Práctica #3

Diseño de un AG simple

1 DICIEMBRE

Algoritmos Genéticos

María Elena Cruz Meza

Práctica #3 Diseño de un AG simple

Objetivo:

Mostrar el conocimiento, la habilidad, y destreza para identificar y manipular los parámetros y procesos involucrados en los AG; específicamente en el diseño de un AG Simple e identificar las bondandes de los métodos de selección, cruza y mutación.

Objetivos particulares:

- Derivado de la práctica No. 1 y 2, ahora el usuario podrá elegir el tipo de Selección de la población, así como los tipos de operadores para la Cruza y Mutación
- Algoritmo Genético simple
    - Selección proporcional Ruleta
    - Cruza o recombinación
        - Cruza de 1 punto
        - Cruza de 2 puntos
        - Cruza uniforme
    - Mutación
        - Básica: Por intercambio de bit
        - Permutación por:
            - Inserción
            - Desplazamiento
            - Intercambio mútuo
            - Heurística
- Gráficar el desempeño del algoritmo, moratar el mejor y peor indivíduo en cada generación.

Instrucciones

- **Conocimientos previos**
  a) **Haber desarrollado la práctica No.1y 2**
  b) Habilidades en la implementación de vectores,
  c) Habilidades en la implementación e interpretación de histogramas

- **Entrega de evidencia**
  a) Trabajo colaborativo: a partir de esta práctica, el resto de las prácticas y el proyecto serán desarrolaldas y entregadas por equipo. En caso de trabajar solo un integrante, deberás indicar a la maestra.
  b) Medio para su entrega: Entregar la tarea en la plataforma Microsoft Teams una vez terminada toda la práctica completa. Elaborar un informe del desarrollo de esta práctica en un archivo con formato de video (consultar rúbrica para conocer los indicadores de evaluación).
  c) Crear un archivo en formato de video, elija la herramienta de su preferencia y subirlo a alguna plataforma para compartir el link para visualizarlo (drive, Dropbox, youtube, etc). El archivo debe enviarse con una nomenclatura específica y con el siguiente contenido:
    - Nombre al archivo: Si se elige una plataforma donde se tiene acceso al archivo, este deberás nombrarlo como: P3-AGSimple.FOR, donde "P2" indica el número de práctica reportada, "AGSimple" indica que continua el diseño de un AG simple. "FOR" es el formato correspondiente al archivo tipo video, *etc.* Si optas por compartirlo en youtube o alguna herramienta similar, entonces el nombre deberá ser: "Practica #3-AGSimple", y en la descripción correspondiente indicar los nombres de los participantes. Como solo es para fines académicos se sugiere compartirlo en forma privada, pero, tú decides su publicación libre.
    - Portada: Es importante agregar algunos datos al inicio del vídeo: identidad politécnica (Nombre o logos de la unidad académica: IPN y Escom), Nombre de la unidad de aprendizaje, Número de la práctica, Objetivo de la práctica, Nombre de los integrantes del equipo, es opcional mostrar el Nombre del(a) facilitador(A) (profesora) y Fecha de entrega.
    - Marco teórico: El vídeo debe incluir una breve información formal que sustenta el trabajo que se desarrolla (para este caso puede tomar como base o referencia lo que se anexa al final de este documento, notas del curso facilitadas en la plataforma o consultar algún autor), si consulta alguna página, esta debe ser validada por alguna universidad u organismo educativo (papper o artículos, cursos en línea, presentaciones de docentes o investigadores, libros gratuitos en línea o por alguna editorial, etc.).
    - Programa y pruebas: En el vídeo debe explicarse que lenguaje de

programación es utilizado para implementar el algoritmo, explicar los segmentos de código principales, y posteriormente el funcionamiento de este con las funciones de aptitud acorde a los métodos que se le aplicaran para optimizar.

- Al entregar el vídeo también anexa un archivo en formato PDF o TXT, con el código principal completo para su verificación y ponderación respectiva conforme los indicadores marcados en la rúbrica. El nombre de este archivo deberá nombrarse como el archivo de vídeo.
    - En caso de no terminar todos o bien crea que aun tenga fallas en la programación, presentar los que se hayan realizado. Esto es con fin de evitar que en esta tarea quede en ausente (vacío o cero), repercutiendo en tu evaluación final.
- Créditos: Al final del video agregar las referencias consultadas para el desarrollo de la actividad.
- Para trabajos con uno o dos integrantes la duración del vídeo es recomendable entre 5 a 10 minutos. Para equipos de mas de dos integrantes es recomendable no más de 15 minutos.

Fundamentos

El diseño de un Algoritmo Genético Simple, tambien conocido como Canónico (ver figura 1) involucra varios procesos, necesita una codificación o representación del problema, requiere una función de ajuste o adaptación al problema, la cual asigna un numero real a cada posible solución codificada. Luego, durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción (recombinación o cruza), para luego estos padres seleccionados cruzarlos y generen dos hijos, uno por cada padre, y finalmente se elige sobre cuales actuará un operador de mutación. Al terminar el primer ciclo también conocido como generación, la combinación de todos estos operadores proveerán un conjunto de individuos (posibles soluciones al problema), los cuales conforme evolucione el Algoritmo Genético, formarán parte de la siguiente población.

```
t ← 0
población(t) ← poblaciónInicial
EVALUAR(población(t))

while not (condición de terminación)
      t ← t + 1
      población(t) ← SELECCIONAR(población(t-1))
      población(t) ← CRUZAR(población(t))
      población(t) ← MUTAR(población(t))
      EVALUAR(población(t))
      EVALUAR(población(t))

return población(t)
```

Imagen 1. Estructura de un algoritmo genético clásico

Cuando se opta por diseñar un algoritmo genético simple, este se diferenciará del algoritmo estándar, principalmente por el uso de distintas técnicas con los operadores de selección, cruza y mutación. Ahora, continuemos con el diseño de un Algoritmo Genético Simple, con otros operadores de selección, cruza y mutación.

1. MECANÍSMOS DE SELECCIÓN

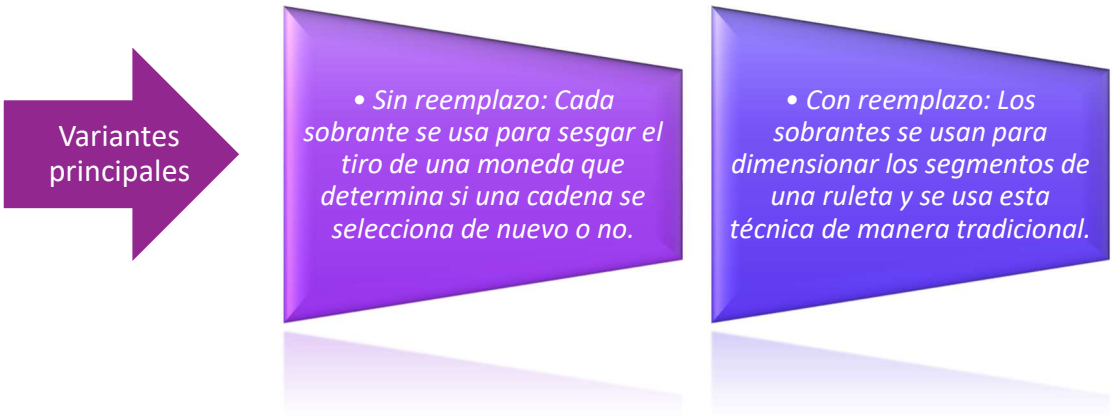
Un algoritmo genético puede utilizar muchas técnicas diferentes para seleccionar a los individuos que deben copiarse hacia la siguiente generación. Aquellos individuos que compiten por los recursos de forma más efectiva tienen más posibilidades de reproducirse. La selección suele considerar la población en su conjunto y la probabilidad de reproducción se define sobre la generación actual de la población. Se asignan probabilidades de selección a los individuos de la población (padres) en función de su fitness. Sin embargo, se suele emplear un mecanismo de selección estocástico:

- Los mejores individuos es más probable que se seleccionen (aunque nada lo garantiza).
- Incluso el peor individuo de la población puede ser seleccionado.
- La selección estocástica nos ayuda a escapar de óptimos locales.

a. Método de Selección por Sobrante Estocástico

Propuesta por Booker [24] y Brindle [32] como una alternativa para aproximarse más a los valores esperados (Valesp) de los individuos. El operador actúa así:

1. Asignar de manera determinística el conteo de valores esperados a cada individuo (valores enteros).
2. Los valores restantes (sobrantes del redondeo) se usan probabilísticamente para rellenar la población.



Caso 1. Sin reemplazo

- Supongamos el problema – adaptado de (Goldberg. 1989) – de encontrar el máximo de la función $f(x) = x^2$ sobre los enteros {1, 2, . . . , 32}.

1.- Inicialización, evaluación y selección de padres.

No.	Población Inicial (fenotipos)	x Valor (genotipo)	f(x) (F(x)=x2) (Fitness o función de adaptación)	f(x)/∑f(x) Probabilidad de selección	Enteros	diferencia
1	110100		220	1.23	1	0.23
2	011010		140	0.78	0	0.78
3	111001		315	1.76	1	0.76
4	001101		42	0.23	0	0.23
Suma			717	4.00	2	
Media			179.25	1.00		
Mejor			315	1.76		
Selección: padres ind(1) e ind(3) (parte entera)						

Hacer flip(f(x)/∑f(x)), hasta tener el número de padres requeridos

Caso 2.Con reemplazo

- Supongamos el problema – adaptado de (Goldberg. 1989) – de encontrar el máximo de la función $f(x) = x^2$ sobre los enteros {1, 2, . . . , 32}.

1.- Inicialización, evaluación y selección de padres.

No.	Población Inicial (fenotipos)	x Valor (genotipo)	f(x) (F(x)=x2) (Fitness o función de adaptación)	f(x)/∑f(x) Probabilidad de selección	enteros	diferencia
1	110100		220	1.23	1	0.23
2	011010		140	0.78	0	0.78
3	111001		315	1.76	1	0.76
4	001101		42	0.23	0	0.23
Suma			717	4.00	2	
Media			179.25	1.00		
Mejor			315	1.76		

Armar una ruleta

Caso 2.Con reemplazo

- Ejecutando la ruleta

Evaluación de padres:

No.	f(x)/∑f(x) (Diferencia)	% del total	%
1	0.23	0.115	11.5%
2	0.78	0.39	39%
3	0.76	0.38	38%
4	0.23	0.115	11.5%
Suma	2.00	1.00	

Selección por ruleta elegiría:

(ind1) suma = 0.12 < r
(ind2) suma = 0.39 > r

➔ Ya que es el primero que supera a r

2. MECANÍSMOS DE CRUZA

Combinan la información de los padres para crear nuevos descendientes

- La selección de qué información de los padres se combina es estocástica.
- Muchos descendientes pueden ser peores que los padres (en términos de la función de fitness).
- Se espera que algunos de ellos sean mejores al combinar los elementos de sus genotipos que conducen a la obtención de mejores fenotipos
- Los métodos de cruza suelen aplicarse a la representación binaria, estas son generalizables a alfabetos de cardinalidad mayor.

Tipos de métodos de cruza o recombinación:

- Basados en las frecuencias de los alelos (votación p-sexual, generalización del cruce uniforme)
- Basados en la segmentación de los padres (cruce diagonal, generalización del cruce en n puntos)
- Basados en operaciones numéricas sobre alelos con valores reales (p.ej. centro de masas)

a. Cruza Uniforme

Propuesta por Ackley Syswerda, se ve como una cruza de n puntos (Figura 1), aquí el número de puntos de cruza no se fija previamente.

La herencia de un gen es ahora independiente de su posición en el cromosoma.

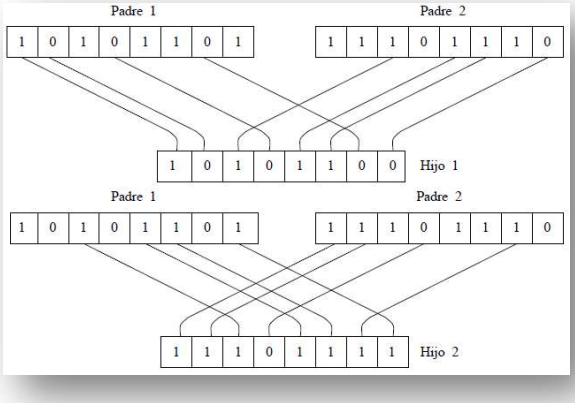


Figura 1: Ejemplo de Cruza uniforme.

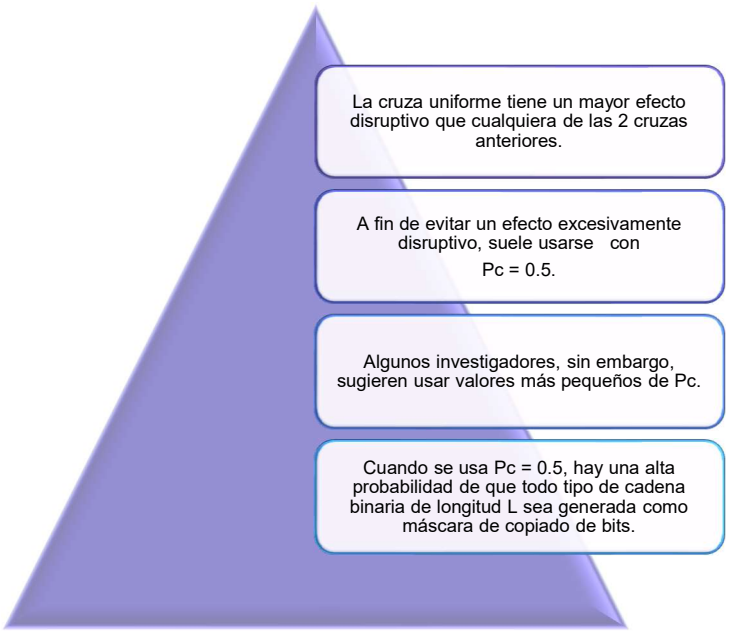
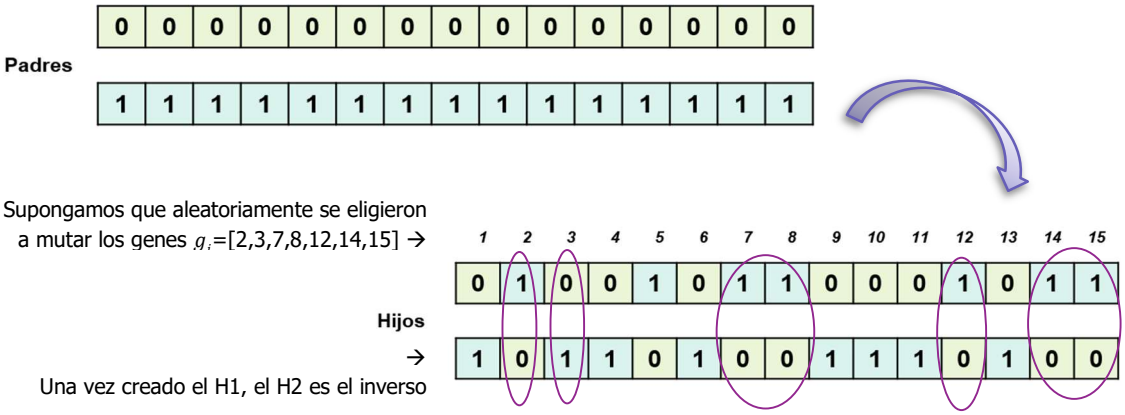


Figura 2. Algunas consideraciones para la implementación del método de cruza uniforme

Algoritmo

1. Se elige al azar el padre del que proviene cada gen para el primer hijo
    a. Generar los hijos, uno por cada padre

Ejemplo:



3. MECANÍSMOS DE MUTACIÓN PARA PERMUTACIONES

Una vez que se han creado los individuos descendientes, algunos de estos se someterán a un proceso de mutación en el que, cada una de sus posiciones, puede verse modificada con una probabilidad p. Este paso es importante para añadir diversidad al proceso y evitar que el algoritmo caiga en mínimos locales porque todos los individuos sean demasiado parecidos de una generación a otra. Así, encontramos que existen diferentes estrategias para controlar la magnitud del cambio que puede provocar una mutación:

- ✓ Distribución uniforme: la mutación de la posición i se consigue sumándole al valor de i un valor extraído de una distribución uniforme, por ejemplo una entre [-1,+1].

- ✓ Distribución normal: la mutación de la posición i se consigue sumándole al valor de i un valor extraído de una distribución normal, comúnmente centrada en 0 y con una determinada desviación estándar. Cuanto mayor la desviación estándar, con mayor probabilidad la mutación introducirá cambios grandes.

- ✓ Aleatorio: la mutación de la posición i se consigue reemplazando el valor de i por nuevo valor aleatorio dentro del rango permitido para esa variable. Esta estrategia suele conllevar mayores variaciones que las dos anteriores.

1. Mutación por intercambio recíproco

Este método permite preservar casi toda la información de adyacencia (4 links rotos), pero el orden se modifica más (en el método se rompían tres links)

Algoritmo

1. Elegir dos genes al azar
2. Intercambiar sus alelos

Ej. Supongamos el siguiente padre, un cromosoma de representación entera:

P= 153426789

Genes elegidos →

1	2	3	4	5	6	7	8	9

Intercambiando posiciones →

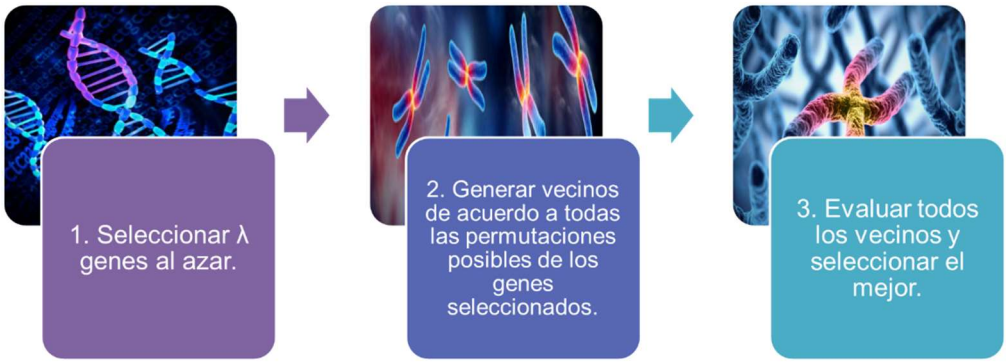
1	5	3	4	2	6	7	8	9

Descendencia mutada →

1	5	3	4	2	6	7	8	9

2. Mutación Heurística

Método propuesto por Gen y Chen, el algoritmo consta de tres pasos:



Donde λ es el número de bit (genes) a elegir aleatoriamente.

Ej. Consideremos el siguiente padre, un cromosoma de representación entera:

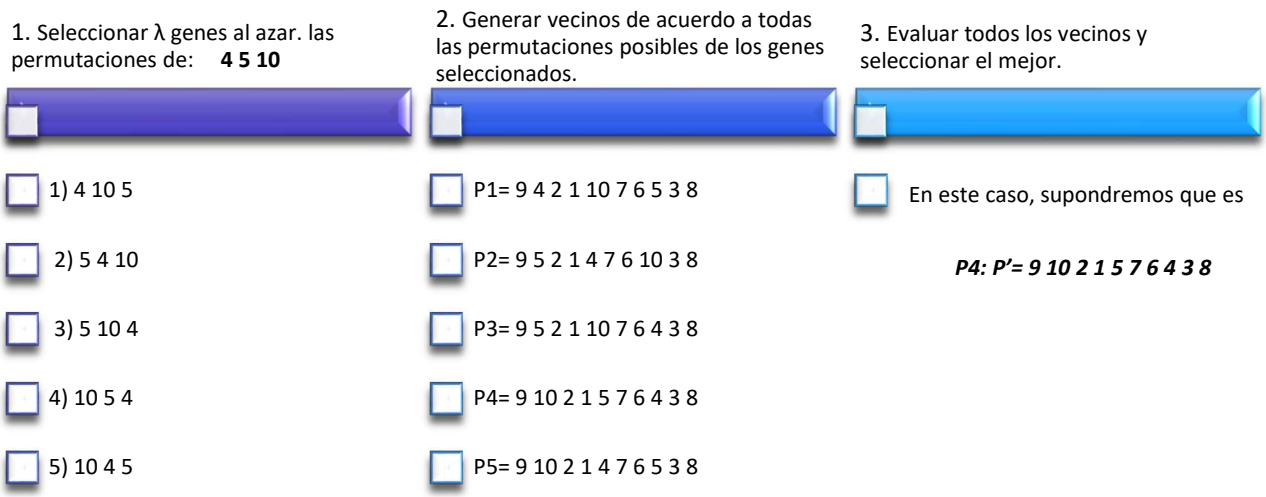
P= 9 4 2 1 5 7 6 10 3 8

De acuerdo con el algoritmo, debemos hacer la selección del número de genes a permutar, asignando estos a λ

i. Si al azar se eligen tres genes (λ =3)

ii. λ = 2, 5, 8, los alelos correspondientes son: 4, 5 y 10

Los pasos completos se muestran en el siguiente esquema:



Fuentes consultadas

- Sancho C. F. Blogs:"Temas selectos". Profr. en el Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla. Fecha de consulta: Agosto/2018. Disponible en: http://www.cs.us.es/~fsancho/?e=65.
- Beasley et al., 1993. Beasley, D., Bull, D. R., and Martin, R. R. (1993). An overview of genetic algorithms: Part 1,fundamentals. niversity Computing, 15(2):58-69.
- Blickle, T. and Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. Technical Report 11, Computer Engineering and Comunnication Network Lab (TIK), Gloriastrasse 35, 8092 Zurich, Switzerland.
- Darwin, C. (1859). On the Origin of Species by Means of Natural Selection. John Murray, London.
- Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). Artificial Intelligence through Simulated Evolution. John Wiley & Sons, New York.
- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor. Republished by the MIT press, 1992.
- Jong, K. A. D. (1975). An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan.

- Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). The MIT Press.
- Rechenberg, I. (1973). Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.
  Frommann-Holzboog, Stuttgart.
- Tomassini, M. (1995). A survey of genetic algorithms. Annual Reviews of Computational Physics, III:87-118.