Utilización de algoritmos de enjambre para resolver problemas critpoaritméticos.

María Laura Acuña\*, Marcelo Espinoza\*, Cecilia María Luciana Gómez\*

\*Alumnos de 5to año de la carrera ingeniería en sistemas de información, Facultad Regional Resistencia, Universidad Tecnológica Nacional – French 414 – Resistencia, Chaco, Argentina. CP 3500.

{Marilau\_ml, marceloespinoza00, [Cecilia.mlgz}@gmail.com](mailto:Cecilia.mlgz%7d@gmail.com)

**Abstract.** En el siguiente paper se demuestra una forma de resolver un problema criptoaritmeticos de suma o restas, utilizando para ello algoritmo de enjambre, específicamente el algoritmo firefly desarrollado por Yang [1], tal algoritmo fue adaptado para que se ajuste al problema planteado. La resta es una adaptación de la suma, será realizada intercambiando los operandos de entrada y resultados.

**Keywords:** Algoritmos genéticos, criptoarimética, algoritmos de enjambre, firefly

1. Introducción

Los problemas criptoaritmeticos son puzles donde las letras son reemplazadas por número, son problemas de restricciones (Mounier – Aguirre – Barboza) [2], que si se intenta resolver manualmente genera un gran espacio de búsqueda, dificultando así la generación de la solución. Existen varias alternativas para encontrar la solución, en (Mounier – Aguirre – Barboza) [2] se plantea una solución utilizando algoritmos genéticos con operaciones de cruza y mutación, en Ishaque y otros [3] se plantea una solución utilizando también algoritmos genéticos, con operaciones de mutación, y se demuestra la mejora en performance respecto a la búsqueda heurística. En SARAEI [6] se describe cómo solucionar el problema del viajantes aplicando el algoritmo propuesto por Yang [1]. Por tales motivos se plantea una solución implementando, con algunas adaptaciones el algoritmo planteado por Yang [1], redefiniendo una función distancia y la función acercamiento.

En la sección 2 se presenta una introducción de los algoritmos de enjambre, especificando el algoritmo firefly. En la sección 3 se define el problema criptoaritmético. En la sección 4 se describe el modelo planteado. En la sección 5 se muestran los casos de prueba y los resultados obtenidos. Finalmente, en la sección 6 las conclusiones del trabajo.

* 1. Problema Planteado

El problema consiste en resolver puzles criptoaritmeticos, de sumas o restas, con la implementación del algoritmo firefly [1].

Este problema presenta las restricciones siguientes:

* + La cantidad distintas de letras de los operandos a sumar (o restar) no debe ser mayor a 10.
  + Cada letra se identifica con un único número y ese número representa a una única letra.
  + El resultado deber ser acorde a la suma algebraica de los operandos.

El problema presenta la complejidad de, como es una función discreta, resolverlo mediante la utilización de un algoritmo para encontrar máximos locales en funciones continuas.

1. ¿Qué son los algoritmos de enjambre?

Los algoritmos de enjambre simulan el comportamiento que tienen ciertas especies que se organizan grupalmente para subsistir. Algunos de los algoritmos de enjambres más conocidos son “La optimización de colonias de hormigas” – Dorigo [8], “La optimización de colonias de abejas” – De los Cobos [9], y entre estos tipos de algoritmos se encuentra el desarrollado por Yang [1].

La inteligencia de enjambre estudia el comportamiento colectivo compuestos por muchos individuos interactuando localmente y con su entorno – Molina [5].

En un sistema de optimización por enjambre de partículas, la búsqueda se realiza utilizando una población de partículas que corresponden a los individuos, cada uno de los cuales representa una solución candidata al problema. Las partículas cambian su estado al moverse a través del espacio de búsqueda hasta que se ha encontrado un estado relativamente estable, la solución.

* 1. Algoritmo de luciérnagas

El algoritmo luciérnaga (firefly en inglés) conocido como FA, está basado en el comportamiento de las luciérnagas en la naturaleza.

El algoritmo se basa entre reglas básicas:

* + - Todas las luciérnagas son unisexuales y se sienten atraídas por otras luciérnagas, independientemente de su sexo.
    - El grado de atracción de una luciérnaga es proporcional a su brillo, y por lo tanto para cualquier par de luciérnagas, la que menos brillo tiene se moverá hacia la que más brillo tiene.
    - El brillo depende de la distancia entre ambas. En el caso de que las dos luciérnagas tengan el mismo brillo, se moverán aleatoriamente.

El brillo de una luciérnaga se determina por el valor de la función objetivo.

1. Funcionamiento del Algoritmo desarrollado

Los valores de entrada se colocan en un vector que sirve como base para el desarrollo de todo el algoritmo. Por ejemplo si se ingresa APPLE + LEMON = BANANA, el vector de inicio conformado, luego de acomodarlo y quitarle los elementos repetidos (se completa con “-“ hasta las diez elementos dentro del vector), queda:

LETRAS DE ENTRADA: e n a l o p m b - -

POSICION: 0 1 2 3 4 5 6 7 8 9

Las posiciones de las letras sirven para controlar las posibles soluciones.

Luego, se crean una matriz de vectores de diez números aleatorios cada uno, la población inicial. Cada vector representa una luciérnaga, la cual se va adaptando para llegar a ser una solución.

Por cada luciérnaga, se extraen los valores, APPLE, LEMON y BANANA, que corresponden a su posición en el vector de inicio, y se calcula la suma algebraica entre APPLE y LEMON, el brillo de la misma, calculado entre la suma algebriaca y la correspondencia del vector resultado en el vector inicio, cuanto más alto sea el brillo, más cerca se encuentra de ser una solución. Dado una luciérnaga cuyo vector esg:

0 6 5 2 8 1 3 4 9 7

e n a l o p n b - -

Apple = 5 1 1 2 0

Lemon = 2 0 6 8 6

Banana (correspondencia en el vector inicio)= 7 2 1 2 1 2

Banana (Suma algebraica) = 7 1 8 0 6

Brillo = 0

* 1. La función objetivo

Se definió la función objetivo como “obtenerBrillo” la que recibe como parámetro la suma algebraica de los valores que corresponden a los operadores de entrada, y el resultado que corresponden a los valores del operando ingresado como resultado.

**Algoritmo 1: Función objetivo**

Public function obtenerBrillo( $suma, $resultado)

{ $res=$this->intToArray($resultado);

$sum=0;

$i=(sizeof($res)-1);

$j=(sizeof($suma)-1);

$counter=0;

if(sizeof($res)<sizeof($suma)){

$counter=sizeof($res);

}else{

$counter=sizeof($suma);

}

for ($k = ($counter-1); $k >=0; $k--) {

if($res[($i)]==$suma[($j)]){

$i--;

$j--;

$sum++;

}

}

return $sum;

Su funcionamiento consiste en devolver un valor de brillo que depende de la cantidad de elementos que tenga el vector de resultado ingresado, y las coincidencias que tenga con el vector suma, comparando posición a posición..

**Fig. .**Inicio del algoritmo planteado

La función distancia definida en la figura 3, recibe como parámetros dos vectores que corresponden a la suma algebraica de dos luciérnagas diferentes y devuelve un valor entre uno y diez.

**Algoritmo 2: Función Distancia**

public function distancia($X1, $X2) {

$i = count($X1);

$d = 0;

for ($j=0; $j < $i; $j++) {

if ($X1[$j] = $X2[$j]) {

$d = $d + 10\*$j;

}

}

$d = fmod($d, 10);

return $d;

}

**Fig. .**Desarrollo de la solución

1. Resultados
   1. Casos de prueba

|  |  |  |
| --- | --- | --- |
| **ENTRADAS** | **Cantidad de soluciones** | **Cantidad de caracteres distintos** |
| AB + CD = ABD | MUCHAS | 4 |
| DOS+DOS=TRES | MUCHAS | 6 |
| TWO+TWO=FOUR | MUCHAS | 6 |
| CAT + DOG = PETS | MUCHAS | 9 |
| SEND+MORE=MONEY | UNA | 8 |
| APPLE+LEMON=BANANA | UNA | 8 |
| PLAYS+WELL+BETTER | UNA | 10 |

* 1. Resultados obtenidos

Para 5 luciérnagas o vectoresdistintos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ENTRADAS** | **Corridas** | **Cant. de corridas con solución** | **Promedio Iteraciones** | **Resultado Solución** |
| AB + CD = ABD | 10 | 10 | 6 | 10 + 92 = 102 |
| DOS+DOS=TRES | 10 | 10 | 13 | 860+860=1720 |
| TWO+TWO=FOUR | 10 | 10 | 16 | 938+938=1876 |
| CAT + DOG = PETS | 10 | 10 | 53 | 846+723=1569 |
| SEND+MORE=MONEY | 10 | 2 | 175 | 9567+1085=10652 |
| APPLE+LEMON=BANANA | 10 | 1 | 191 | 67794+94832=162626 |
| PLAYS+WELL=BETTER | 10 | 0 | 200 | NO ENCONTRÓ |

Para 10 luciérnagas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ENTRADAS** | **Corridas** | **Cant. de corridas con solución** | **Promedio Iteraciones** | **Resultado Solución** |
| AB + CD = ABD | 10 | 10 | 1 | 10 + 96 = 106 |
| DOS+DOS=TRES | 10 | 10 | 2 | 780+780=1560 |
| TWO+TWO=FOUR | 10 | 10 | 3 | 928+928=1856 |
| CAT + DOG = PETS | 10 | 10 | 17 | 765+293=1058 |
| SEND+MORE=MONEY | 10 | 2 | 173 | 9567+1085=10652 |
| APPLE+LEMON=BANANA | 10 | 4 | 167 | 67794+94832=162626 |
| PLAYS+WELL=BETTER | 10 | 1 | 184 | 97426+8077=105503 |

1. Conclusión

Los algoritmos de enjambre son utilizados no solamente para resolver problemas que se presentan en la naturaleza, sino que tienen múltiples aplicaciones. En este trabajo, nos centramos en los problemas criptoaritméticos y agregamos restricciones para su resolución. Específicamente, nos basamos en el algoritmo firefly debido a su generalidad. Demostramos que la cantidad de luciérnagas a considerar es un factor importante a la hora de resolver este tipo de problemas.

Además, la eficacia de los algoritmos de enjambre, concretamente del algoritmo firefly ha sido demostrada por la resolución del problema planteado.

Referencias

1. X.-S. Yang, Firefly algorithms for multimodal optimization, in: Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computer Sciences, Vol. 5792, pp. 169-178 - SAGA 2009.
2. M. Mounier, F. Aguirre1, M. Barboza: Resolución de Problemas Criptoaritméticos Utilizando Algoritmos Genéticos 43 JAIIO - EST 2014 - ISSN: 1850-2946 17º Concurso de Trabajos Estudiantiles. – 2014.
3. A. S. Md. Ishaque, Md. BahlulHaider, M. Al M. Wasid, S. M. Alaul, Md. Kamrul Hassan, T. Ahsan, M. Sh. Alam: Evolutionary Algorithm to Solve Cryptarithmetic Problem Transactions on engineering, computing and technology VI – World Enformatika Society Decembre 2004.
4. John H. Holland: Algoritmos genéticos. Investigación y Ciencia 1992.
5. S. Molina; M. F. Piccoli; G. Leguizamón: Algoritmos de inteligencia de enjambres sobre GPU: una revisión exhaustiva. XX Congreso Argentino de Ciencias de la Computación - Buenos Aires, 2014.
6. M. SARAEI, R. ANALOUEI, P. MANSOURI: Solving of Travelling SalesmanProblemusingFireflyAlgorithmwithGreedyApproach. CumhuriyetUniversityFaculty of Science. Science Journal (CSJ), Vol. 36, No: 6 Special Issue (2015).
7. Fateen y Bonilla-Petriciolet: IntelligentFireflyAllgorithmfor Global Optimization. X.-S. Yang (ed.), *CuckooSearch and FireflyAlgorithm*, Studies in ComputationalIntelligence. Springer International Publishing Switzerland 2014.
8. Dorigo, Marco, and Christian Blum. "Ant colony optimization theory: A survey." *Theoretical computer science* 344.2 (2005): 243-278.
9. De los Cobos Silva, Sergio Gerardo, et al. "Colonia de abejas artificiales y optimización por enjambre de partículas para la estimación de parámetros de regresión no lineal." *Revista de Matemática: Teoría y Aplicaciones* 21.1 (2014): 107-126.