Algoritmos Bioinspirados

Unidad III : Cómputo Evolutivo – Programación Genética

Programa: Ingeniería en Inteligencia Artificial

Población inicial

La inicialización de la población es el primer paso del proceso de evolución. Implica la creación de las estructuras del programa que luego evolucionarán.

Generalmente se inicia con un grupo de miles de programas de computadora generados aleatoriamente.

Se inicia por seleccionar una función del conjunto F al azar para que sea la raíz del árbol. Se restringe la selección de la raíz del árbol al conjunto de funciones F porque se debe generar una estructura jerárquica y no una estructura con un solo nodo terminal.

Método *"Full"*:

- Se selecciona una función de forma aleatoria con probabilidad uniforme del conjunto de funciones F para que sea la raíz del árbol; sea $\mathbf{z}$ la aridad de la función seleccionada.
- Se seleccionan n nodos con probabilidad uniforme del conjunto de funciones F, para que sean sus hijos;
- Para cada función dentro de estos n nodos, se invoca recursivamente el método de crecimiento, es decir, sus hijos se seleccionan del conjunto F, a menos que el nodo tenga una profundidad igual a $d_{max} - 1$. En este último caso, sus hijos se seleccionan de T.

La profundidad de un nodo es el número de aristas que deben atravesarse para llegar al nodo a partir del nodo raíz del árbol.

La profundidad de un árbol es la profundidad de su hoja más profunda.

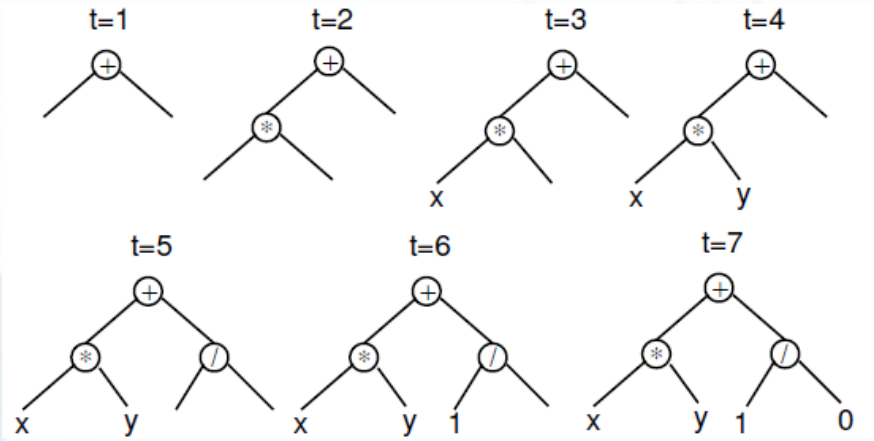
Método *"Grow"*:

- Se selecciona una función de forma aleatoria con probabilidad uniforme del conjunto de funciones F para que sea la raíz del árbol; sea $\mathbf{z}$ la aridad de la función seleccionada.
- Se seleccionan n nodos con probabilidad uniforme de la unión del conjunto de funciones y el conjunto terminal, $F \cup T$, para que sean sus hijos;
- Para cada función dentro de estos n nodos, se invoca recursivamente el método de crecimiento, es decir, sus hijos se seleccionan del conjunto $F \cup T$, a menos que el nodo tenga una profundidad igual a $d_{max} - 1$. En este último caso, sus hijos se seleccionan de T. Para los nodos terminales se detiene el crecimiento.

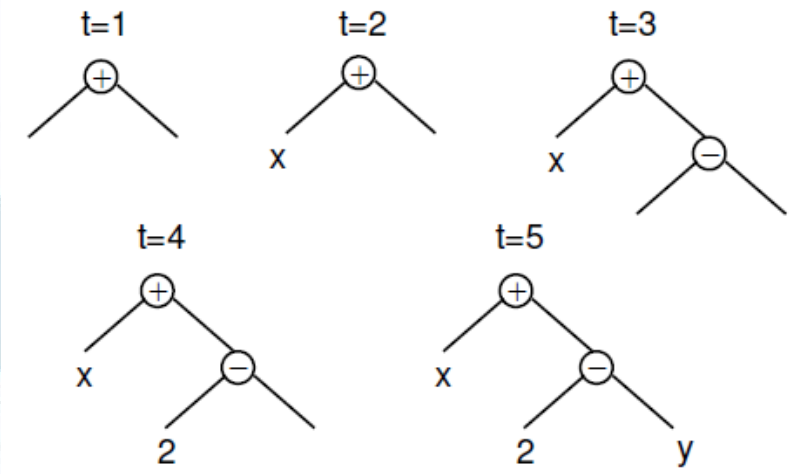
El rango de tamaños y formas de los programas producidos por el método *Full* puede ser limitado.

El método *Grow*, por el contrario, permite la creación de árboles de formas y tamaños más variados.

Ejemplo de árbol creado con el método *Full*.



Ejemplo de árbol creado con el método *Grow*.



Método *"ramped half-and-half"*:

- Según Koza, las poblaciones generadas con cualquiera de los 2 métodos presentados anteriormente pueden mostrar falta de diversidad.
- Por lo que propuso el método *ramped half-and-half* ;
- En este método:
  i. Una fracción $\frac{1}{d_{max}}$ de la población se inicializa con árboles de profundidad 1, otra fracción $\frac{1}{d_{max}}$ se inicializa con árboles de profundidad 2, y así consecutivamente hasta llegar a d_{max}.
  ii. Además, por cada grupo de las diferentes profundidades, la mitad de los árboles son inicializados con el método *Grow* y la otra mitad con el método *Full*.

Individuos duplicados en la población desperdician recursos computacionales y reducen la diversidad genética, por lo tanto, es deseable, pero no necesario, evitar duplicados.

La creación de individuos duplicados a través de las operaciones genéticas de reproducción es una parte inherente de los procesos genéticos.

La distribución de las formas de los árboles depende de manera no trivial de la relación entre el número de funciones de cada aridad y el número de terminales.

Si hay una gran cantidad de terminales, los árboles producidos por el método de crecimiento tienden a ser pequeños. Por el contrario, si hay pocos terminales y muchas funciones con una aridad de 2 o más, las ramas tenderán a alcanzar la profundidad máxima.

Selección

Al igual que con la mayoría de los algoritmos evolutivos, los operadores genéticos en PG se aplican a individuos que se seleccionan probabilísticamente en función de la aptitud.

- Selección por torneo
- **Selección proporcional (ruleta)**
- Selección posicional (ranking)
- Cualquier mecanismo de selección estándar de algoritmo evolutivo puede ser usado.

Reproducción

La operación de reproducción es asexual ya que opera con un solo padre y produce un solo hijo.

i. Seleccionar un padre de la población de acuerdo con algún método de selección basado en la aptitud.

ii. El individuo seleccionado es copiado, sin alteración, de la población actual a la nueva población.

Nota: el padre permanece en la población mientras se realiza la selección durante la generación actual. Es decir, la selección se realiza con reemplazo (reselección). Los padres pueden seleccionarse y, en general, se seleccionan más de una vez para la reproducción durante la generación actual. Esto se cumple para todos los operadores genéticos.

Cruza

La operación de cruce (recombinación sexual) para PG crea una variación en la población al producir nuevos hijos creados con partes tomadas de cada padre.

La implementación más común es el cruce de subárboles;

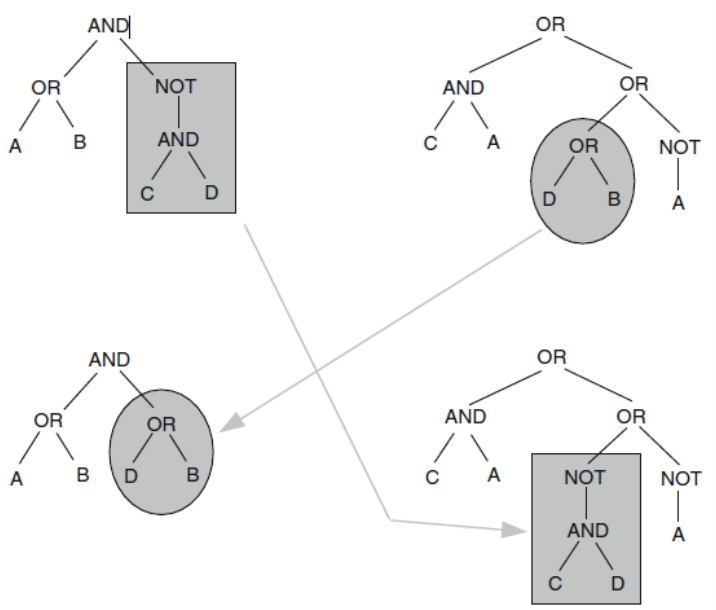
i. Seleccionar 2 padre de la población de acuerdo con algún método de selección basado en la aptitud.

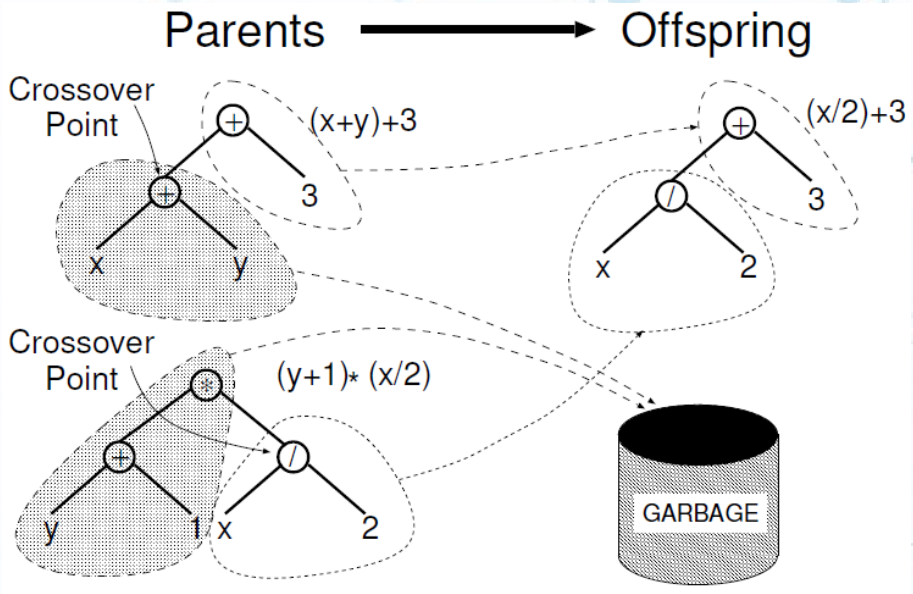
ii. Escoger 1 nodo al azar en cada árbol, a los cuales se les llama punto de cruza.

iii. Se produce 1 hijo eliminando el fragmento de cruce del padre 1 e insertando el fragmento de cruce en el padre 2 en el punto de cruce del padre 2. El segundo hijo se produce de forma simétrica.

El *fragmento de cruce* para un padre en particular es el subárbol cuya raíz en el punto de cruce.

Ejemplos:

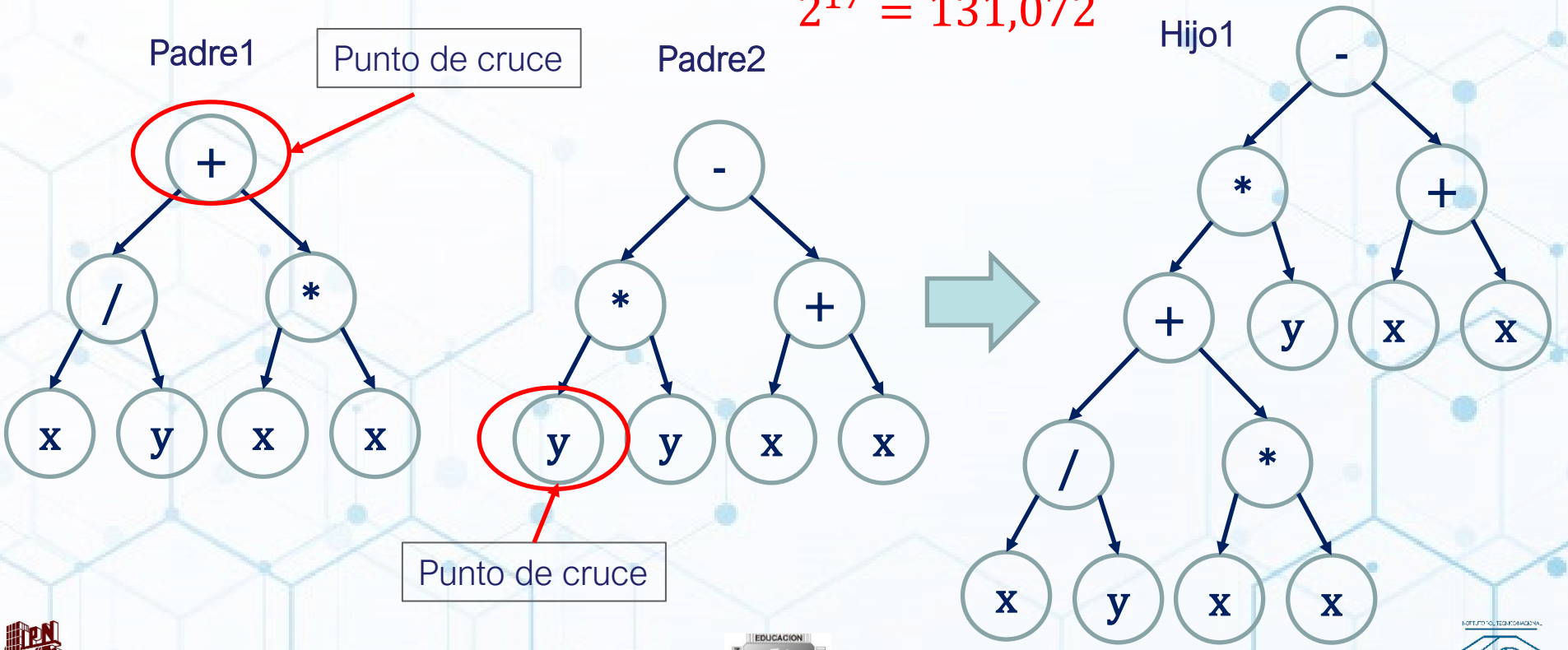




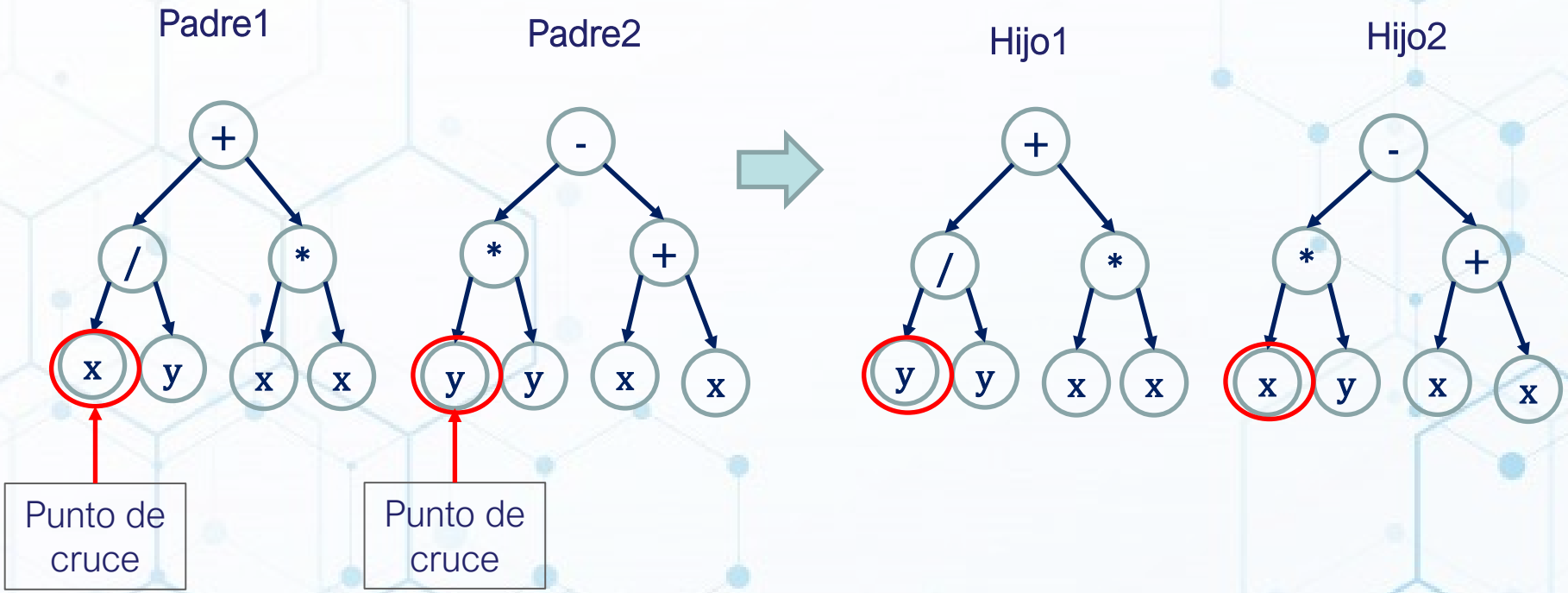
- Existen variaciones donde en lugar de generar 2 hijos solo se genera 1 y el otro se descarta.
- Debido a que se intercambian subárboles completos y debido a la propiedad de clausura, el cruce siempre produce programas sintácticamente válidos.

Cuando el punto de cruce de un padre es una hoja mientras y en el otro padre es el nodo raíz, la descendencia tendrá más profundidad que los padres. Si bien esto puede ser deseable en las primeras etapas de una corrida, en el presencia del fenómeno ***bloat*** puede ser necesario imponer algún límite al tamaño de la descendencia.

$2^{17} = 131{,}072$



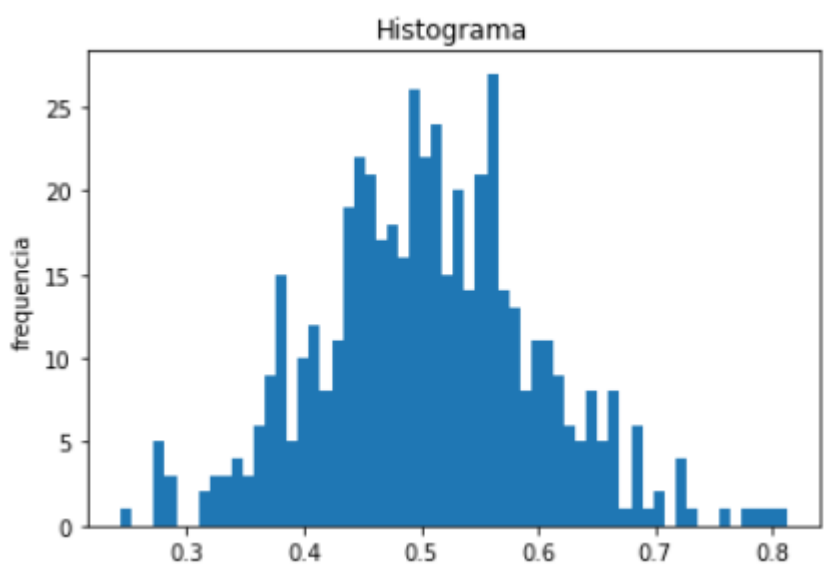
Si los puntos de cruce en cada padre se encuentran en nodos terminales, la operación de cruce simplemente intercambia terminales de un árbol a otro. El efecto de cruce, en este caso, es similar a una mutación puntual. Por lo tanto, la mutación puntual ocasionalmente es una parte de la operación de cruce.



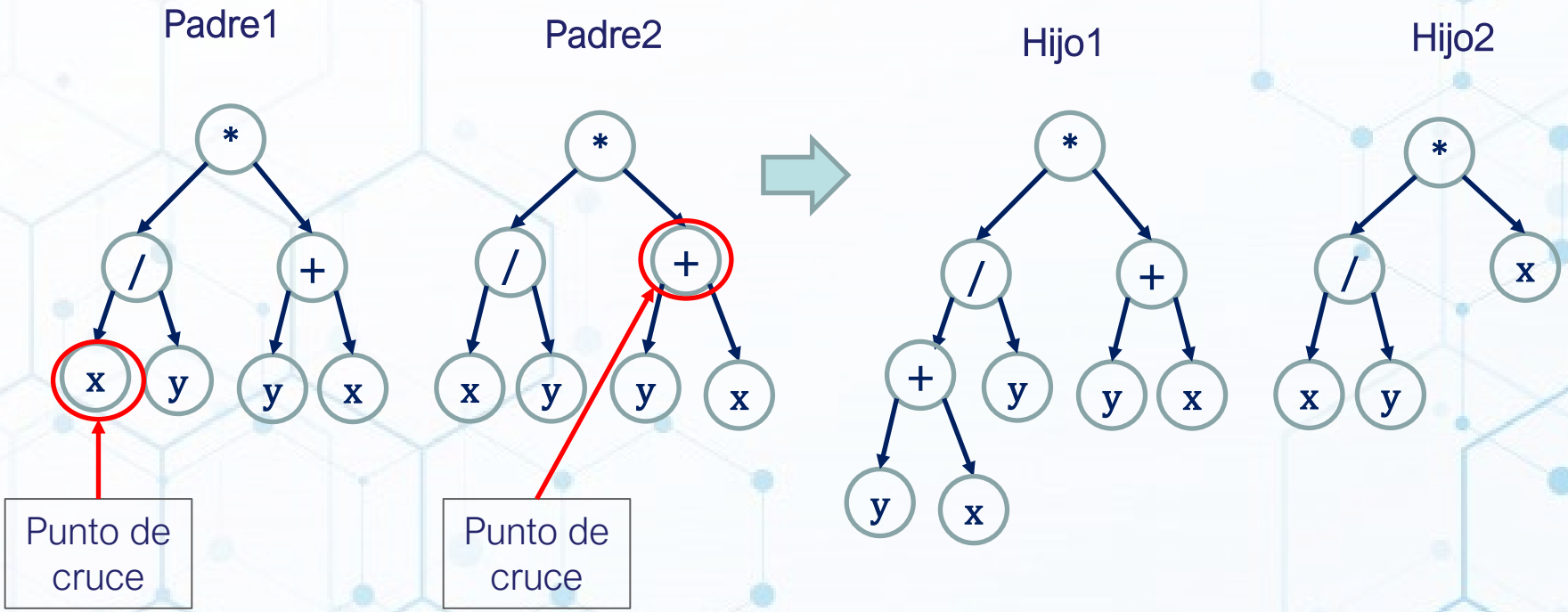
Los puntos de cruce no deberían seleccionarse con probabilidad uniforme. Dado que la mayoría de los nodos serán hojas, la selección uniforme seleccionara más frecuentemente nodos hojas llevando a un intercambio de poco material genético. Para contrarrestar esto Koza sugiere que el 90% de las veces se seleccionen funciones y el 10% restante terminales.

```
mu, sigma = 0.5, 0.1 # media y desvio estandar
datos = np.random.normal(mu, sigma, 500) #creando muestra de datos

# histograma de distribución normal.
cuenta, cajas, ignorar = plt.hist(datos, 60)
plt.ylabel('frequencia')
plt.xlabel('valores')
plt.title('Histograma')
plt.show()
```



Cuando un individuo se aparea incestuosamente consigo mismo o cuando dos individuos idénticos se aparean, los dos descendientes resultantes serán generalmente diferentes (porque los puntos de cruce seleccionados son, en general, diferentes para los dos padres).



Mutación

Es un operador genético asexual por lo que solo requiere la selección de 1 padre y como resultado se tiene un solo hijo.

Probabilidades de mutación bajas o nulas El libro clásico de Koza sobre GP de 1992 [6] aconseja a los usuarios establecer la probabilidad de mutación (P_m) en 0, es decir, sin mutación. Más recientemente, Banzhaf *et al.* recomiendan 5% probabilidad de mutación [7].

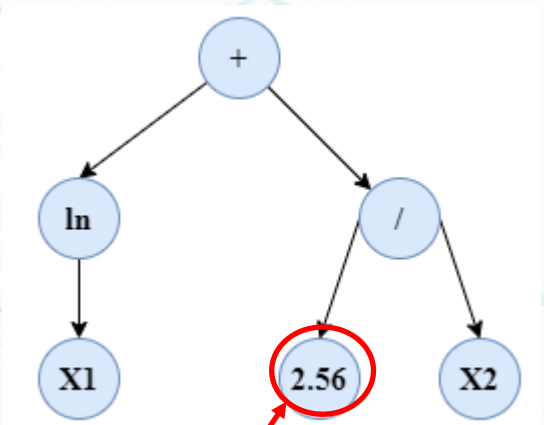
¿Por qué $P_m = 0$?

PG utiliza un número reducido de funciones y terminales, si se compara con el número de cromosomas usados en AG, por lo que es poco probable que desaparezcan completamente de la población.

La cruza en algunos casos funciona como mutación de un punto.

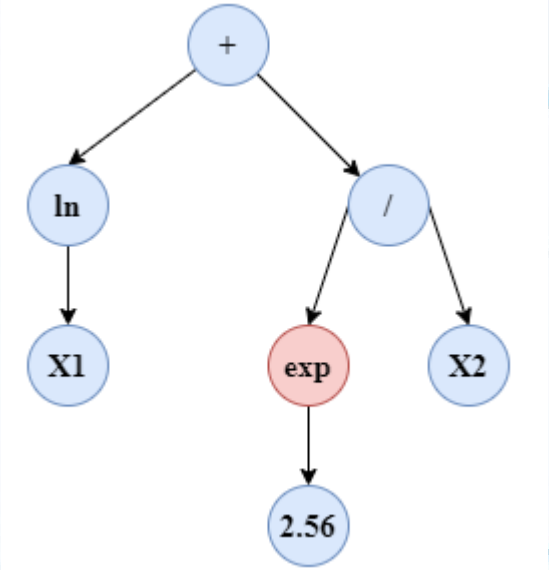
La mutación estándar en PG, a menudo llamada *mutación de subárbol*, comienza eligiendo un punto de mutación al azar, con probabilidad uniforme, dentro del individuo seleccionado. Luego, se elimina el subárbol cuya raíz es el punto de mutación y se inserta un nuevo subárbol generado aleatoriamente en ese punto.

Individuo sin mutar



Punto de mutación

Individuo mutado

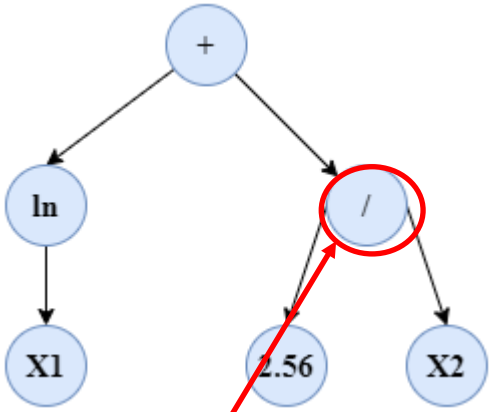


Mutación de subárbol

Un parámetro debe controlar la profundidad del subárbol generado, generalmente es el mismo parámetro que controla la profundidad de los árboles en la generación inicial (d_{max}).

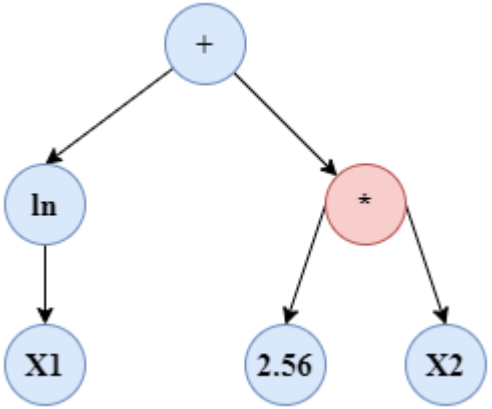
Mutación de 1 punto

Individuo sin mutar



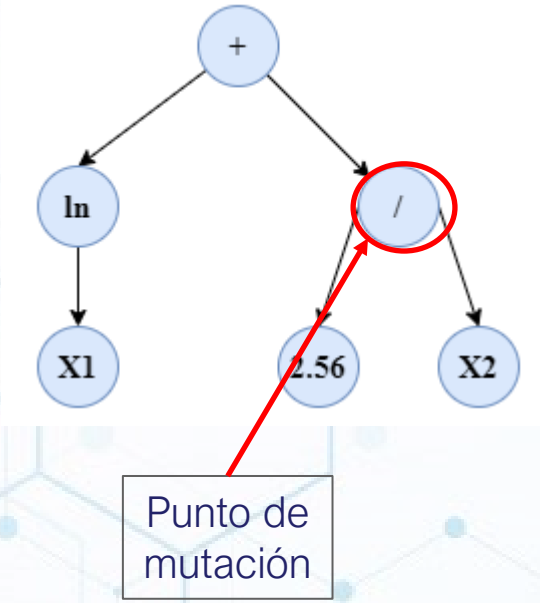
Punto de mutación

Individuo mutado

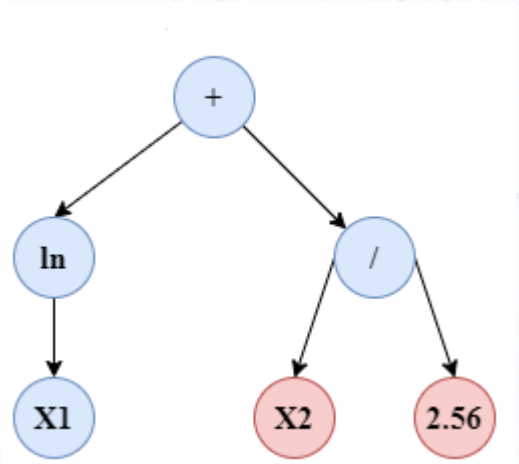


Mutación de intercambio

Individuo sin mutar

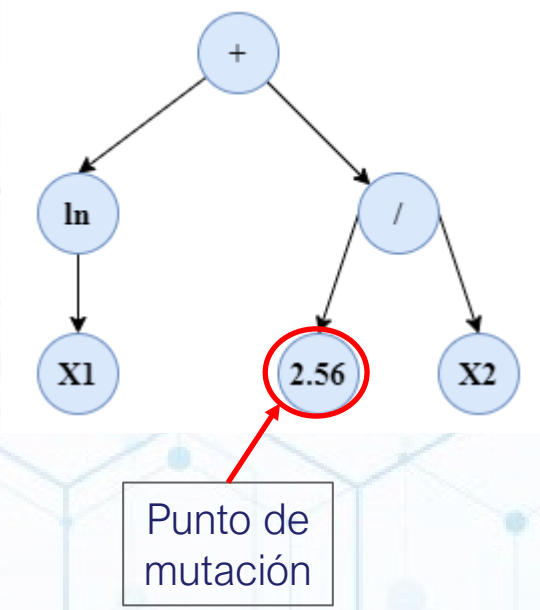


Individuo mutado

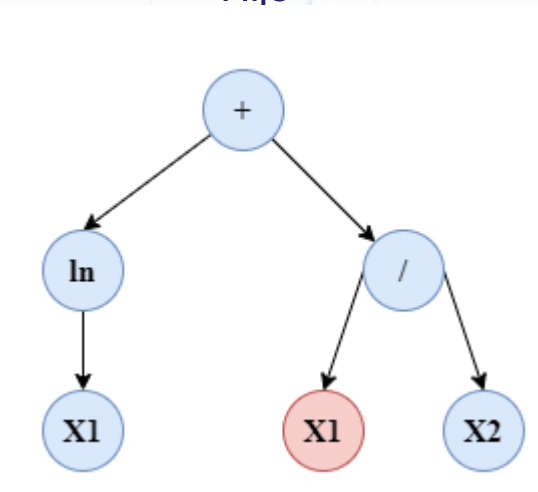


Mutación de 1 nodo

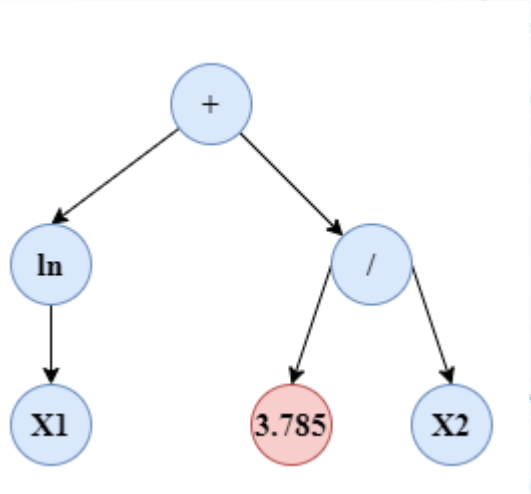
Padre



Hijo

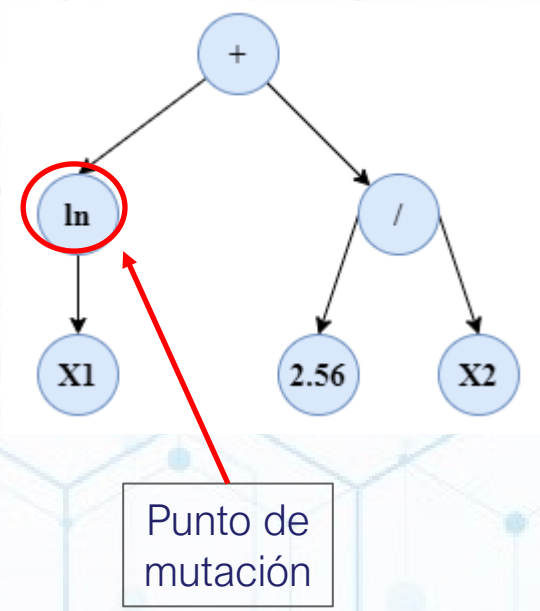


Hijo

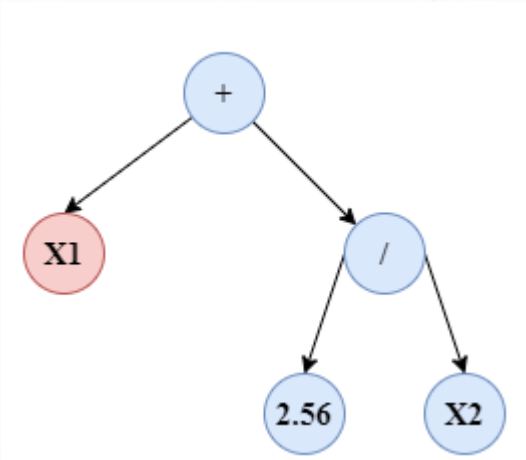


Truncar nodo

Padre



Hijo



La elección de cuál operador genético usar para crear una descendencia es probabilística. Los operadores en PG normalmente son mutuamente excluyentes (a diferencia de otros algoritmos evolutivos donde la descendencia a veces se obtiene a través de una composición de operadores).

Configuración 1

$P_c = 90\%$

$P_r = 10\%$

Configuración 2

$P_c = 90\%$

$P_m = 5\%$

$P_r = 5\%$

$$P_c + P_m + P_r = 1$$

PG Benchmarks

Koza [6] definió un conjunto de problemas que pueden considerarse "problemas típicos de PG", dado que imitan algunas aplicaciones importantes de la vida real y son simples de definir y aplicar a GP. Por esta razón, han sido adoptados por la comunidad de investigación como un conjunto de puntos de referencia [8].

(i). ***Paridad par k:*** encontrar una función booleana de k argumentos que devuelva verdadero si un número par de sus argumentos se evalúa como verdadero y falso en caso contrario.

(ii). ***Regresión simbólica:*** encontrar un programa que coincida con una función matemática objetivo dada.

(iii). **Hormiga artificial (Santa Fe Trail):** encontrar una estrategia de navegación objetivo para un agente en una cuadrícula toroidal limitada.

Aplicaciones

La literatura, que cubre más de 5000 usos registrados de PG, reporta una enorme cantidad de aplicaciones en las que GP se ha utilizado con éxito como una herramienta de programación automática, una herramienta de aprendizaje automático o un motor automático de resolución de problemas. [8].

i. Procesamiento de imágenes y señales.
ii. Aplicaciones financiera, series de tiempo y modelado económico.
iii. Control en procesos industriales.
iv. Medicina y bioinformática.
v. Entretenimiento y juegos de computadoras.

Un ejemplo de aplicación financiera se encuentra en el sitio https://github.com/giladbi/algorithmic-trading/tree/master/Rahul_Genetic_Program donde se usa PG para la predicción del valor de las acciones de Apple. Se evoluciona una población basada en árboles para minimizar el error entre el precio previsto y el precio real.

Serengeti-World-Genetic-Programming

En https://github.com/halucinka/Serengeti-World-Genetic-Programming hay un ejemplo de un aplicación para simular el proceso evolutivo en el mundo biológico con 2 tipos de especies. Solución implementada en java.

Regresión lineal

En https://github.com/rishavray/LinearGP esta una biblioteca desarrollada para java que resuelve problemas de regresión lineal.

Artículos:

1. Khan, S. U., Ullah, N., Ahmed, I., Chai, W. Y., & Khan, A. (2018). **MRI images enhancement using genetic programming based hybrid noise removal filter approach**. Current Medical Imaging, 14(6), 867-873.

https://onlinelibrary.wiley.com/doi/full/10.1111/coin.12459.

2. Salgotra, R., Gandomi, M., & Gandomi, A. H. (2020). **Time series analysis and forecast of the COVID-19 pandemic in India using genetic programming**. Chaos, Solitons & Fractals, 138, 109945.

https://www.sciencedirect.com/science/article/pii/S0960077920303441

Artículos:

3. Chenar, S.S. & Deng,Z. (2018). **Development of genetic programming-based model for predicting oyster norovirus outbreak risks.** Water Research, vol. 128, pp. 20-37.

https://www.sciencedirect.com/science/article/pii/S0043135417308692

4. Ashofteh, P.S., Haddad, O.B., Akbari-Alashti, H., & Marino, M.A. (2015). **Determination of irrigation allocation policy under climate change by genetic programming**. Journal of Irrigation and Drainage Engineering, vol. 141(4).

https://ascelibrary.org/doi/abs/10.1061/(ASCE)IR.1943-4774.0000807

5. Silva, B.M., Bernardino, H.S., & Barbosa, H.J. (2021). **Human Activity Recognition Using Parallel Cartesian Genetic Programming**. In 2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 474-481). IEEE.

Bibliografía

1. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., & Lanza, G. (2006). Genetic programming IV: Routine human-competitive machine intelligence, vol. 5, Springer Science & Business Media.

2. A.L.Samuel. (1983). AI: where it has been and where it is going. In: IJCAI, pp 1152–1157.

3. Poli, R., Langdon, W.B. & McPhee, N.F. (2008). A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd.

4. S. N. Sivanandam & S. N. Deepa. 2008. Introduction to Genetic Algorithms (1st. ed.). Springer Publishing Company, Incorporated.

5. A. E. Eiben and James E. Smith. 2015. Introduction to Evolutionary Computing (2nd. ed.). Springer Publishing Company, Incorporated.

6. Koza, J.R. (1994). Genetic programming as a means for programming computers by natural selection. Statistics and computing, vol. 4(2), pp. 87-112.

7. Banzhaf, W., Nordin, P., Keller, R.E., & Francone, F.D. (1998). Genetic Programming: An Introduction. Morgan Kaufmann, San Francisco.

8. Rozenberg, G., Bäck,T., & Kok, J.N. (2011). Handbook of Natural Computing (1st. ed.). Springer Publishing Company, Incorporated.

¡ Gracias !

Thanks !

Obrigado

Xie xie ni

Domo arigatou

Спасибо

Merci

Grazie