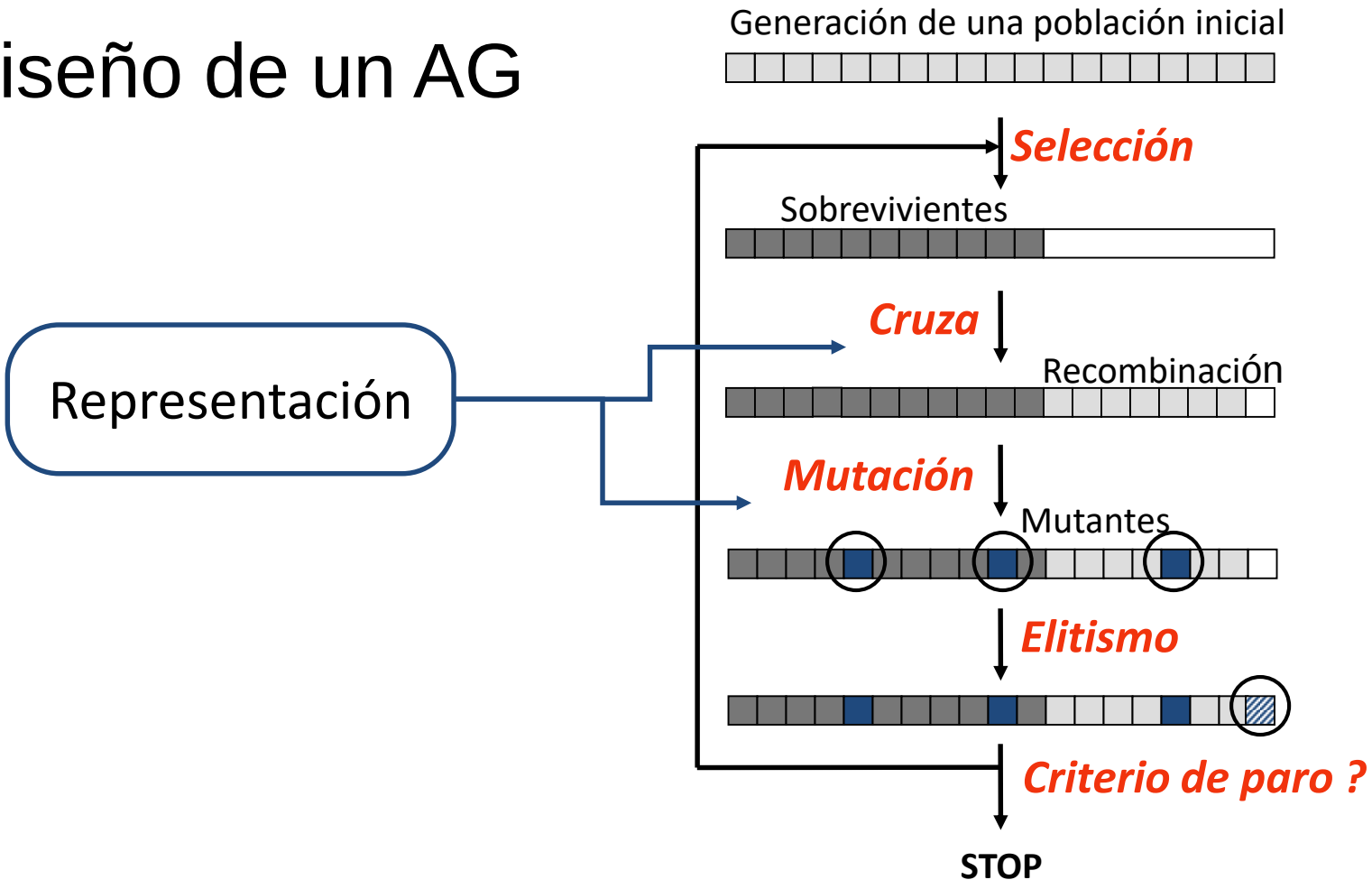
AG: Importancia de la codificación

Introducción

Diseño de un AG



Introducción

- Técnicas clásicas de representación
  - Cadenas binarias (tradicional)
  - Códigos de Gray (binaria)
  - Punto flotante (binaria)
  - Punto flotante (real)
  - Punto flotante (entera)
  - Listas Binarias de Longitud Variable (*messy-GA*)
  - Representación de árbol
  - Híbridos (AG estructurado)

Cadenas binarias

- Representación tradicional usada para codificar un conjunto de soluciones
  - Cromosoma: cadena de la forma $<b_1, b_2, \ldots, b_m>$
  - $b_1, b_2, \ldots, b_m$ = alelos (0 ó 1)
- Motivaciones para el uso de cadenas binarias (a parte del factor histórico)
  - Representación universal
  - Argumento teórico: teorema de los esquemas

Cadenas binarias

- Holland (1975) compara el desempeño de AGs con
  - Cadenas largas con pocos alelos (binaria) 2^{80}
  - Cadenas cortas con muchos alelos (decimal) 10^{24}
  - El número de esquemas que se pueden generar de una cadena es $(c+1)^l$ donde
    - c: cardinalidad del alfabeto usado
    - l: longitud de la cadena
- Por tanto
  - una cadena binaria genera más esquemas, 3^{80},que una cadena decimal, 11^{24}.
  - una cadena de tipo binario da pie a un grado mayor de paralelismo implícito que una cadena de tipo decimal

Cadenas binarias

- Paralelismo implícito
  - Estimando las aptitudes de los individuos en una población, un AG estima de forma implícita las aptitudes promedio de un número mucho más alto de cadenas cromosómicas (aptitudes promedio de los “bloques constructores”)
  - Contar con más esquemas favorece la diversidad e incrementa la probabilidad de que se formen buenos “bloques constructores” en cada generación, lo que en consecuencia mejora el desempeño del AG con el paso del tiempo

Cadenas binarias

- Justificación biológica
  - En genética es más usual tener cromosomas con muchas posiciones y pocos alelos por posición que pocas posiciones y muchos alelos por posición

- Finalmente:
  - Uso de alfabetos de mayor cardinalidad funciona, pero el alfabeto binario es el que ofrece el mayor número de esquemas posibles por bit de información
  - Largo debate sobre alfabetos no binarios: el uso de la representación binaria tiene varias desventajas cuando el AG se usa para resolver ciertos problemas del mundo real

Cadenas binarias

- Principales desventajas de una codificación binaria
  - Epístasis, el valor de un bit puede suprimir las contribuciones de aptitud de otros bits en el genotipo.
  - Representación natural: algunos problemas (como el del viajero) se prestan de manera natural para la utilización de representaciones de mayor cardinalidad que la binaria
  - Soluciones ilegales : los operadores genéticos utilizados pueden producir con frecuencia (e incluso todo el tiempo) soluciones ilegales si se usa una representación binaria

Cadenas binarias

- Principales desventajas de una codificación binaria
  - El cambio en un gene puede ocasionar igualmente cambios importantes o mínimos sobre el valor decodificado del mismo
  - Escalabilidad, optimizar una función con alta dimensionalidad (50 variables), y trabajar con una buena precisión (cinco decimales), entonces el mapeo de números reales a binarios generará cadenas muy largas (1000 bits), y el AG tendrá problemas para producir resultados aceptables.

Códigos de Gray

Distancia de Hamming

- La distancia de Hamming (Richard Hamming), introdujo el término para establecer una métrica capaz de establecer un código para la detección y auto-corrección de códigos.
- Se emplea en la transmisión de información digitalizada para contar el número de desvíos en cadenas de igual longitud y estimar el error

  Por ejemplo:

  - La distancia entre 1011101 y 1001001 es 2.
  - La distancia entre 2143896 y 2233796 es 3.
  - La distancia entre "tener" y "reses" es 3.

Códigos de Gray

- Un problema detectado desde los inicios de los AG's, la representación binaria no mapea adecuadamente el espacio de búsqueda con el espacio de representación
  - Ejemplo: 0111 = 7

    1000 = 8
  - Diferencia de 1 en el fenotipo y de 4 en el genotipo (distancia de *Hamming*)

⇒ *Risco de Hamming* (*Hamming cliff*)

Códigos de Gray

- Varias propuestas de representación alternativa en la que se preserve la propiedad de adyacencia del espacio de búsqueda al espacio de representación
- El código Gray (Frank Gray), es un sistema de numeración binario en el que dos valores sucesivos difieren solamente en uno de sus dígitos.
- Para convertir un número binario a código Gray, se aplica una operación XOR con el mismo número desplazado un bit a la derecha.

Códigos de Gray

- El operador lógico disyunción exclusiva, XOR ó $\oplus$, es una disyunción lógica de dos operandos que es verdad si sólo un operando es verdad pero no ambos.

$$0 \oplus 0 = 0$$

$$1 \oplus 0 = 1$$

$$0 \oplus 1 = 1$$

$$1 \oplus 1 = 0$$

Códigos de Gray

Codificación de Gray

Decimal	Binario	Código de Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111

Códigos de Gray

- Procedimiento de conversión “Binario a Gray”
  - Se asume: $b =< b_1, ..., b_m >$ es un número binario

    y $g =< g_1, . . . , g_m >$ es un número de Gray

```
procedure Binario-a-Gray
begin
        g_1 = b_1
        for k = 2 to m do
                g_k = b_{k-1} XOR b_k
end
```

Códigos de Gray

- Procedimiento de conversión “Gray a Binario”
  - Se asume: $b =< b_1, ..., b_m >$ es un número binario

    y $g =< g_1, . . . , g_m >$ es un número de Gray

```
procedure Gray-a-Binario
begin
        valor = g_1
        b_1 = valor
        for k = 2 to m do
        begin
                if g_k = 1 then valor = NOT valor
                b_k = valor
        end
end
```

Códigos de Gray

- Procedimiento de conversión “Gray a Binario”
  - Definimos un vector g con los dígitos en gray y otro vector b destinado a los dígitos en Base 2.
  - g_0 es el dígito que se encuentra en el extremo izquierdo de la representación en código gray.
  - b_0 es el dígito que se encuentra en el extremo izquierdo en la representación Base 2.
  - $b_0 = g_0$
  - Para $n > 0$: $b_{n+1} = g_{n+1} \oplus b_n$

Códigos de Gray

- Algunos investigadores han demostrado empíricamente el beneficio del uso de códigos de Gray
  - Mejora del desempeño del AG al aplicarse a las funciones de prueba clásicas de De Jong
  - Incluso se ha demostrado que la codificación de Gray altera el número de óptimos locales en el espacio de búsqueda así como el tamaño de las buenas regiones de búsqueda (Mathias & Whitley, 1994)

    (un *hill-climber* con códigos de Gray funciona muy bien con funciones de prueba)

Codificación de números reales

Números reales / punto flotante

- Los códigos de Gray son útiles para representar enteros
- El problema de mapear correctamente el espacio de búsqueda es más complicado con números reales.
- En el enfoque tradicional se usa un número binario para representar un número real.
- Se definen límites inferiores y superiores para cada variable, así como la precisión deseada.

Números reales / punto flotante

- Varias técnicas
  - Cadenas binarias
  - Notación IEEE
  - Codificación real
  - Codificación entera

… y un intenso debate …

Cadenas binarias

Números reales / punto flotante

- Cadenas binarias: descripción
  - Discretizar el rango de variación de la variable considerada
  - Fijar la precisión para determinar el número de bits requeridos
    - Ejemplo: codificar una variable entre 0.35 y 1.40, con una precisión de dos decimales.
    - Necesitaríamos $\log_2(140 - 35) \approx 7$ bits para representar un número real dentro de ese rango.
    - Fórmula general: tamaño = (*int*) $\log_2[(v_{sup} - v_{inf})*10^{precisión}] + 1$
  - Nota: como en el caso de codificación de enteros, se pueden ocupar códigos de Gray

Números reales / punto flotante

- Cadenas binarias: características
  - Descripción compacta
  - Decodificación sencilla
  - Alta dimensionalidad: si una buena precisión es requerida, implica cadenas muy largas

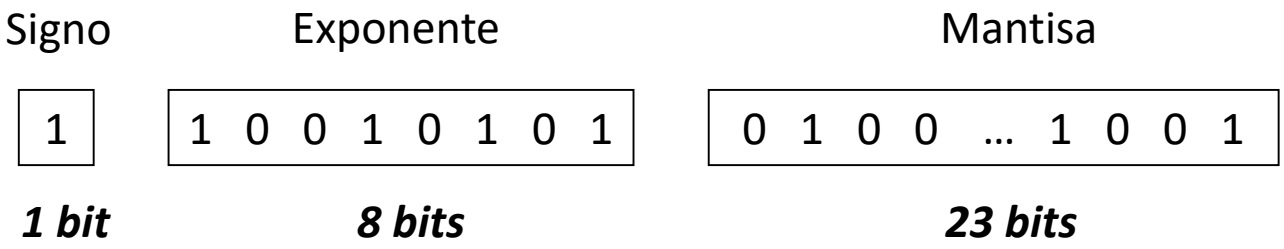
    ➡ pobre desempeño del AG

    (efecto combinatorio muy fuerte)
  - No reproduce necesariamente la “gradualidad” de funciones continuas (uso de códigos de Gray)

Representación IEEE

Números reales / punto flotante

- Representación estándar IEEE: descripción



  - Signo: 0 indica "+", 1 indica "–"
  - Exponente: notación en exceso = 2^n-1= 127

    ⇒ exponente entre -126 y 127
  - Mantisa: codifica "*fracción*" en "1.*fracción*"

Números reales / punto flotante

- Representación estándar IEEE: características
  - Mapeo genotipo $\leftrightarrow$ fenotipo muy compleja

    ⇒ decodificación muy costosa

    (mucho más que para una codificación binaria)
  - Impacto (sobre el fenotipo) muy variable de un cambio según la posición del bit modificado

Codificación entera

Números reales / punto flotante

- Uso de enteros: opción 1
  - Adoptando una posición fija para el punto decimal en cada variable (pero puede variar de una variable a otra)

  - Poca adaptabilidad (codificación específica para cada variable)
  - Precisión limitada por la longitud de la cadena
  - Posibilidad de usar los operadores de cruza tradicionales; mutación sencilla

Números reales / punto flotante

- Uso de enteros: opción 2
  - Un entero largo para cada variable (posición del punto decimal fijo en cada variable)

1365947	259847	489512

➡ codifica: 1.365947 ; 2.59847 ; 489.512

  - Ganancia en términos de memoria (almacena enteros en vez de reales)
  - Necesidad de usar operadores genéticos específicos
  - Sacrificios importantes con respecto a versatilidad de la representación

Codificación real

Números reales / punto flotante

- Representación real
  - El gene codificado (espacio de representación) es igual a la variable (espacio de búsqueda)

3.523	-6.890	9.152	-2.507

  - Esta técnica, en realidad, opera al nivel *fenotípico*
  - Buen grado (obviamente) de precisión, sin incrementar el aspecto combinatorio
  - Necesidad de definir operadores genéticos adaptados

Números reales / punto flotante

- Teóricos y prácticos: puntos de vista opuestos
  - Teóricos afirman que alfabetos pequeños (y cadenas largas) producen mejores resultados
    - Con alta cardinalidad, se volvería más errático/difícil de predecir el comportamiento del AG
  - Prácticos han demostrado, a través una gran cantidad de aplicaciones, que el uso directo de números reales mejora el desempeño del AG
    - Intentos para emular el efecto de cruza/mutación en alfabetos binarios
    - Capacidad de la repr. real de explotar la “gradualidad” de las funciones de variables continuas
    - En este sentido, las codificaciones mediante enteros pueden verse como un compromiso entre codificaciones binaria y real

Codificación de permutaciones

Codificación de permutaciones

- Numerosas aplicaciones
  - TSP
  - Programaciónde producción (*job*/*flow-shop*)
  - Coloración de grafos

- En un Algoritmo Genético, la codificación puede efectuarse
  - "tal cual" (fenotipo = genotipo)

    ⟹ formulación de operadores de variación adaptados
  - Con codificaciones adaptadas

Codificación de permutaciones

- Mediante cadenas binarias: prioridades binarias
  - Propuesta de Nakano y Yamada (1991)
  - Uso de variables binarias para especificar el orden
    - $x_{ij} = 0$ implica $j \rightarrow i$
    - $x_{ij} = 1$ implica $i \rightarrow j$
    - Permutación de n elementos: $n(n-1)/2$ variables
- Producción de permutaciones inválidas
  - Proceso de armonización local

$$M = \begin{bmatrix} - & 1 & 1 & 1 & 1 \\ 0 & - & \times^{1} & 1 & 1 \\ 0 & \times^{0} & - & 0 & 1 \\ 0 & 0 & 1 & - & 1 \\ 0 & 0 & 0 & 0 & - \end{bmatrix} \quad S_j = \begin{bmatrix} 4 \\ \times \; 3 \\ \times \; 1 \\ 2 \\ 0 \end{bmatrix}$$

Codificación de permutaciones

- Mediante cadenas de reales
  - Llaves aleatorias (Bean, 1994): permutación determinada según el orden creciente de las variables asociadas

    Ex: [0.41 0.68 0.02 0.85 0.37]

    { 2 4 0 1 3 }
  - Prioridades binarias aplicadas con redondeo al valor {0 , 1} más cercano

Conclusiones

- Tendencias futuras
  - Uso de una representación adecuada simplifica enormemente el proceso de búsqueda
  - Importancia de conocer las estrategias existentes (distintas a la binaria tradicional)
  - Ahorro de muchos esfuerzos en el diseño de operadores genéticos especiales

Conclusiones

- Recomendaciones de Palmer (1994) para el diseño de una buena codificación
  - Capacidad de representar todos los fenotipos posibles
  - Ausencia de sesgos (todos individuos igualmente representados en el conjunto de los genotipos posibles)
  - No codificar soluciones infactibles
  - Facilidad para decodificación
  - Característica de localidad (cambios pequeños en genotipo cambios pequeños en fenotipo)

Conclusiones

- Recomendaciones de Roland (1997)
  - Ajustó a un conjunto de operadores genéticos preservando los buenos bloques constructores de padres a hijos
  - Minimización de epístasis
  - Preferencia a soluciones factibles
  - Mapeo apropiado genotipo - fenotipo
  - Evitar formas isomórficas (fenotipo codificado con más de un genotipo) (aunque ha sido debatido)