Cómo resolver el juego *Mastermind* a través de computación evolutiva

Lizzeth Cabana Tejada
Ciencia de Computación
Universidad Católica San Pablo
Arequipa, Perú
lizzeth.cabana@ucsp.pe

Yván Jesús Túpac Valdivia
Ciencia de Computación
Universidad Católica San Pablo
Arequipa, Perú
ytupac@ucsp.pe

***Abstract*—En este trabajo se modela e implementa una estrategia de solución del juego "*Mastermind*" que emplea un algoritmo evolutivo discreto con aplicación de algunos operadores genéticos discretos, operadores de orden y operadores personalizados. Se presenta el modelo del juego, la estrategia evolutiva planteada y también los resultados obtenidos, los cuales evidencian que el modelo evolutivo consigue solucionar el juego Mastermind en una cantidad de intetntos bastante razonable.**

***Keywords*-Mastermind, algoritmos evolutivos discretos, operadores de orden**

***Abstract*—This paper models and implements a strategy for solving the *Mastermind* game using a discrete evolutionary algorithm applying some discrete genetic operators, order-based operators and custom operators. We present a model of the game, the evolutionary strategy proposed addition to the results obtained, which evidence that the evolutionary model solves the Mastermind game in a quite reasonable number of attempts.**

***Keywords*-Mastermind, discrete evolutionary algorithms, order-based operators**

I. Introducción

Mastermind©, o *Master Mind* es un juego de mesa para dos personas bastante conocido inventado en la década de 1970 por Mordecai Meirowitz y cuyos derechos intelectuales hoy en día son de propiedad de *Pressman Toy Corporation*. Este juego recuerda a un antiguo juego de lápiz y papel denominado *bulls and cows* con más de un siglo de existencia. La Figura 1 nos muestra la carátula de una de las primeras versiones vendidas en América del Norte en 1972.

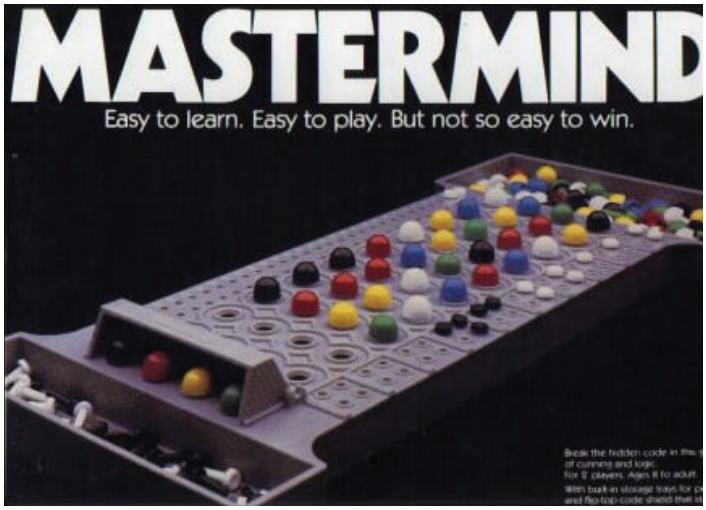

Fig. 1. Edición antigua de *Mastermind* vendidas en 1972

Mastermind se juega en una mesa usando un conjunto $\mathscr{S}_{bw}$ de fichas blancas y negras, y un segundo conjunto $\mathscr{S}_c$ de fichas de N colores que son un poco más grandes. El juego tiene la siguiente lógica:

1) El jugador 1 (el desafiante) escoge L fichas de colores ($L = 4$ en la versión original) del conjunto $\mathscr{S}_c$ y genera una combinación que no es visible al segundo jugador.
2) El jugador 2 (el desafiado) retira algunas piezas de colores del conjunto $\mathscr{S}_c$ e intenta adivinar la combinación oculta.
3) El desafiante genera un *feedback* con las fichas blancas y negras haciendo una fila de 0 a L fichas de acuerdo a la siguiente regla:
   - *una ficha blanca* por cada color adivinado en lugar equivocado.
   - *una ficha negra* por cada color adivinado en lugar correcto. La posición de la ficha negra no tiene relación con la verdadera posición del color adivinado.

El juego finaliza cuando se descubre la combinación secreta, (*obtención* de L fichas negras) o cuando se alcanza un número máximo de intentos t_{max} sin adivinar. Es común que $t_{max} = 15$ [1].

De acuerdo a lo descrito, se observa que *Mastermind* es un juego relativamente simple; sin embargo, muchos investigadores tomaron interés en solucionarlo [2] usando técnicas de Inteligencia Artificial. Así, este juego es un problema de optimización combinatoria que puede ser solucionado mediante un algoritmo de búsqueda.

Este trabajo apunta a mostrar que, aplicando un modelo de algoritmo evolutivo discreto con operadores genéticos convencionales, operadores de orden y algunos operadores personalizados, es posible alcanzar una estrategia que encuentre una solución óptima para el juego *Mastermind*.

A. Organización del artículo

Este artículo está organizado de la siguiente manera: En la sección 2 se muestran algunos trabajos que implementan estrategias *Mastermind*, en la sección 3 se tratan los Algoritmos

Evolutivos Discretos y de orden, en la sección 4 se detalla el modelo evolutivo propuesto para el juego *Mastermind*, en la sección 5 se muestran los experimentos realizados y los resultados obtenidos y la sección 6 expresa las conclusiones de este trabajo.

II. Estrategias Mastermind

En la literatura se encuentran trabajos que tratan de resolver el problema del *Mastermind* usando diferentes técnicas, que se pueden categorizar como:

Unos trabajos lanzan adivinanzas estratégicas, con la intención de extraer información y reducir el tamaño del espacio de búsqueda. Knuth [1], alcanza una respuesta, para $L = 4$ y $N = 6$, construyendo un conjunto de patrones de prueba (adivinanzas) y se elige uno de ellos en cada jugada, tal que se minimice el número máximo de posibilidades restantes. Irving [3], basado en el trabajo de Knuth, minimiza el número esperado de posibilidades restantes.

Kooi [4] propone la "Estrategia de muchas partes", la cual dice que, mientras el código a adivinar tenga mayor número de respuestas posibles (fichas negras y blancas) será mejor, ya que así se puede llegar a la solución más pronto. Bestavros [5] utiliza las estratégias "MaxMin y MaxEnt", en las que el decodificador elige la adivinanza que ofrece la máxima cantidad de información. La ventaja de estos algoritmos es que tratan de minimizar el espacio de búsqueda, pero por otro lado se requiere más tiempo computacional para realizar dichas estrategias.

Otros autores se basan en la búsqueda exhaustiva, como Koyoma [6] con $L = 4$ y $N = 6$, donde, para una adivinanza, se verifican todos los códigos posibles manteniendo el mejor encontrado (en fichas negras y blancas). Chen [7] generalizó la estrategia de Koyoma [6] para $L = n$ y $N = m$. Estos algoritmos demandan un gran tiempo de ejecución, puesto que, es necesario enumerar todos los posibles códigos.

Se encuentran trabajos que aplican búsqueda aleatoria. Shapiro [8] propone un algoritmo simple, donde el decodificador crea una lista de posibles adivinanzas en un orden aleatorio y las va probando una por una. Swaszek [9], ordena la lista creada por el decodificador en función al color. En el trabajo de Rosu [10] se generan combinaciónes aleatorias que son usadas sucesivamente en adivinanzas hasta conseguir una combinación que coincida con el código secreto. Una gran desventaja de estos modelos es tomar en cuenta la información dada por las adivinanzas previas.

Finalmente, otros autores proponen usar métodos de búsqueda dirigidos como los algoritmos genéticos con un valor de aptitud para elegir una solución. La ventaja de estas técnicas es que se adaptan al juego *Mastermind* el cual, al ser un problema de optimización combinatoria, se puede solucionar mediante un algoritmo evolutivo discreto. Así, Bento [11] utilizó $L = 5$ y $N = 8$, guardando la mejor combinación de cada generación, cada combinación tenía asociada una aptitud dada por la diferencia entre la cantidad de fichas blancas y negras.

El trabajo de Berghman [2] tiene como idea central la creación de información que será almacenada a lo largo de las diferentes generaciones del modelo genético y las siguientes jugadas se determinan basadas en la información obtenida anteriormente. Singley [12], empieza con un número de adivinanzas arregladas para descubrir el número de ocurrencias de cada color en el código secreto y luego continúa con un algoritmo de búsqueda tabú. El fitness de una combinación es una función que responde a las adivinanzas previas. La búsqueda de una nueva adivinanza se hace a través de la última adivinanza jugada, en la cual dos posiciones son escogidas aleatoriamente, intercambiadas y se agrega a la lista tabú. El fitness de la nueva combinación es comparado al mejor fitness obtenido hasta el momento y reemplaza al último cuando el nuevo fitness es mejor. Merelo [13], sólo incluye una determinada combinación dentro de una población, si cumple que la distancia a la combinación secreta es mínima. Cada adivinanza se obtiene mediante el cálculo de proximidad a conseguir las cuatro fichas negras como respuesta del codificador.

III. Algoritmos Evolutivos Discretos

Son algoritmos evolutivos caracterizados por que sus individuos pertenecen a un espacio de problema $\mathbb{X}$ discreto y finito que puede ser generado por alguna de las siguientes formas:

1) Por el producto cartesiano $\{x_1,\ldots,x_n\}$ de un subconjunto de n valores $N \subset \mathbb{N} = \{1,\ldots,n\}$ denominados alfabeto.
2) Por el conjunto de permutaciones $\{\phi(1),\ldots,\phi(n)\}$ de n valores en $N \subset \mathbb{N}$, $\phi(i) \in N, \forall i = 1\ldots n$, $\phi(i) \neq \phi(j), \forall i \neq j$.

Así, se puede definir un modelo de algoritmos evolutivos discretos de la siguiente forma:

Definición 1. Algoritmo genético discreto: *Es el modelo de algoritmo evolutivo que se basa en una población* $\mathbf{X}_t = \{x_i\}_{i=1}^m$ *cuyos individuos* $\mathbf{x}_i \in \mathbf{X}_t$ *están compuesto por un conjunto de genes* $\mathbf{x} = (x_1,\ldots,x_n)$ *donde cada* x_j *puede tomar alguno de los valores del alfabeto* $N = \{1,\ldots n\} \subset \mathbb{N}$.

En este modelo, el espacio de problema $\mathbb{X}$ es generado como:

$$\mathbb{X} = N \times N \times \ldots \times N \subset \mathbb{N}^n \tag{1}$$

A partir de esta definición se puede llegar a una definición más generalizada:

Definición 2. Algoritmo evolutivo discreto general *Es el algoritmo evolutivo en el que para un individuo* $\mathbf{x} \in \mathbf{X}_t$*, cada gene* $x_k \in \mathbf{x}$ *puede tomar valores de un alfabeto propio* $N_k = \{i\}_{i=1}^{n_k}$ *y el espacio de problema* $\mathbb{X}$ *quedará definido por:*

$$\mathbb{X} \equiv N_1 \times N_2 \times \ldots \times N_n \subset \mathbb{N}^n \tag{2}$$

Los operadores genéticos del modelo evolutivo discreto o discreto generalizado resultan siendo una extensión de los operadores que se aplican en el algoritmo genético canónico [14], [15], tales como el cruce de 1 punto, cruce de 2 puntos,

cruce uniforme y las mutación de 1 bit. La figura 2 ilustra un cruce de 2 puntos empleado en este trabajo.

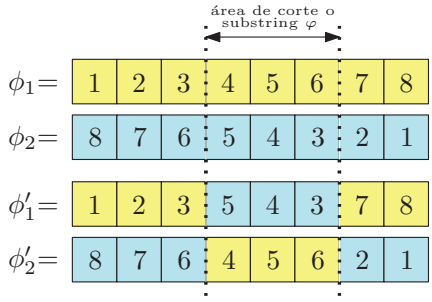

Fig. 2. Crossover de 2 puntos

A. Algoritmos Evolutivos de Orden

Un caso particular de los algoritmos evolutivos discretos es el modelo de algoritmos evolutivos de orden en el cual, el espacio de problema $\mathbb{X}$ es generado por el conjunto de permutaciones de un alfabeto $N = \{1,\ldots,n\}$.

Un modelo evolutivo de orden se define como:

- Sea $\Phi_t = \{\phi_i\}_{i=1}^m$ una población de m individuos en la generación t.
- Cada individuo $\phi_i \in \Phi_t$ está compuesto por un conjunto de valores $\phi_i = \{\phi_i(1),\ldots,\phi_i(n)\}$, donde $\phi_i(j) \in N = \{1,2,\ldots,n\}$.
- Se debe cumplir $\phi_i(p) \neq \phi_i(q) \Leftrightarrow p \neq q$
- Para dos individuos ϕ_i y ϕ_j, con $i \neq j$ no se cumplirá necesariamente que $\phi_a(p) = \phi_b(p)$.

Un modelo de orden requiere operadores especializados en mantener la característica de permutaciones de los individuos $\phi \in \Phi$, como los siguientes:

1) **Mutación de Desorden** que desordena una sublista seleccionada del individuo, como se muestra en la figura 3.

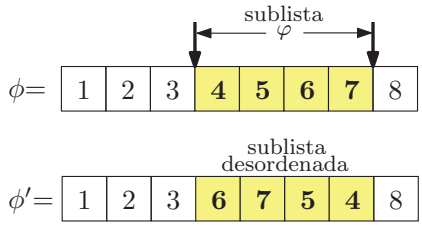

Fig. 3. Operador Mutación de Desorden.

2) **Mutación Basada en Posición** que intercambia las posiciones de dos genes seleccionados aleatoriamente, como se muestra en la figura 4.

Es interesante resaltar que los operadores de orden pueden aplicarse en modelos con algoritmos evolutivos discretos sin ningún problema como en este trabajo.

IV. Modelo Evolutivo para el Mastermind

El problema del juego *MasterMind* se puede formular como la búsqueda de un código dirigido por pistas dadas por una caja negra. El código secreto y las adivinanzas son cadenas de longitud L extraídos de un alfabeto con cardinalidad N que es la cantidad de colores disponibles, por lo tanto, el tamaño del espacio de búsqueda es N^L . El juego tiene dos variantes: "clásico", con $N = 6$, $L = 4$ y "super MasterMind" con $N = 8$, $L = 5$. En el primer caso, el tamaño del espacio de búsqueda es 1296 y en el segundo 32768 [13]

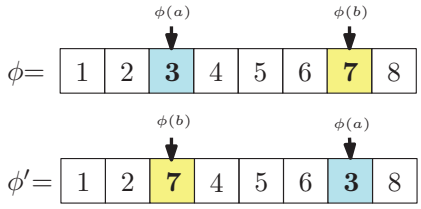

Fig. 4. Mutación Basada en Posición.

En el trabajo de Rodriguez [16], se indica que las fichas negras y blancas del juego limitan el espacio de posibles soluciones ayudando a que el decodificador encuentre la solución. Un ejemplo de esto se ilustra en la Figura 5.

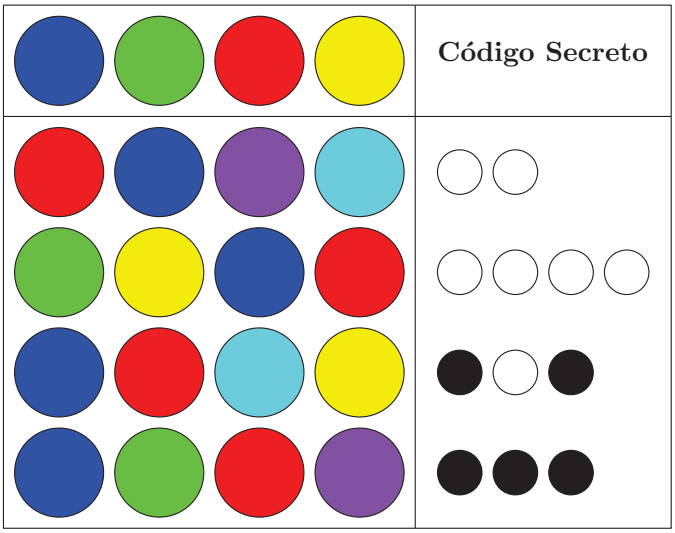

Fig. 5. Ejemplo del *Mastermind* con una combinación secreta de tamaño 4 y ejemplos de adivinanzas con sus *feedbacks* dados por el desafiante.

A. Representación

Dado que se cuenta con un alfabeto $N = \{1,2,\ldots,n\}$ de los n colores disponibles, un individuo $\mathbf{x} = \{x_1,\ldots,x_n\}$ representa una posible adivinanza, la longitud del individuo es L. En la figura 6 se muestra un ejemplo para $N = 6$ y $L = 4$.

B. Función de aptitud y método de selección

En este trabajo se toma como referencia la función desarrollada en el trabajo de Kalister [17], donde se emplea un Algoritmo Genético con la característica clásica de reproducción proporcional a la aptitud [18] que también fue utilizado en el trabajo de Merelo [13].

Para una combinación $\mathbf{x}_i$ propuesta, la aptitud consiste en observar las fichas negras y blancas de esta combinación y

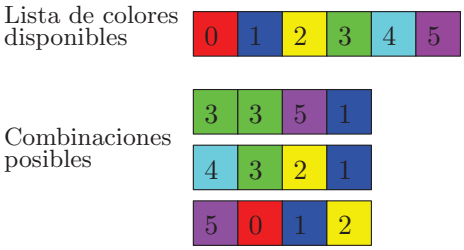

Fig. 6. Representación de cromosomas.

luego aplicar la siguiente expresión:

$$f(\mathbf{x}_i) = (2B+W) + \sum_{k=1}^{N-1} k \tag{3}$$

donde:

N	Total de fichas negras y blancas $(B+W)$
B	Total de fichas negras.
W	Total de fichas blancas.

Esta aptitud premia a la combinación $\mathbf{x}_i$ por la cantidad de aciertos, por ejemplo, para combinaciones con $L=4$, la aptitud sería $f(\mathbf{x}_i)=14$ si se acertó totalmente la combinación secreta y sería 0 si no se acertó ninguna posición ni color. La Figura 7 ilustra algunos ejemplos de la aplicación de la función de aptitud $f(\mathbf{x}_i)$. La estrategia de selección es la ruleta.

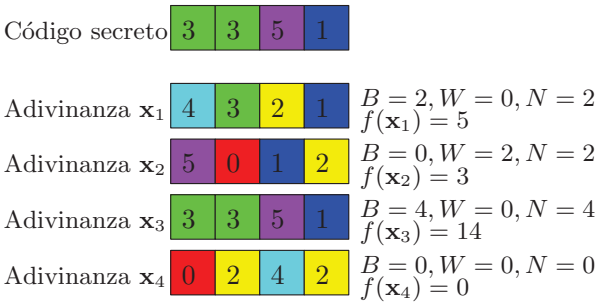

Fig. 7. En la fila superior, un código secreto de tamaño 4, en las filas siguientes, posibles adivinanzas con el cálculo de sus respectivos Scores o fitness según la función *fitness* de Kalister [17].

C. Operadores

Se utilizaron operadores genéticos que pertenecen a modelos evolutivos discretos, modelos evolutivos de orden y un operador personalizado para este problema como se describe a continuación:

- Cruce: Se usó el cruce de dos puntos ilustrado en la figura 2.
- Mutaciones: Se usó la Mutación de Desorden de la figura 3, Mutación de Posición de la figura 4 y además la Mutación Mastermind, que es un operador personalizado para el problema *Mastermind*. Este operador consiste en eligir un individuo $\mathbf{x}_i \in \mathbf{X}_t$ aleatoriamente y seleccionar un gen $x_j \in \mathbf{x}_i$ cambiándolo por el próximo color en la lista de colores disponibles del juego, y si el color del gen coincide con el último color de la lista, se cambiará por el primer color disponible. La figura 8 ilustra este operador.
- Elitismo
  Se toma el mejor individuo de cada generación y se preserva para las siguientes generaciones.

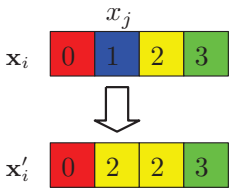

Fig. 8. Ejemplo de la mutación *Mastermind*.

V. Experimentos y Resultados

Durante la realización de las pruebas, que consistieron en ejecutar el modelo evolutivo varias veces obteniendo las siguientes curvas que permiten observar mejor el comportamiento de nuestro modelo:

a. Curva *best-so-far*
Para cada experimento, guarda las aptitudes de los mejores elementos $x_{best}(t)$ por cada generación t obteniéndose vectores de mejores aptitudes por cada experimento. Habiendo más de un experimento, calcular la media aritmética de los vectores y graficar la curva promediada.

b. Curva *off-line*
Una curva *off-line* permite, para un experimento, observar la obtención de buenas soluciones sin importar el tiempo tomado para encontrarlas. En un experimento, por cada generación t se toma el valor:

$$v_{off}(t) = \frac{1}{t+1}\sum_{t=0}^{t} f(x_{best}(t)) \tag{4}$$

donde $f(x_{best}(t))$ es la evaluación del mejor elemento de la población $\mathbf{X}_t$. Esta curva grafica la media de los mejores individuos hallados hasta la generación t. En caso de varios experimentos, también se calculan la media aritmética de los vectores resultantes y se grafica este promedio

c. Curva *on-line*
Permite ver la rápida obtención de buenas respuestas y también la convergencia de los individuos de la población. En un experimento, por cada generación se toma el valor:

$$v_{on}(t) = \frac{1}{t+1}\sum_{t=0}^{t} \overline{f}(\mathbf{X}_t) \tag{5}$$

donde $\overline{f}(\mathbf{X}_t)$ es la media de las evaluaciones de todos los individuos de la población $\mathbf{X}_t$. Esta curva grafica la media de todos los individuos hasta la actual generación.

Es importante aclarar que para cada evaluación se hicieron en total 20 pruebas consecutivas, calculando el promedio de los resultados de las curvas de los mejores elementos, curva offline y online.

A continuación se mostrarán dos diferentes parametrizaciones que se tomaron en cuenta para la realización de las pruebas:

A. *Parametrización 1*

Se usaron los parámetros de la Tabla I: En esta parametrización se aplica siempre el operador de cruce se aplica siempre en cada generación.

TABLE I
PARAMETRIZACIÓN 01

Parámetro	Valor
Número de generaciones	150
Modo de juego *Mastermind*	Clásico
Colores disponibles (*N*)	6
Tamaño de la cadena (*L*)	4
Individuos en una población	40
Aptitud máxima	14

Aplicando el operador de Mutación de Desorden en un 30%, el operador de Mutación de Posición en un 30% y finalmente el operador de Mutación Mastermind en un 40%, se logra alcanzar casi siempre el fitness 14, los resultados se muestran en la figura 9.

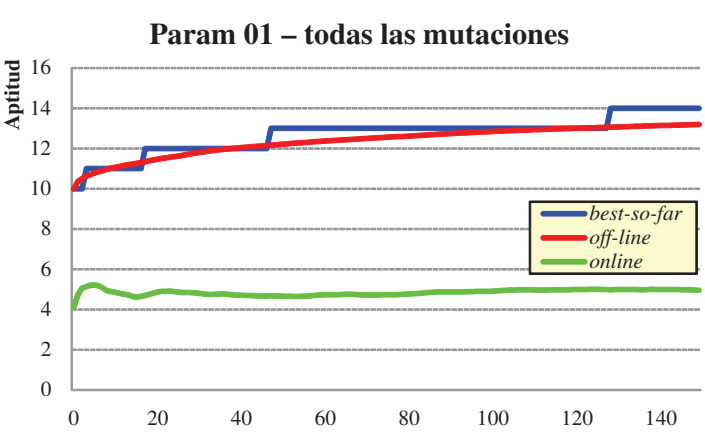

Fig. 9. Curva de los mejores elementos, curva *offline* y *online* aplicando Parametrización 1.

B. *Parametrización 2*

Se usaron los parámetros de la tabla II:

TABLE II
PARAMETRIZACIÓN 02

Parámetro	Valor
Número de generaciones	150
Modo de juego *Mastermind*	*Super Mastermind*
Colores disponibles (*N*)	8
Tamaño de la cadena (*L*)	5
Individuos en una población	40
Aptitud máxima	20

Para esta parametrización se formularon los siguientes tres experimentos intentando determinar qué tanto contribuye cada operador de Mutacion en la solución del problema.

a) Aplicando sólo el operador de Mutación de Desorden, el algoritmo tiende a llegar a soluciones que no sobrepasan el fitness 14, los resultados se muestran en la figura 10.

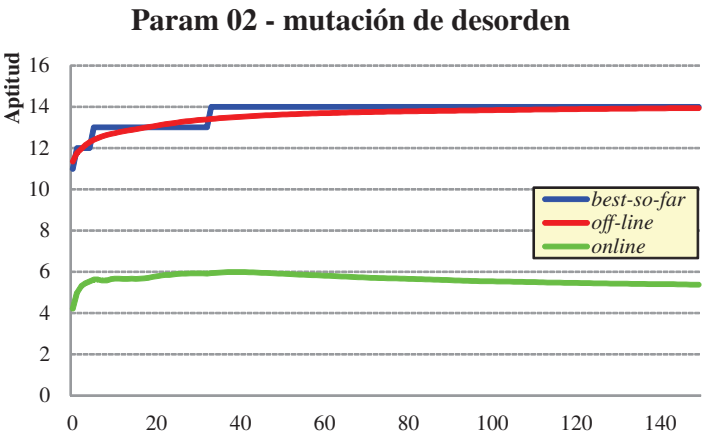

Fig. 10. Curva de los mejores elementos, curva offline y online aplicando sólo Mutación de Desorden.

b) Aplicando sólo el operador de Mutación de Posición, el algoritmo tiende a llegar a soluciones con fitness no mayor a 15, los resultados se muestran en la figura 11.

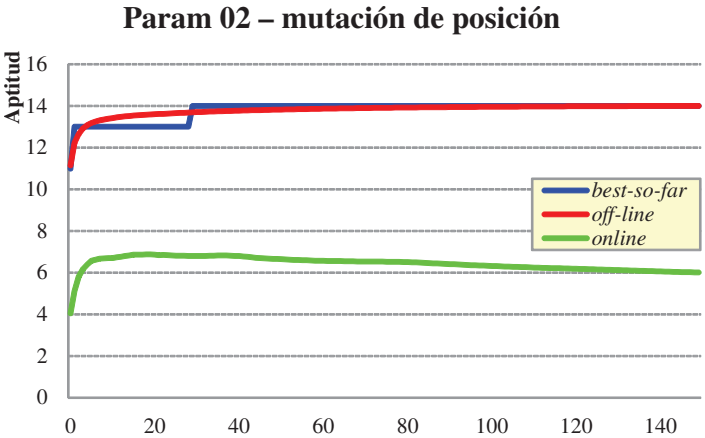

Fig. 11. Curva de los mejores elementos, curva offline y online aplicando sólo Mutación de Posición

c) Aplicando sólo el operador de Mutación Mastermind, el algoritmo tiende a llegar a soluciones que sobrepasan el fitness 16 pero que rara vez llega a 20, los resultados se muestran en la figura 12.

Finalmente, después de experimentar con los porcentajes se llegó a la conclusión que aplicando el operador de Mutación de Desorden en un 20%, el operador de Mutación de Posición en un 10% y finalmente el operador de Mutación Mastermind en un 70%, se logra alcanzar la mayoría de veces el fitness 20, llegando a solucionar el juego, los resultados se muestran en la figura 13.

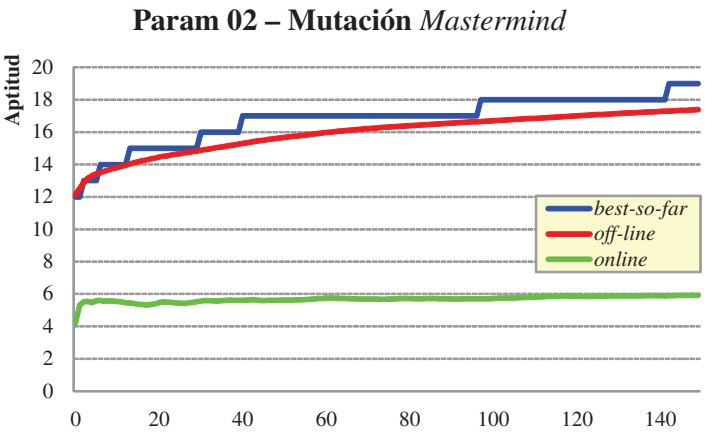

Fig. 12. Curva de los mejores elementos, curva offline y online aplicando sólo Mutación Mastermind.

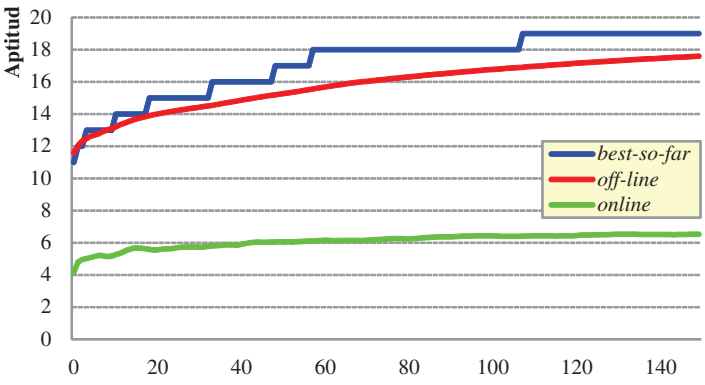

Fig. 13. Curva de los mejores elementos, curva offline y online aplicando Mutación de Desorden 20%, Mutación de Posición 10% y Mutación Mastermind 70%.

VI. Conclusiones y Trabajos Futuros

En este trabajo se demostró que es factible encontrar con AGs buenos resultados para el problema de Optimización Combinatoria que existe en el juego "*Mastermind*". En las pruebas experimentales que se plantearon, el modelo evolutivo AG produjo comparables a los obtenidos mediante otras estrategias aplicadas al juego. Los resultados muestran que los modelos evolutivos pueden localizar los picos altos en el espacio de búsqueda más rápido que usando el método de búsqueda exhaustiva ya que llega a resolver el juego en mejor tiempo. Igualmente se verificó que, con modelos evolutivos es conceptualmente más fácil realizar este tipo de búsqueda para este tipo de problemas ya que las operaciones que maneja son simples y fáciles de comprender. Si bien, el modelo evolutivo se probó en un problema de Optimización Combinatoria que es relativamente pequeño, es factible utilizar el mismo en problemas combinatorios con espacios de búsqueda mucho más grandes.

Como tema de investigación en el futuro, se desarrollaran mejoras, como poder evaluar todas las combinaciones, no solo midiendo su distancia a la combinación secreta, sino también por su poder predictivo; No todas las combinaciones son capaces de extraer la misma información como se vio en algunos trabajos previos que estudian el hecho de lanzar adivinanzas estratégicamente como en [1]; se puede tomar este trabajo como base para implementar un modelo híbrido Evolutivo – Adivinanzas Estratégicas, con la intención de extraer información y reducir el tamaño del espacio de búsqueda. Así se espera llegar a la solución del juego de forma aún más rápida.

References

[1] D. E. Knuth, "The computer as master mind," *Journal od Recreational Mathematics*, vol. 9, pp. 1–6, 1976-77.
[2] L. Berghman, D. Goossens, and R. Leus, "Efficient solutions for mastermind using genetic algorithms," *Computers & operations research*, vol. 36, no. 6, pp. 1880–1885, 2009.
[3] R. Irving, "Towards an optimum mastermind strategy," *Journal of Recreational Mathematics*, vol. 11, no. 2, pp. 81–87, 1978.
[4] B. Kooi, "Yet another mastermind strategy," *ICGA Journal*, vol. 28, no. 1, pp. 13–20, 2005.
[5] A. Bestavros and A. Belal, "Mastermind. a game of diagnosis strategies," in *Alexandria University*. Citeseer, 1986.
[6] K. Koyoma and W. Lai, "An optimal mastermind strategy," *Recreational Mathematics*, pp. 251–256, 1994.
[7] Z. Chen, C. Cunha, and S. Homer, "Finding a hidden code by asking questions," *Computing and Combinatorics*, pp. 50–55, 1996.
[8] E. Shapiro, "Playing mastermind logically," *ACM SIGART Bulletin*, no. 85, pp. 28–29, 1983.
[9] P. Swaszek, "The mastermind novice," *Journal of Recreational Mathematics*, vol. 30, no. 3, pp. 193–198, 2000.
[10] R. Rosu, "Mastermind," *Master's thesis, North Carolina State University, Raleigh, North Carolina*, 1999.
[11] L. Bento, L. Pereira, and A. Rosa, "Mastermind by evolutionary algorithms," in *Proceedings of the 1999 ACM symposium on Applied computing*. ACM, 1999, pp. 307–311.
[12] A. Singley, "Heuristic solution methods for the 1-dimensional and 2-dimensional mastermind problem," Ph.D. dissertation, UNIVERSITY OF FLORIDA, 2005.
[13] J. Merelo-Guervós, P. Castillo, and V. Rivas, "Finding a needle in a haystack using hints and evolutionary computation: the case of evolutionary mastermind," *Applied Soft Computing*, vol. 6, no. 2, pp. 170–179, 2006.
[14] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 1st ed. Ann Arbor, Michigan: University of Michigan Press, 1975.
[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Co., 1989.
[16] F. Rodrigues and M. Leonhardt, "Inteligência artificial utilizando algoritmos genéticos para evoluir uma estratégia para mastermind," *Anais SULCOMP*, no. 0, 2010.
[17] T. Kalisker and D. Camens, "Solving mastermind using genetic algorithms," in *Genetic and Evolutionary Computation- GECCO 2003*. Springer, 2003, pp. 206–206.
[18] I. S. for Genetic Algorithms, *Foundations of Genetic Algorithms*, ser. Lecture notes in computer science. Morgan Kaufmann Publishers, 1991, no. v. 1. [Online]. Available: http://books.google.com.pe/books?id=Df12yLrlUZYC