

Algoritmos genéticos

Fundamentos de la
inteligencia artificial



tech

CONTENIDO

- 1. Objetivos**

- 2. Introducción**

- 3. Historia**

- 4. Base biológica**

- 5. Codificación de problemas**

- 6. Generación de la población inicial**

- 7. Algoritmo principal y operadores genéticos**

- 8. Evaluación de individuos: *fitness***

- 9. Resumen**

- 10. Bibliografía**

OBJETIVOS

- Comprender los algoritmos genéticos.
- Distinguir intercambio de Información genética entre los individuos.
- Identificar cómo se genera la población de individuos.
- Calcular la evaluación de los individuos.

INTRODUCCIÓN

La ciencia se ha intuido en la naturaleza para crear modelos exitosos en una variedad de entornos. Cabe destacar que, gracias el reconocimiento que se le ha realizado a la observación de estos patrones, se ha logrado avances en muchas ramas del trabajo científico, incluida la inteligencia artificial (IA). Asimismo, un algoritmo genético no es más que una tecnología de IA inspirada en la idea de que lo que queda es lo mejor que se adapta a su entorno, tomando en cuenta que es la base de la teoría de la evolución desarrollada por Charles Darwin, idea que formuló para unir la evolución con la genética. Este proceso encuentra una solución a un problema específico que involucra mecanismos que imitan la evolución de las especies biológicamente.

HISTORIA

Los modelos originarios de lo que ahora podríamos llamar algoritmos genéticos aparecieron a fines de la década de 1950 cuando Alan Turing propuso la máquina de aprendizaje, tomando en cuenta que es paralelo a los principios evolutivos, de igual manera a principios de la década de 1960, se proyectan en computadoras por biólogos evolutivos, que buscaban modelar sin ambigüedades aspectos de la evolución natural. Ninguno de ellos cree que esta estrategia se pueda aplicar en general a problemas sintéticos, pero tal enfoque no tardará mucho en el futuro, ya que las computadoras evolutivas ciertamente han existido desde los primeros días de las computadoras electrónicas [3].

En 1962, investigadores como Box, Friedman, Bledsoe y Bremermann desarrollaron, de forma independiente, los algoritmos inspirados en la evolución para la optimización de funciones y el aprendizaje automático, pero su trabajo generó poca respuesta. Sin embargo, el trabajo de John Holland sobre sistemas adaptativos sentó las bases para desarrollos posteriores. Es importante destacar que Holland también fue la primera persona en sugerir, explícitamente, el cruzamiento y otras combinaciones de operadores.

Un desarrollo más exitoso ocurrió en 1965, cuando Ingo Rechenberg, en la Universidad Técnica de Berlín, introdujo una técnica que llamó estrategia evolutiva, aunque es similar a los algoritmos escaladores en lugar de algoritmos genéticos. El siguiente gran desarrollo en este campo se produjo en 1966, cuando Vogel, Owens y Walsh introdujeron una técnica que llamaron programación evolutiva. En este enfoque, las soluciones candidatas a los problemas se representaron como máquinas de estados finitos simples.

"Los algoritmos genéticos se originaron en la década de 1970 gracias a John Henry Holland. Es básicamente una estrategia utilizada en la inferencia aleatoria sobre la base de problemas de búsqueda óptima. La idea es simular el proceso de selección natural. Tomando en cuenta que estos algoritmos provocan la evolución de un grupo de individuos haciéndolos realizar acciones aleatorias similares a las de la evolución biológica, es decir, mutación y recombinación. Asimismo, como seleccionados de acuerdo con ciertos criterios, a partir de los cuales se determinan los individuos más adecuados, que sobreviven y se descartan cuáles son de menos idoneidad" [1].

Estos trabajos fundamentales demostraron un interés más generalizado en la computación evolutiva. Desde principios hasta mediados de la década de 1980, los algoritmos genéticos se aplicaron en una variedad de áreas, desde problemas matemáticos abstractos, como el llenado de contenedores y la coloración de gráficos, hasta problemas de ingeniería más concretos como el control de flujo en línea, el reconocimiento de patrones, la clasificación estructural y la optimización.

En general, la computación evolutiva se puede definir como un conjunto de modelos computacionales inspirados en la evolución. Formalmente, el término computación evolutiva se refiere al estudio de los fundamentos y aplicaciones de ciertos métodos heurísticos basados en los principios de la evolución natural. Estos métodos experimentales se pueden clasificar en tres grandes categorías que conducen a la siguiente ecuación evolutiva.

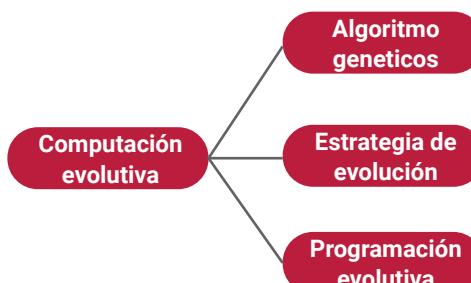


Figura 1. Computación evolutiva.

Antes de que ocurran los vínculos para producir descendencia, se realiza una mutación en los progenitores. Por el contrario, la computación evolutiva puede considerarse como una de las áreas de estudio de lo que se conoce como computación blanda o *soft computing*.

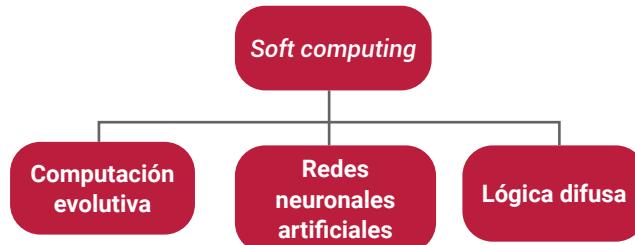


Figura 2. Tipos de soft computing.

BASE BIOLÓGICA

En disciplinas tan diversas como es el caso de la biología, las matemáticas o la informática, muchas ideas asombrosas provienen de puntos de vista aparentemente diferentes pero que, en el fondo, son idénticos. En los últimos años, los biólogos se han interesado cada vez más por la base molecular de las funciones biológicas. Actualmente, el único modelo biológico que se posee de los seres vivos es el elemento observable que los constituye. Esto, que parece limitante, resulta beneficioso a la hora de estudiar la gran diversidad de seres al provenir todos ellos de un origen común. Por tanto, la definición de vida y sus características deben basarse en un modelo único: la célula. Ella constituye la forma más simple ya sea en organismo uni o multicelulares. Conviene excluir de la consideración de seres vivos a los virus que, por su singular naturaleza, no se les considera como tales.

Partiendo de lo dicho, en la naturaleza, los individuos de una población compiten entre sí por recursos como alimentos, agua y refugio. Incluso, los miembros de la misma especie a menudo compiten por parejas. Los individuos más exitosos en supervivencia y atractivo. Asimismo, los compañeros tienen más probabilidades de producir más descendencia. Por el contrario, los individuos menos talentosos producen menos generaciones de descendientes. Esto significa que los genes de los individuos más adaptados se reducirán en generaciones posteriores a medida que aumente el número de individuos.

Una combinación de buenas cualidades de diferentes ancestros, a veces, puede producir una descendencia supraindividual cuyas adaptaciones son mucho mayores que las de cualquiera de sus antepasados. De esta forma, las especies evolucionan para adquirir características

cada vez más apropiadas al medio en el que viven.

"Un algoritmo genético trabaja entre el conjunto de soluciones a un problema denominado fenotipo y el conjunto de individuos de una población normal, codificando la información de cada solución en una secuencia, generalmente binaria, denominada cromosomas. Los códigos que componen la secuencia se denominan genes. Cuando los cromosomas se representan mediante secuencias de números binarios, se denomina genotipo. Los cromosomas se desarrollan a través de muchas repeticiones, llamadas generaciones. En cada generación, los cromosomas se evalúan utilizando una escala física específica. Asimismo, las generaciones subsiguientes, es decir, los cromosomas nuevos, se producen mediante la aplicación repetida de operadores genéticos, que son factores de selección, cruce, mutación y reemplazo" [2].

Tomando en cuenta que los algoritmos genéticos utilizan una similitud inmediata con la conducta natural. Sin embargo, estos trabajan con un grupo de individuos, cada uno de los cuales representa una posible solución a un problema en particular. Asimismo, a cada uno de los individuos se le asigna un valor o puntaje, relacionado con la calidad de dicha solución. En esencia, esto equivale a la eficiencia con la que un organismo puede competir por ciertos recursos. Cuando el individuo esté adaptado al problema, lo más probable es que elija engendrar cruzando su material genético con otro individuo escogido. Esta intersección dará como resultado la producción de nuevos individuos, es decir, son descendientes que comparten algunas características

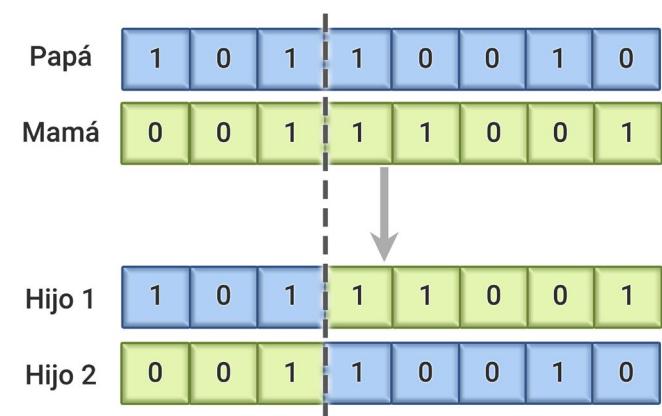


Figura 3. Intercambio de información genética entre dos individuos [7].

de sus progenitores. Sin embargo, cuanto menos apto es un individuo, menos probable es que exista.

Este se selecciona para la preproducción, por lo que su material genético se transmite a la siguiente generación. De esta forma, se crea un nuevo conjunto de posibles soluciones, que sustituye al anterior y comprueba la interesante propiedad que tiene una mayor proporción de buenas características en la solución.

En comparación con la población anterior, así, a lo largo de las generaciones, los rasgos positivos se extendieron entre la población. Al favorecer a los individuos mestizos que son más adaptables, se están explorando áreas prometedoras de investigación. Si el algoritmo genético está bien diseñado, la población convergerá hacia la solución óptima del problema.

Tomando en cuenta que la fuerza de los algoritmos genéticos proviene del hecho de que son una tecnología poderosa y pueden resolver con éxito muchos problemas de varios campos, incluidos aquellos que son difíciles con otros métodos. Aunque no hay garantía de que un algoritmo genético encuentre la solución óptima a un problema, existe evidencia empírica de que encuentran soluciones aceptables en el tiempo que compiten con el resto de los algoritmos de optimización armónica. En los casos en que existen técnicas especializadas para resolver un problema en particular, es probable que supere al algoritmo genético en términos de velocidad y eficiencia. El amplio campo de aplicación de los algoritmos genéticos se ocupa de problemas para los que no existe una tecnología especializada, incluso, en el

caso de aquellas técnicas estén en su lugar y funcionan bien, y se pueden hacer mejoras cruzándolas y usando algoritmos genéticos.

CODIFICACIÓN DE PROBLEMAS

Cualquier posible solución a un problema se puede dar otorgando valores a un conjunto de parámetros. Tomando en cuenta, y como se comentó anteriormente, el conjunto de todos los parámetros, es decir, genes en términos de algoritmos genéticos codificados en una serie de valores se denomina cromosoma.

"El algoritmo genético representa los posibles procedimientos en el espacio de investigación de individuos que estén representados por un solo cromosoma con ciertos valores. La secuenciación completa es el genotipo de un cromosoma y la función que asigna el cromosoma a la solución se evalúa en una función de aptitud que determina su fenotipo. Asimismo, la codificación de los algoritmos genéticos indica el tipo de valores que puede asumir cada locus de genotipo. Cuando estos valores se derivan del alfabeto binario, esta es la notación

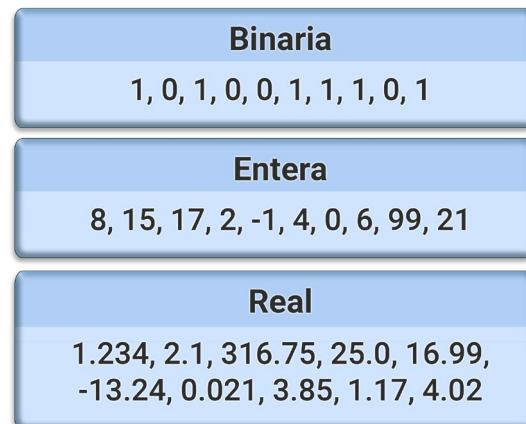


Figura 5. Tipos de codificación en algoritmo genético [4].

binaria. Si los valores pueden tomar números enteros, esta es la notación de números enteros. Cuando el valor de los cromosomas puede ser un número real, tenemos una codificación real" [4].

El conjunto de parámetros representados por un cromosoma en particular se denomina genotipo y este contiene la información necesaria para construir el organismo, tomando en cuenta que sería verdadera solución al problema, conocido como fenotipo. Es decir, biológicamente, la información genética en el ADN de un individuo sería el genotipo, mientras que la expresión de ese ADN del mismo individuo sería el fenotipo. De igual manera:

"Desde el trabajo original de John Holland, la codificación, a menudo, se realiza utilizando valores binarios. A cada parámetro se le asigna un cierto número de *bits* y arbitra la variable que representa

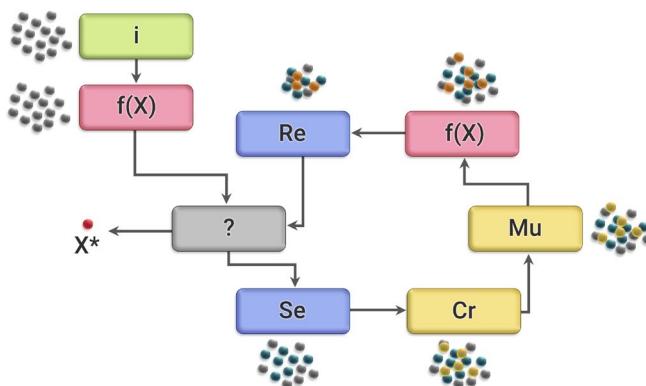


Figura 4. Algoritmo genético [2].

cada gen. El número de *bits* asignados dependerá del nivel de sintonía requerido. Obviamente, no todos los parámetros deben codificarse con el mismo número de *bits*. Cada *bit* que pertenece a un gen, a menudo, se denomina alelo" [5].

Es posible tener representaciones que codifiquen cada parámetro directamente utilizando valores enteros,

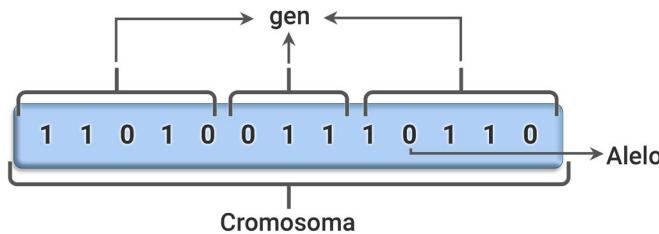


Figura 6. Individuo genético binario [5].

reales o de punto flotante. Aunque se ha acusado a estas representaciones de mitigar el potencial paralelismo de las representaciones binarias y, de esta manera, se permite el desarrollo de factores genéticos más específicos del campo de aplicación de los algoritmos genéticos.

GENERACIÓN DE LA POBLACIÓN INICIAL

Anteriormente, cuando se habló de individuo, se hacía referencia a todas las posibles soluciones de los problemas a resolver. Tomando en cuenta que en el caso de maximizar o minimizar una función, generalmente, cada individuo representa un conjunto de valores de las variables.

El principal objetivo de crear un conjunto inicial es identificar los puntos de partida para la investigación en el espacio de soluciones. Tomando en cuenta que el primer paso del algoritmo genético es crear un conjunto aleatorio inicial de individuos. La siguiente función es crear una matriz donde cada fila consta de un conjunto de valores numéricos aleatorios. Además, el valor de cada variable se puede dividir en un rango. Esta determinación es útil para acelerar el proceso de optimización, pero requiere información que ayude a acotar el rango de valores en los que se encuentra la solución óptima [6].

La definición del tamaño de la población es muy importante, ya que:

- Un valor demasiado grande aumentará el consumo

de recursos.

- Un valor demasiado pequeño ocupará un espacio de búsqueda más pequeño.

Y, por lo tanto, llevará mucho tiempo para encontrar la mejor solución. La población inicial se genera aleatoriamente en el Algoritmo Genético Simple (SGA).

A continuación, se presenta ejemplos facilitado en [6]:

Ejemplo 1:

Supóngase que la función objetivo es $J(x,y,z)$ que depende las variables x,y,z , tomando en cuenta que el individuo contiene los siguientes valores 3,9.5,-0.5 que equivalen a la combinación de valores

$$x=3$$

$$y=9.5$$

$$z=-0.5$$

Ejemplo 2:

Se crea una población de 10 individuos de longitud 2, con los valores de la primera variable acotados entre [-100, +100] y la segunda con únicamente el límite inferior [-20, NA].

```
poblacion <- crear_poblacion(
  n_poblacion = 10,
  n_variables = 2,
  limite_inf = c(-100, -20),
  limite_sup = c(+100, NA),
  verbose = TRUE
)
## 
## Población inicial creada
## -----
## Fecha creación: 2020-11-12 12:21:44
## Número de individuos = 10
## Límites inferiores de cada variable = -100, -20
## Límites superiores de cada variable = 100, 1000
## Población
##      [,1]   [,2]
## [1,] -62.28138 335.615078
## [2,] -98.78874 240.589471
## [3,] -16.62883  5.461185
## [4,]  77.30608 -18.495540
```

```

## [5] 43.67225 709.264950
## [6] 82.62979 549.810212
## [7] -62.29848 427.149130
## [8] 27.80573 933.015135
## [9] 54.55204 522.964812
## [10] -60.75474 990.104857
## attr("fecha_creacion")
## [1] "2020-11-12 12:21:44 UTC"

```

```

## attr("numero_individuos")
## [1] 10
## attr("class")
## [1] "matrix" "poblacion"

```

ALGORITMO PRINCIPAL Y OPERADORES GENÉTICOS

"Los algoritmos genéricos operan sobre una

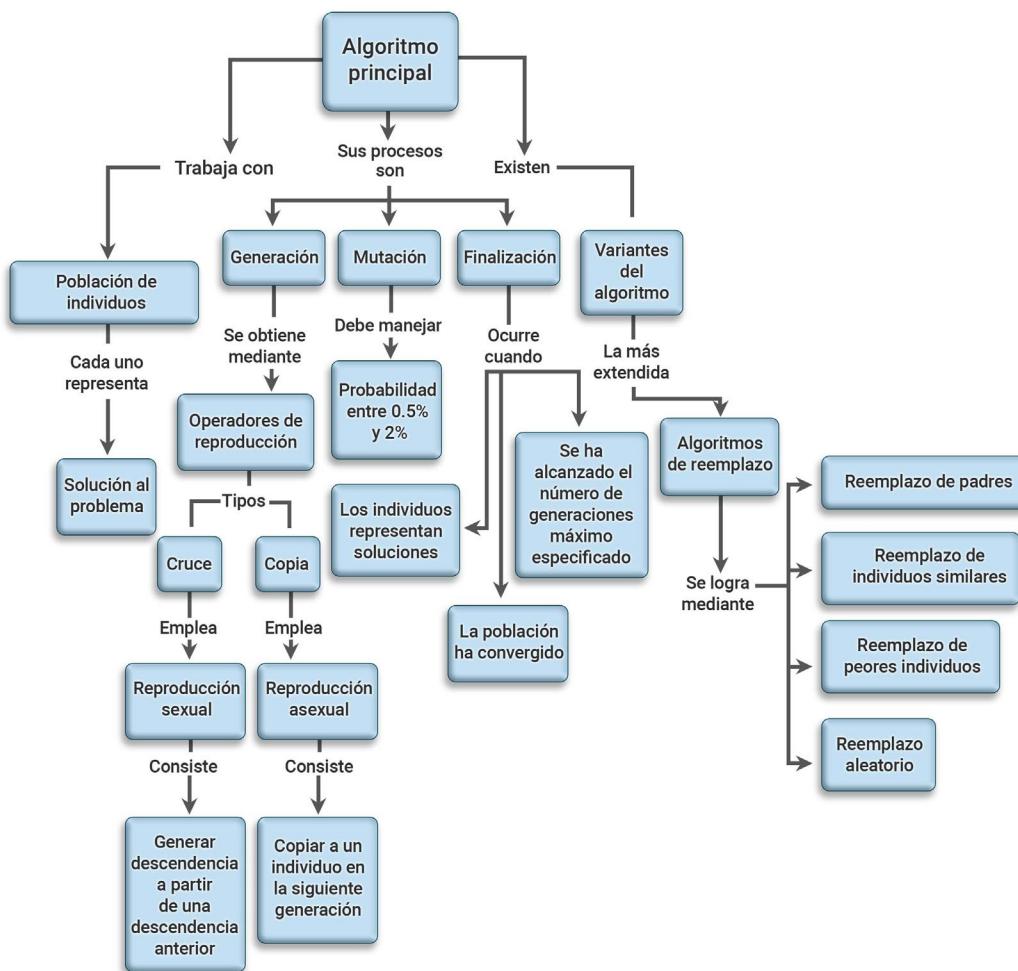


Figura 7. Algoritmo principal [11].

población de individuos. Cada uno de ellos representa una posible solución al problema a resolver. Cada individuo tiene un ajuste asociado con la calidad al problema de la solución que representa. Esencialmente, el equivalente sería una medida de la eficacia del individuo en la lucha por los recursos" [8].

En [11] se indica que el funcionamiento general del algoritmo genético se puede ver en el siguiente pseudocódigo:

Iniciar población actual aleatoriamente

MIENTRAS no se cumpla el criterio de terminación

crear población temporal vacía

MIENTRAS población temporal no llena

seleccionar padres

cruzar padres con probabilidad P_c

SI se ha producido el cruce

mutar uno de los descendientes con probabilidad P_m

evaluar descendientes



```

añadir descendientes a la población temporal
SINO
añadir padres a la población temporal
FIN SI
FIN MIENTRAS

```

```

aumentar contador generaciones
establecer como nueva población actual la población temporal

```

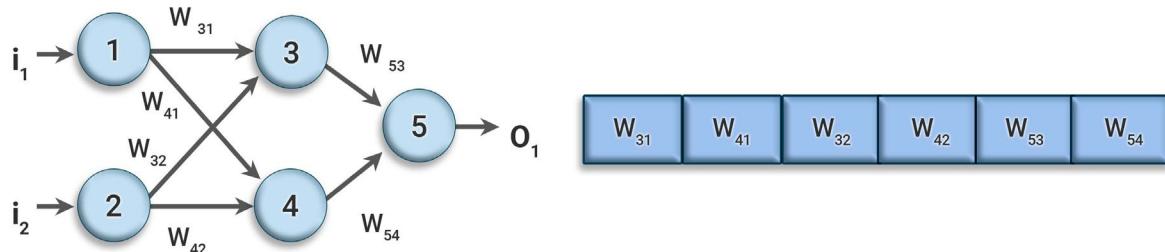


Figura 8. Ejemplo de codificación de la red neuronal artificial [5].

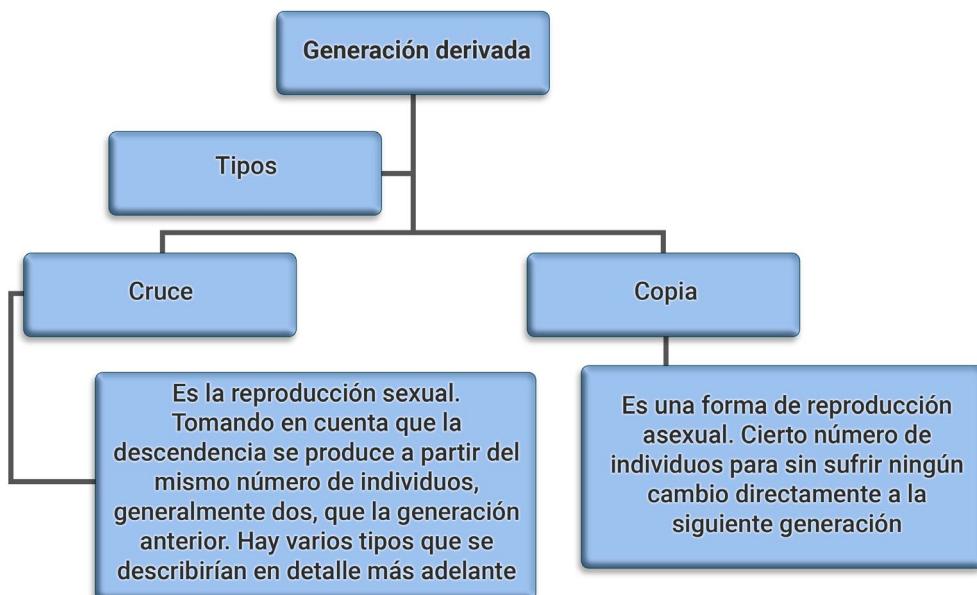


Figura 9. Generación derivada [8].

FIN MIENTRAS

Tomando en cuenta que existen dos tipos de una generación derivada de la generación anterior gracias a los operadores de reconstrucción. Estos son los siguientes:

Al crearse nuevos individuos, se realiza una mutación con un potencial P_m . Tomando en cuenta que la probabilidad de una mutación suele ser muy baja, normalmente entre el 0,5 % y el 2 %. Este proceso finaliza cuando se cumple cualquiera de los criterios de cierre especificados. Los más comunes suelen ser los siguientes:

- La población contiene individuos que representan soluciones que son lo suficientemente buenas para resolver el problema.
- Un gen converge cuando el 95 % de la población tiene el mismo valor, en el caso de trabajar con

codificaciones binarias o con valores dentro de un rango específico, si se llegara el caso de trabajar con otras codificaciones. Sin embargo, una vez que todos los genes convergen, la población converge. Cuando sucede este caso, la media de la población se acerca al bien del individuo.

- Se alcanza el número máximo de generaciones especificado.

Se han identificado varias variaciones de este algoritmo propuesto originalmente por John H. Holland. Quizás uno de los métodos más comunes es eliminar temporalmente la población para que los operadores de intercambio de genes y las mutaciones se apliquen directamente al acervo genético. Con esta variante, el proceso de cruzar el mar cambia ligeramente. Ahora bien, en el caso del

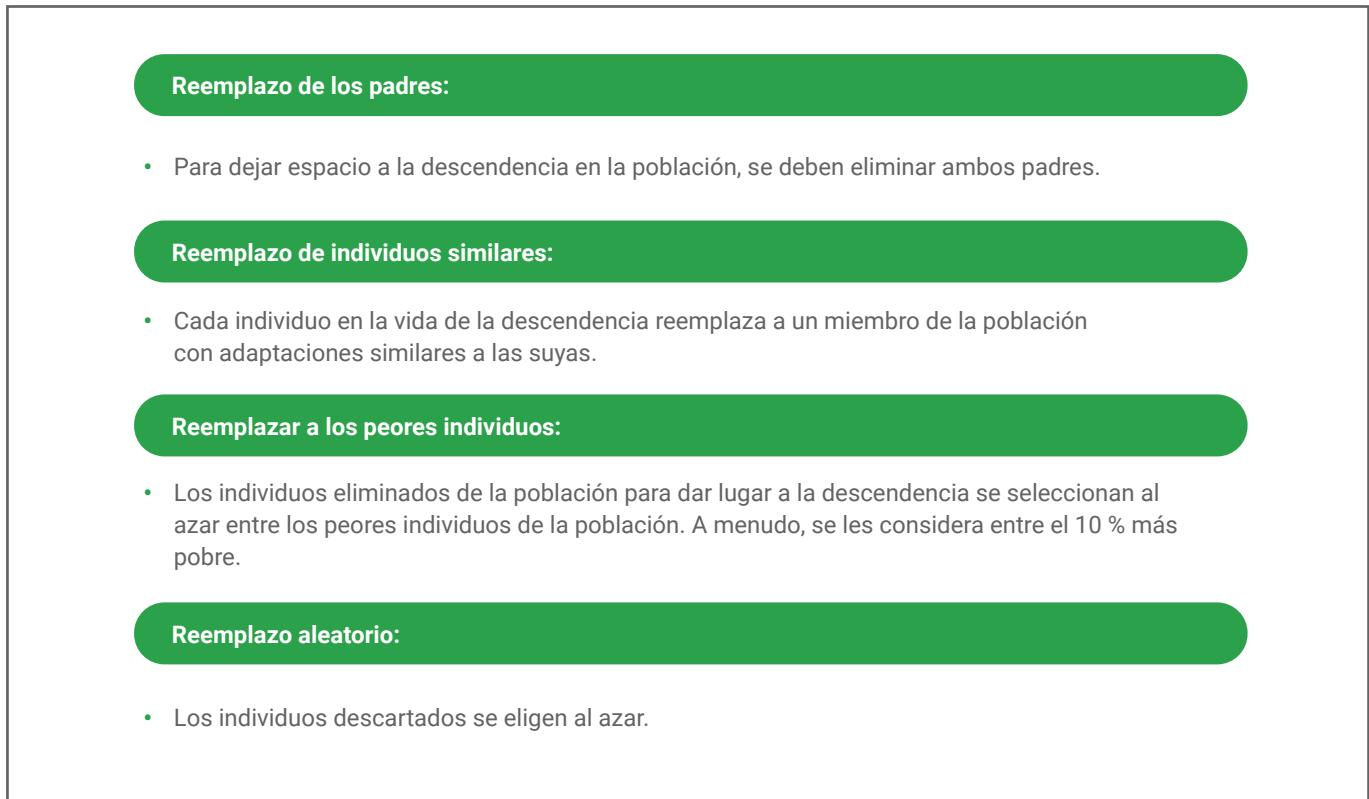


Figura 10. Descendencia de individuos en una población.

mestizaje, no basta con introducir directamente a la descendencia en la comunidad. Dado que el número de individuos de la población debe mantenerse constante, se debe crear un espacio antes de que la descendencia se introduzca en la población. Hay varias opciones para esto:

Trabajando con una sola población no se puede decir que se pasa a la siguiente generación cuando la población está llena, porque siempre está llena. En este caso, la transición a la siguiente generación ocurrirá una vez que se alcancen algunos cruces y mutaciones. Este número dependerá de la mutación definida por el usuario y las tasas de mutación y el tamaño de la población. Por lo tanto, con una tasa de hibridación del 90 %, una tasa de mutación del 0,02 % y un efecto sobre 100 individuos se pasará a la siguiente generación cuando se alcancen los 45 cruces (cada híbrido produce 2 individuos de los cuales 90 se introdujeron en la población), eso es 90 % o dos mutaciones.

Así, el algoritmo genético opera a nivel genético de soluciones a través de la siguiente secuencia:

- Se inicia con un conjunto o población inicial que se puede generar aleatoriamente.
- Estimar el *fitness* de cada individuo.
- Utilizar el factor de selección basado en la proporcionalidad de la población.
- Utilizar el factor u operador genético de reproducción, es decir, cruce o mutación a la

población actual para crear la próxima generación de la población.

- Ir al segundo paso hasta que se cumpla la condición de parada.
- Cuando se cumple la condición del segundo paso, se devuelve al mejor individuo hallado.

EVALUACIÓN DE INDIVIDUOS: *FITNESS*

“La función de evaluación en general es una función objetivo, es decir, lo que se quiere mejorar u optimizar. Es necesario decodificar la solución contenida en el cromosoma para su evaluación. Cabe señalar que la función de aptitud (*fitness*) es la que nos permite evaluar las actitudes de un individuo y siempre debemos asumir valores positivos” [9].

A menudo, ambas funciones son iguales, pero es posible que la función de destino sea demasiado compleja y tome un valor negativo que proporcione un valor numérico, por lo que se debe especificar otra función adecuada. Se debe tener en cuenta que, para que un algoritmo genético funcione correctamente, se debe tener una forma de saber si los individuos de una población son buenas soluciones para el problema en cuestión. Por tanto, para resolver cada tipo de problema es necesario encontrar un nuevo método, como ocurre con el propio

cifrado o codificación de los individuos único.

En un algoritmo genético, la información debe estar codificada para poder trabajar correctamente con ella. Hay muchos sistemas de codificación. Tomando en cuenta que una vez que se haya decidido qué esquema de codificación usar, se debe ver cómo funcionan los operadores de selección, cruce y mutación.

Tomando en cuenta que de esto se encarga la función de evaluación, esta asigna una medida numérica para verificar qué tan buena es una solución. Este procedimiento se llama modificación o ajuste. En la naturaleza, la aptitud de un individuo puede considerarse como la probabilidad de que sobreviva hasta la edad fértil y reproductiva. Sin embargo, esta posibilidad debe ser ponderada por el número de descendiente.

Un 25% de probabilidad de reproducción en una población de varios cientos de individuos no corresponde a la misma probabilidad en una población de varios millones de individuos. En el mundo de los algoritmos genéticos, esta métrica se utiliza para controlar aplicaciones de factores genéticos. En otras palabras, admitirá controlar la cantidad de selecciones, cruces, clonaciones o copias y mutaciones que se realizan.

El enfoque más común es establecer medidas de ajuste explícitas para cada miembro o individuo de la población. Se asigna un valor de ajuste estándar a cada individuo a través de un proceso de evaluación bien definido. Como se mencionó, este procedimiento de evaluación será

específico para el área del problema al que se aplica el algoritmo genético. La idoneidad también se puede calcular mediante el método de coevolución. Es decir, la adaptación de la estrategia de un juego se determina aplicando esa estrategia contra una población completa o alternativamente una muestra de estrategias opuestas.

$$r_{(i,t)} = \sum_{j=1}^{N_c} |s(i,j) - c(i,j)|$$

A continuación, en [10] se presentan los tipos de ajuste o *fitness*:

La ecuación 1 pertenece a *fitness* puro, el cual es la medida de ajuste la que determina naturalmente el mismo problema. Tomando en cuenta que esta especifica el cálculo del buen valor de una instancia de i en el tiempo t.

Es decir:

$$s_{(i,t)} = \begin{cases} r_{(i,t)} & \text{minimo} \\ r_{\max} - r_{(i,t)} & \text{máximo} \end{cases}$$

Donde:

- $s(i,j)$: valor anhelado para el individuo i en el caso de j
- $c(i,j)$: valor logrado por el individuo i para el caso j

Mientras que la ecuación 2 concierne a *fitness* estandarizado, para dar solución a la dualidad cuando se encuentran en problemas de mínimos o máximos, varía el ajuste puro con respecto a la siguiente ecuación:

El problema de mínimo se utiliza llanamente la medida

$$a_{(i,t)} = \frac{1}{1+s(i,t)}$$

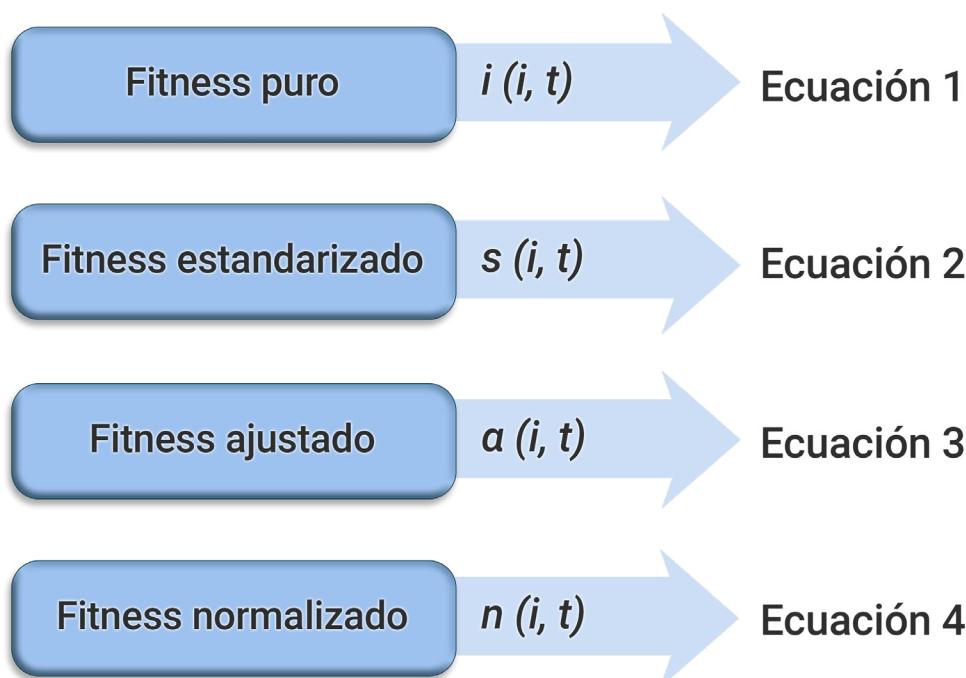


Figura 11. Tipos de fitness [10].

de *fitness* puro. Al tener un problema de maximización simple, entonces el ajuste neto se resta del límite superior r_{\max} del error. Usando la escala, qué tan bueno es un individuo cuando el valor ajustado es cercano a cero. Así, en la generación t , la instancia i siempre será mejor que la instancia j si se comprueba que $s(i,t) < s(j,t)$.

Posteriormente, la ecuación 3 corresponde a *fitness* ajustado, donde se obtiene aplicando una transformación de reflexión.

De esta forma, el ajuste asumirá siempre los valores del intervalo [0...1] cuando más cerca esté el físico de un individuo de 1, mayor será su bondad.

Con la ecuación 4 se obtiene una ecuación de tamaño n , asimismo, estos diferentes ejemplos de *fitness* que se han visto hasta ahora indican qué tan bueno es el individuo. Tomando en cuenta que el ajuste estándar introduce una nueva dimensión, indicando qué tan buena es una solución en comparación con el resto de las soluciones representadas en la población.

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^n a(k,t)}$$

El *fitness* ajustado siempre tomará valores en el rango [0...1], donde los mejores individuos están más cerca de la unidad. Pero, a diferencia de lo anterior, un valor cercano a 1 indica no solo que este individuo es una buena solución al problema, sino también una solución mucho mejor que la proporcionada por el programa. La suma de los valores normales de *fitness* de todos los individuos será siempre 1. Este tipo de afinación se utiliza en la mayoría de los métodos de selección proporcional a la conveniencia.

RESUMEN

Los modelos originarios de lo que se ahora podría llamar algoritmos genéticos aparecieron a fines de la década de 1950 cuando Alan Turing propuso la máquina de aprendizaje, tomando en cuenta que los algoritmos genéticos se originaron en la década de 1970 gracias a John Henry Holland. Es básicamente una estrategia utilizada en la inferencia aleatoria sobre la base de problemas de búsqueda óptima. La idea es simular el proceso de selección natural.

En las áreas biología, matemáticas e informática, muchas ideas realmente asombrosas provienen de los siguientes puntos de vista aparentemente diferentes, son idénticos. En los últimos años, los biólogos se han interesado cada vez más en la base molecular de sus funciones biológicas. Actualmente, el único modelo biológico que tienen de los seres vivos son los que envuelven al ser humano.

BIBLIOGRAFÍA

- [1] A. Fernández, "Breve historia de los algoritmos genéticos", 2017. [En línea]. Disponible en: <https://planetachatbot.com/entendiendo-los-algoritmos-geneticos-caso-de-uso-entorno-organizacional/#:~:text=Los%20algoritmos%20gen%C3%A9ticos%20nacieron%20en,en%20simular%20la%20selecci%C3%B3n%20natural>
- [2] Wikipedia, enciclopedia libre, "Algoritmo genético", 2022. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico
- [3] F. S. Caparrini, "Algoritmos Genéticos", 2019. [En línea]. Disponible en: <http://www.cs.us.es/~fsancho/?e=65>
- [4] M.C.Soberani, "Codificación en algoritmos genéticos", 2020. [En línea]. Disponible en: <https://medium.com/soldai/codificaci%C3%B3n-en-algoritmos-gen%C3%A9ticos-a1b6516747f2#:~:text=La%20codificaci%C3%B3n%20en%20los%20algoritmos,enteros%2C%20es%20una%20codificaci%C3%B3n%20entera>
- [5] M. Gestal, "Codificación de Problemas". [En línea]. Disponible en: <http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/node6.html>
- [6] J. A. Rodrigo, "Optimización con algoritmo genético y Nelder-Mead", 2019. [En línea]. Disponible en: https://www.cienciadedatos.net/documentos/48_optimizacion_con_algoritmo_genetico#:~:text=El%20primer%20paso%20del%20algoritmo,acotado%20dentro%20de%20un%20rango
- [7] R. G. Juárez, "Algoritmos genéticos", 2018. [En línea]. Disponible en: <https://conogasi.org/articulos/algoritmos-geneticos/>
- [8] M. Gestal, "5. Algoritmo principal". [En línea]. Disponible en: <http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/node7.html>
- [9] J. C. López, "Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA", 2010. [En línea]. Disponible en: <https://www.adictosaltrabajo.com/2010/10/07/jgap/>
- [10] M. Gestal, "7. Evaluación". [En línea]. Disponible en: <http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/node20.html>
- [11] "Algoritmo Principal". [En línea]. Disponible en: <https://cursa.ihmc.us/ríd=1KNKWCM8D-1LB5J38-1B8L/Algoritmo>