optimización se dice que es multi-objetivo si se optimizan simultáneamente k criterios de decisión sobre un mismo espacio de búsqueda.*

$$min f_1(x_1, x_2, ..., x_n) \tag{13}$$

$$min f_2(x_1, x_2, ..., x_n) \tag{14}$$

$$\vdots \tag{15}$$

$$min f_k(x_1, x_2, ..., x_n) \tag{16}$$

sujeto a restricciones de igualdad y desigualdad:

$$f_1(x_1, x_2, ..., x_n) = 0 \tag{17}$$

$$\vdots \tag{18}$$

$$f_s(x_1, x_2, ..., x_n) = 0 \tag{19}$$

$$g_1(x_1, x_2, ..., x_n) = 0 \tag{20}$$

$$\vdots \tag{21}$$

$$g_l(x_1, x_2, ..., x_n) = 0 \tag{22}$$

Para solucionar esta deficiencia se recomienda optimizar las S-cajas desde una perspectiva multi-objetivo, es decir, teniendo en cuenta simultáneamente todas las propiedades. La forma más sencilla de optimizar varias funciones objetivos simultáneamente es ponderándolas dentro de una misma función. Este enfoque tiene como ventaja no modificar los algoritmos de optimización. Siendo la función:

$$f_{comp} = \alpha * OT - (1 - \alpha) * NL \tag{23}$$

Donde $\alpha \in [0, 1]$ es un parámetro que determina el peso asignado a cada una en la función compuesta. Para encontrar el valor de α se utilizaron 2 enfoques a partir de la siguiente ecuación:

$$\alpha * OT = (1 - \alpha) * NL \tag{24}$$

En el primer enfoque se toma como valor la mediana de todas las corridas efectuadas por todos los algoritmos obteniéndose α=0.928 . En el segundo se igualan las variaciones observadas para cada propiedad entre el mejor y el peor resultado alcanzado en todas las ejecuciones, llegando así a α=0.952.

Al analizar los resultados los algoritmos búsqueda aleatoria y LaF muestran un peor rendimiento. Los algoritmos híbridos los que alcanzan los mejores resultados en ambas funciones. Confirmando la hipótesis de que combinar la exploración no sesgada de LaF con búsqueda local puede ser efectivo al optimizar problemas combinatorios. Es importante señalar que para cada función objetivo se alcanza el mejor resultado con un híbrido distinto. También, al igual que en el orden de transparencia, el algoritmo genético alcanza muy buenos resultados.

3.4 Discusión

La combinación de ambas propiedades en una sola función objetivo y una selección adecuada del parámetro de ponderación permite a los algoritmos enfocar la búsqueda en aquellas regiones del espacio donde se localizan S-cajas con el correspondiente balance. Los resultados alcanzados evidencian además que ajustando α se puede controlar la preferencia por una u otra propiedad. Los mejores resultados se alcanzaron con los algoritmos híbridos. Estos resultados sugieren que un adecuado balance entre un proceso explícito de exploración y otro de explotación puede ser una estrategia efectiva para encontrar S-cajas de utilidad para uso criptográfico. Hasta el momento se ha asignado al parámetro de transición los valores σ_t = 0,3, 0,5 y 0,7. Sin embargo, estos no tienen que ser necesariamente los valores óptimos para σ_t.

4. Algoritmos híbridos con aprendizaje de máquina

Las heurísticas híbridas propuestas hasta ahora utilizaban un momento fijo para efectuar la transición. El motivo de esta decisión, muy frecuente en el diseño de algoritmo híbridos [**?**], es que determinar el momento adecuado depende de múltiples factores difíciles de medir. Factores que más influyen son:

- **Topología de la función objetivo:** Determinar en tiempo de ejecución la topología y cuánta exploración-explotación debe efectuarse para la misma es complejo, más aún si se tiene en consideración que la topología puede variar según la región del espacio de búsqueda que se explora.
- **Proceso de optimización:** Determinar en tiempo de ejecución si se ha alcanzado el punto de transición es extremadamente complejo pues no resulta computacionalmente factible evaluar de manera explícita los campos de atracción encontrados. Debido al carácter estocástico del proceso de optimización este punto ideal no se alcanza siempre en el mismo momento.

Tomar la decisión correcta, en un momento dado del proceso de optimización, respecto a si efectuar o no la transición es un complejo problema de decisión. Este problema puede ser modelado como un problema de aprendizaje supervisado. El aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento. En nuestro caso se estaría prediciendo el momento idóneo para hacer la transición, por lo que puede ser modelado como un problema de clasificación binario. Los ejemplos entrenantes para un problema de clasificación binario pueden ser representados por un vector. Los componentes $x_1, x_2, ..., x_k$ constituyen datos y la componente y representa la clase.

$$EjemplosEntrenantes : (x_1, x_2, ..., x_k, y) \tag{25}$$

Para cada ejecución del proceso de optimización se establecieron momentos fijos para recopilar los datos y ejecutar la búsqueda local. Se tomaron los siguientes puntos: 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 % y 90 % de las evaluaciones. Los atributos que se utilizaron para caracterizar el proceso de optimización fueron:

1 **Número de evaluaciones realizadas:** cantidad de evaluaciones de la función objetivo que ha realizado LaF.

2 **Número de evaluaciones disponibles:** cantidad de evaluaciones de la función objetivo disponibles.

3 **Número de reinicios realizados:** cantidad de reinicios realizados por LaF.

4-8 **Mediana de los líderes en los últimos 5 reinicios:** mediana de la población de los líderes después de cada uno de los últimos 5 reinicios.

9-13 **Mínimo de los líderes en los últimos 5 reinicios:** mínimo de la población de los líderes después de cada uno de los últimos 5 reinicios.

14-18 **Máximo de los líderes en los últimos 5 reinicios:** máximo de la población de los líderes después de cada uno de los últimos 5 reinicios.

19-23 **Mediana de los seguidores en los últimos 5 reinicios:** mediana de la población de los seguidores antes de cada uno de los últimos 5 reinicios.

24-28 **Mínimo de los seguidores en los últimos 5 reinicios:** mínimo de la población de los seguidores antes de cada uno de los últimos 5 reinicios.

29-33 **Máximo de los seguidores en los últimos 5 reinicios:** máximo de la población de los seguidores antes de cada uno de los últimos 5 reinicios.

34-38 **Número de evaluaciones entre los últimos 5 reinicios:** número de evaluaciones de la función objetivo realizadas entre los últimos 5 reinicios.

39-43 **Número de seguidores aceptados entre los últimos 5 reinicios:** número de seguidores aceptados para la nueva población de líderes después de realizarse la selección por torneo en los últimos 5 reinicios.

Se seleccionaron un total de 43 atributos para caracterizar un momento dado del proceso de optimización. Se hace especial hincapié en los valores alcanzados por los líderes y los seguidores en correlación con los reinicios efectuados. Se hizo uso de dos algoritmos de clasificación, Naive Bayes y Árboles de Decisión:

- **Naive Bayes:** consiste en un clasificador bayesiano simple que asume independencia entre las características.
- **Árboles de Decisión:** los árboles de decisión son clasificadores que aproximan una función a partir de la ejecución de un conjunto de pruebas sobre los valores asociados a atributos predefinidos. Algunas ventajas de los árboles de decisión son: permiten interpretar las reglas utilizando un diagrama de árbol, es un modelo no paramétrico e incorpora fácilmente atributos numéricos y nominales. Una desventaja es que tienden a sobreajustarse a los datos.

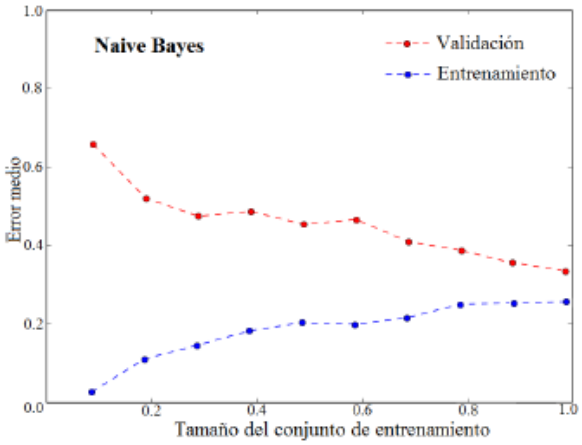
4.1 Resultados

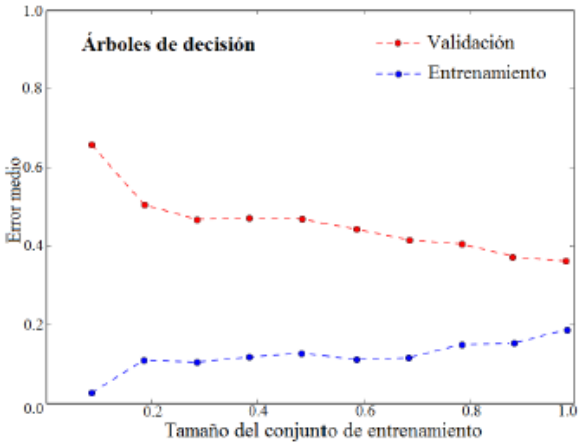
Para estimar la precisión alcanzada por los dos clasificadores se realizaron 30 corridas de validación cruzada con los ejemplos entrenantes obtenidos anteriormente. Se efectuaron 120 ejecuciones del procedimiento. Se tomó como conjunto de entrenamiento un 60 % de los ejemplos y como conjunto de validación el restante 40 %, elegidos de forma aleatoria respetando la distribución de las clases.

Algoritmo	μ	σ
Aleatorio	0,435	0,0288
NaiveBayes	0,694	0,0117
Árboles de Decisión	0,632	0,0121

La tabla muestra los resultados alcanzados por los dos clasificadores propuestos y un clasificador aleatorio tomado como punto de referencia. El valor μ representa la media de la precisión en las 30 corridas y σ la desviación estándar.

Es posible apreciar que el comportamiento ideal ocurre mejor en el caso de Naive Bayes. En el caso del Árbol de





Decisión la convergencia es más lenta y es truncada al final de la gráfica. Esto puede deberse a la inhabilidad del modelo de generalizar bien los datos o que la cantidad de datos de entrenamiento sea insuficiente. Trabajos futuros deberán enfocarse en aumentar el número de ejemplos entrenantes y probar otros modelos de clasificación.

4.2 Hibridación

Entrenado ya el clasificador este se integra dentro del algoritmo híbrido para determinar el momento de transición. Denotaremos estos híbridos como Híbrido-NB e Híbrido-AD según el clasificador que usen.

Algoritmo 3: Algoritmo Híbrido (Híbrido-NB)

Require: maxEvals,Clasificador
1: L ←Inicializar aleatoriamente los líderes
2: S ←Inicializar aleatoriamente los seguidores
3: Transición == *False*
4: **while** evals < maxEvals **or** Transición == *False* **do**
5: lider ←Escoger un lider de L
6: seguidor ←Escoger un seguidor de S
7: Xnueva ←`crearSolución`(lider,seguidor)
8: **if** `f`(Xnueva) < `f`(seguidor) **then**
9: seguidor ←Xnueva
10: **end if**
11: **if** `media`(`f`(S)) < `media`(`f`(L)) **then**
12: L ←`mezclaPoblaciones`(L,S)
13: S ←Iniciar aleatoriamente los seguidores
14: Atributos ←Recopilar Atributos
15: **if** Clasificador(Atributos) == 1 **then**
16: Transición == *True*
17: **end if**
18: **end if**
19: **end while**
20: **for** lider ∈ L **do**
21: **while** evals < maxEvals **do**
22: `BL`(lider, maxEvals)
23: **end while**
24: **end for**

4.3 Resultados

Para evaluar los algoritmos Híbrido-NB e Híbrido-AD se efectuaron 30 ejecuciones independientes tomando como condición de parada 500.000 evaluaciones de la función objetivo.

En ambas tablas que los híbridos con el clasificador alcanzan resultados ligeramente mejores que el resto de los algoritmos. Ambos híbridos son muy estables en los resultados y reportan una baja desviación estándar, menor que los restantes algoritmos híbridos. El híbrido con Naive Bayes reporta para ambas funciones mejor media que el híbrido con árboles de Decisión, sin embargo, se puede notar también que existe una gran similitud en los resultados de ambos híbridos.

4.4 Discusión

En este capítulo se presenta un problema de clasificación, el problema de determinar el momento óptimo para la transición entre LaF y BL. Luego, se entrena los clasificadores de Naive Bayes y árboles de Decisión para dar solución. Cada clasificador presenta evidencia de que ambos clasificadores se comportan de manera distinta. Naive Bayes se observa una mayor precisión y una mayor convergencia en las curvas de aprendizaje. En la clasificación apenas se refleja en los resultados de los algoritmos híbridos. Los resultados alcanzados durante la presente investigación no solo han permitido confirmar de manera positiva las hipótesis planteadas respecto a la hibridación de LaF en problemas combinatorios, sino que también son competitivos con los mejores resultados reportados en la literatura. Un punto de referencia en la optimización de propiedades criptográficas de S-Cajas es la tesis de doctorado de Stepjan Picek culminada en 2015. En esa investigación las dos mejores S-cajas que se reportan de NL 100 tienen un OT de 7,53 y 7.52. No distan mucho de la mejor S-caja reportada en esta investigación para NL 100 y un valor de OT de 7,55. En el caso de NL 98 la mejor S-caja reportada por Picek tiene un OT de 7,36, este valor es peor que la mejor S-caja de 98 encontrada en la presente investigación con OT de tan solo 7,27. Los trabajos futuros deberán explorar la idea de iniciar

los algoritmos de este artículo en regiones cercanas a la S-caja del AES.

5. Conclusiones

La presente investigación ha tenido como propósito aplicar novedosos algoritmos híbridos al problema de diseñar S-cajas con buenas propiedades criptográficas. Estos algoritmos fueron aplicados tanto a las propiedades de manera independiente como a funciones objetivos compuestas que permiten optimizar más de una propiedad simultáneamente y guiar el proceso de búsqueda hacia buenas regiones del espacio. La novedad de los algoritmos híbridos se debe a tres aspectos fundamentales. Primero, a la idea de separar explícitamente la exploración y la explotación. Segundo, la utilización de la recientemente propuesta metaheurística de Líderes y Seguidores para llevar a cabo el proceso de exploración. Tercero, la utilización de técnicas de aprendizaje supervisado para determinar automáticamente el momento de transitar de un proceso a otro. Los resultados alcanzados son competitivos con aquellos reportados en la literatura.

Referencias

[1] Hugo Barbalho, Isabel Rosseti, Simone L Martins, and Alexandre Plastino. A hybrid data mining grasp with path-relinking. *Computers & Operations Research*, 40(12):3159–3173, 2013.

[2] S. Bolufé-Röhler, A. y Chen. Minimum population search - a scalable metaheuristic for multi-modal problems. *Revista Investigación Operacional*, 36(1):85–95, 2015.

[3] Antonio Bolufé-Röhler, Dania Tamayo-Vera, and Stephen Chen. An laf-cmaes hybrid for optimization in multi-modal search spaces. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017.

[4] J. Chen, S. y Montgomery. Particle swarm optimization with thresheld convergence. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 510–516. IEEE, 2013.

[5] Montgomery J. Bolufé-Röhler A. y Gonzalez-Fernandez Y. Chen, S. A review of thresheld convergence. *GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología*, 2015.

[6] John Andrew Clark. *Metaheuristic Search as a Cryptological Tool*. PhD thesis, University of York, 2002.

[7] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[8] Talbi E. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[9] Yasser Gonzalez-Fernandez and Stephen Chen. Leaders and followers—a new metaheuristic to avoid the bias of accumulated information, 2015.

[10] Sylvain Guilley, Philippe Hoogvorst, and Renaud Pacalet. Differential power analysis model and some results. In *Smart Card Research and Advanced Applications Vi*, pages 127–142. Springer, 2004.

[11] Howard M Heys. A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3):189–221, 2002.

[12] Laetitia Jourdan, Clarisse Dhaenens, and El-Ghazali Talbi. Using datamining techniques to help metaheuristics: A short survey. In *International Workshop on Hybrid Metaheuristics*, pages 57–69. Springer, 2006.

[13] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in cryptology—CRYPTO'99*, pages 789–789. Springer, 1999.

[14] Estévez-Velarde-S. Bolufé-Röhler A.-Chen S. y Montgomery J. Piad-Morffis, A. Evolution strategies with thresheld convergence. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2015.

[15] Alexandre Plastino, Richard Fuchshuber, Simone de L Martins, Alex A Freitas, and Said Salhi. A hybrid data mining metaheuristic for the p-median problem. *Statistical Analysis and Data Mining*, 4(3):313–335, 2011.

[16] Emmanuel Prouff. Dpa attacks and s-boxes.

[17] Mark Read. Explicable boolean functions. *Master's thesis, Department of Computer Science, The University of York*, 2007.

[18] Dania Tamayo-Vera, Antonio Bolufé-Röhler, and Stephen Chen. Estimation multivariate normal algorithm with thresheld convergence. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 3425–3432. IEEE, 2016.

Cuadro 1. Resultados para el Orden de Transparencia

Algoritmo	Media	Desviación Estándar	Mejor Valor	Peor Valor
Búsqueda Local	7,63	0,02	7,59	7,67
$LaF-BL-30$	7,61	0,03	7,55	7,66
$LaF-BL-50$	7,63	0,02	7,59	7,66
$LaF-BL-70$	7,67	0,01	7,64	7,70
LaF	7,72	0,01	7,70	7,74
Algoritmo Genético	7,20	0,33	6,90	7,77
Búsqueda Aleatoria	7,79	0,01	7,76	7,82

Cuadro 2. Resultados para la No Linealidad

Algoritmo	Media	Desviación Estándar	Mejor Valor	Peor Valor
Búsqueda Local	98,80	0,98	98	96
$LaF-BL-30$	98,53	0,72	100	98
$LaF-BL-50$	98,40	0,57	100	98
$LaF-BL-70$	98,46	0,66	100	98
LaF	98,33	0,39	100	98
Algoritmo Genético	98,13	0,20	100	98
Búsqueda Aleatoria	93,47	1,71	96	90

Cuadro 3. Resultados para la función compuesta ($\alpha = 0.928$)

Algoritmo	Media	Desviación Estándar	Mejor Valor	Peor Valor
Búsqueda Local	$7{,}54e-02$	$7{,}04e-03$	$6{,}05e-02$	$8{,}48e-02$
$LaF-BL-30$	$-1{,}40e-01$	$2{,}60e-02$	$-1{,}79e-01$	$-1{,}16e-01$
$LaF-BL-50$	$7{,}63e-02$	$3{,}61e-02$	$-4{,}35e-02$	$9{,}94e-02$
$LaF-BL-70$	$9{,}12e-02$	$4{,}42e-02$	$-2{,}07e-02$	$1{,}16e-01$
LaF	$1{,}53e-01$	$4{,}20e-03$	$1{,}44e-01$	$1{,}60e-01$
Algoritmo Genético	$6{,}23e-02$	$3{,}36e-02$	$-5{,}64e-0{,}2$	$8{,}48e-02$
Búsqueda Aleatoria	$5{,}76e-01$	$1{,}85e-01$	$3{,}20e-01$	$7{,}72e-01$

Cuadro 4. Resultados para la función compuesta ($\alpha = 0.952$)

Algoritmo	Media	Desviación Estándar	Mejor Valor	Peor Valor
Búsqueda Local	2,64	$2{,}43e-02$	2,59	2,69
$LaF-BL-30$	2,42	$2{,}33e-02$	2,36	2,46
$LaF-BL-50$	2,38	$2{,}25e-02$	2,32	2,48
$LaF-BL-70$	2,62	$2{,}26e-02$	2,55	2,63
LaF	2,69	$3{,}34e-03$	2,68	2,70
Algoritmo Genético	2,60	$2{,}40e-02$	2,53	2,62
Búsqueda Aleatoria	2,93	$6{,}64e-02$	2,82	3,01

Cuadro 5. Resultados para la función compuesta ($\alpha = 0.928$)

Algoritmo	Media	Desviación Estándar	Mejor Valor	Peor Valor
$LaF-BL-30$	$-1{,}40e-01$	$2{,}60e-02$	$-1{,}79e-01$	$-1{,}16e-01$
$LaF-BL-50$	$7{,}63e-02$	$3{,}61e-02$	$-4{,}35e-02$	$9{,}94e-02$
$LaF-BL-70$	$9{,}12e-02$	$4{,}42e-02$	$-2{,}07e-02$	$1{,}16e-01$
LaF	$1{,}53e-01$	$4{,}20e-03$	$1{,}44e-01$	$1{,}60e-01$
Híbrido-NB	$-1{,}55e-01$	$4{,}01e-03$	$-1{,}60e-01$	$-1{,}49e-01$
Híbrido-AD	$-1{,}523-01$	$3{,}16e-03$	$-1{,}59e-01$	$-1{,}50e-01$

Cuadro 6. Resultados para la función compuesta ($\alpha = 0.928$)

Algoritmo	Media	Desviación Estándar	Mejor Valor	Peor Valor
$LaF-BL-30$	2,42	$2{,}33e-02$	2,36	2,46
$LaF-BL-50$	2,38	$2{,}25e-02$	2,32	2,48
$LaF-BL-70$	2,62	$2{,}26e-02$	2,55	2,63
LaF	2,69	$3{,}34e-03$	2,68	2,70
Híbrido-NB	2,29	$3{,}36e-03$	2,27	2,35
Híbrido-AD	2,32	$3{,}40e-03$	2,27	2,36