Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

4ª Grado en Ingeniería Informática, 2020-21

DOCENCIA VIRTUAL

Finalidad:

Prestación del servicio Público de educación superior (art. 1 LOU)

Responsable:

Universitat Politècnica de València.

Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento conforme a políticas de privacidad:

http://www.upv.es/contenidos/DPD/

Propiedad intelectual:

Uso exclusivo en el entorno de aula virtual.

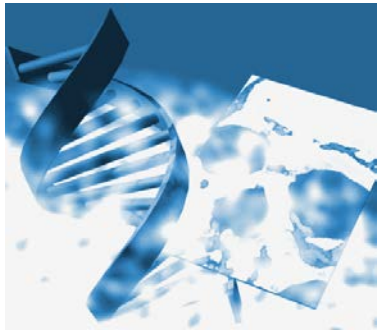
Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.

La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa o civil

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Tema 5.2.- Metaheurísticas Poblacionales (evolutivas): Algoritmos Genéticos

1.- Algoritmos Genéticos. Fundamentos y Conceptos

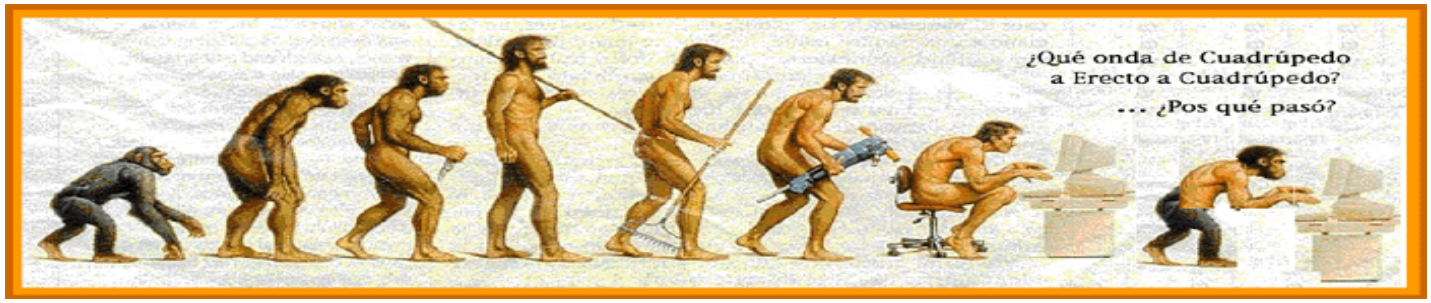


2.- Estructura de un Algoritmo Genético

3.- Etapas de un Algoritmo Genético:

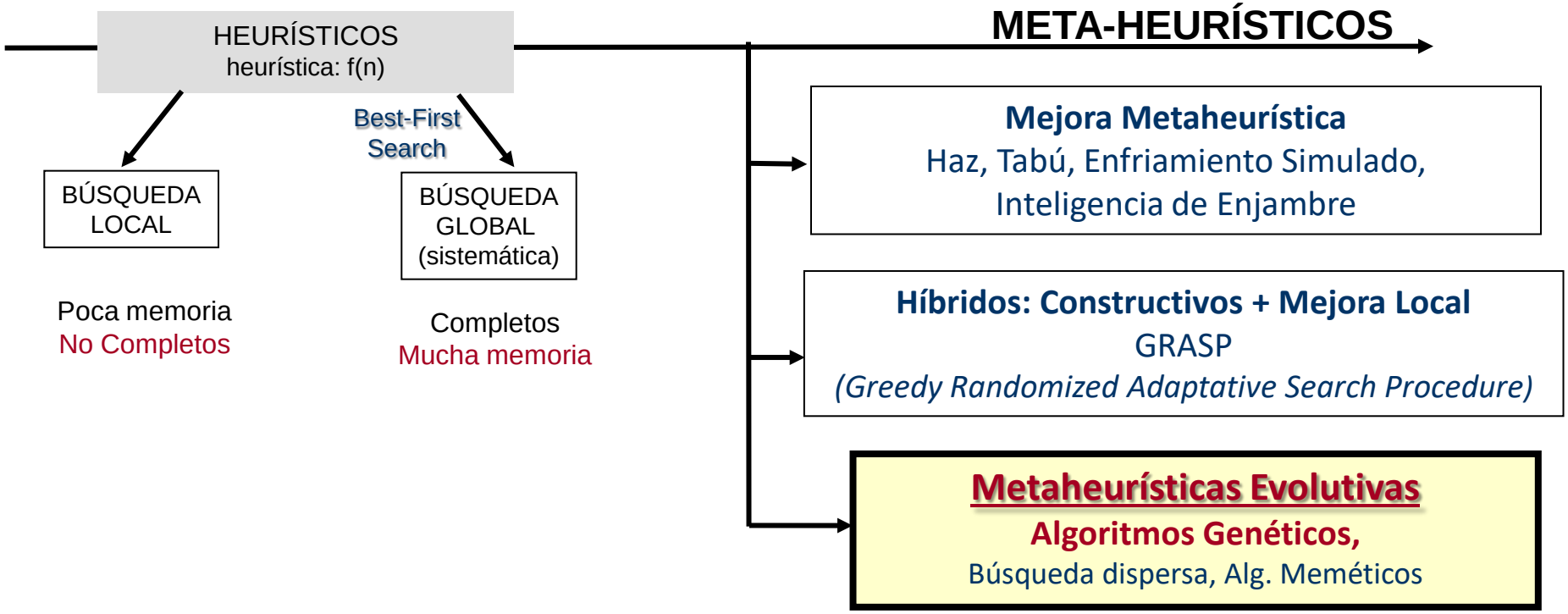
- Representación, Codificación en un AG
- Población Inicial
- Aptitud y fitness
- Estrategia de selección
- Cruce, Mutación y Reemplazo de la Población
- Condición de Parada

4.- Ejemplos



Bibliografía

- **Inteligencia Artificial. Técnicas, métodos y aplicaciones (cap. 11). Palma, Marín (eds). Mc Graw Hill (2008).**
- **Monografía: Metaheurísticas.** Inteligencia Artificial, Vol 7, No 19 (ed. B. Melián, J.A. Moreno Perez, J. Marcos Moreno-Vega) (2003)
- **"Evolutionary Algorithms for Solving Multi-Objective Problems"**. Coello, C.A. Veldhuizen, D.V., Lamont, G.B. Kluwer Academic, NY, 2002
- **Handbook of Evolutionay Computation**, T. Bäck, D.B. Fogel, Z. Michalewicz. Oxford University Press, 2007
- **Genetic Algorithms in Search, Optimization and Machine Learning**. D.E. Goldberg, Addison Wesley, 1989
- **Genetic Algorithm + Data Structures = Evolution Programs.** Z. Michalewicz, Springer Verlag, 1996.
- www.genetic-programming.org, www.geneticprogramming.com



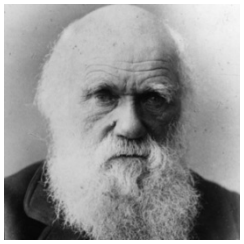
- ✓ Parten de un conjunto de soluciones que se van reconstruyendo y mejorando.
- ✓ Mejoran la búsqueda local, sin la complejidad espacial/temporal de una búsqueda global.
- ✓ Existe una Colaboración y Competición entre soluciones (individuos) mediante las operaciones de cruce y selección. Tiene una componente estocástica.
- ✓ Hay un control centralizado. Los individuos no son autónomos (Alg. Genéticos).
- ✓ Comportamiento any-time: mejora sucesiva de las soluciones.
- ✓ Implementación sencilla. Dificultad: Diseño, Heurística y Ajuste

Conceptos generales.

AG: mejora colectiva de individuos con control centralizado

Ideados por John Holland en 1975,

Inspirándose en la evolución natural de los seres vivos (Darwin).



Idea subyacente:

- La población evoluciona de forma natural, mejorando su adaptación.
- En el transcurso de la evolución se generan poblaciones sucesivas, con información genética de los padres previos, y cuya adecuación al entorno va mejorando sucesivamente.

Los AG se inspiran en los procesos de Evolución Natural y Genética (métodos bioinspirados):

- Un AG evoluciona a partir de una población de soluciones inicial, intentando producir nuevas generaciones de soluciones que sean mejores que la anterior.
- El proceso evolutivo es guiado por decisiones probabilísticas *(componentes aleatorios)* basados en la adecuación de los individuos (fitness). El fitness puede tener una componente heurística.
- Al final del proceso se espera obtener un buen individuo: buena solución global al problema.
- Orientados hacia la resolución (optimización) de problemas combinatorios.
- Los AG son el método metaheurístico más aplicado y con más relevantes resultados prácticos.

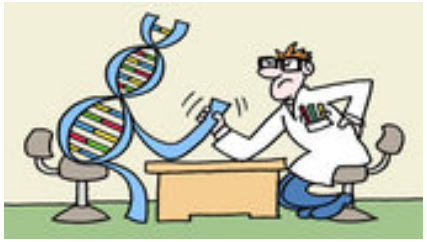
Población de Individuos ⇔ Conjunto de Soluciones
Evolución ⇔ Búsqueda
Mejor individuo ⇔ Solución Final

Computación Evolutiva: métodos metaheurísticos evolutivos (bio-inspirados)

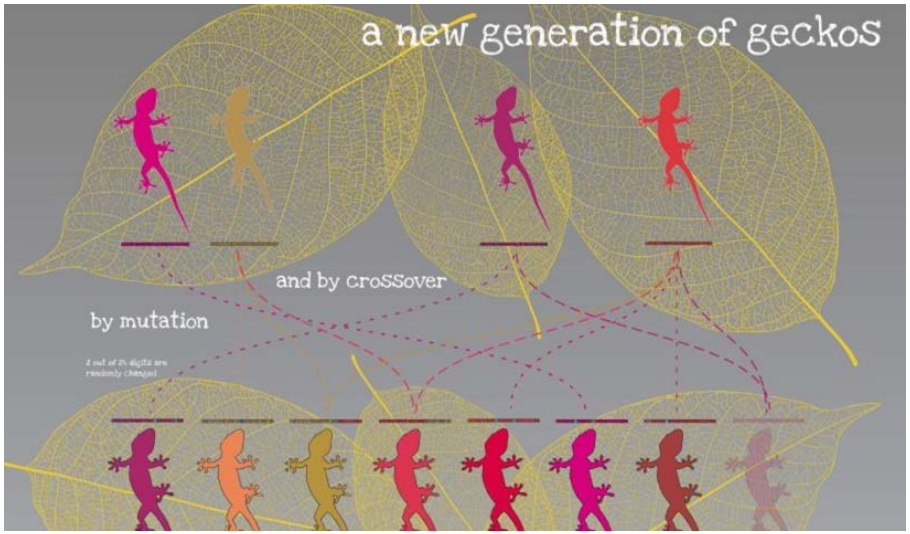
Población de Individuos ⇔ Conjunto de Soluciones
Evolución ⇔ Búsqueda
Mejor individuo ⇔ Solución Final

Aplicaciones de los AG

- Optimización combinatoria y en dominios reales
- Modelado e identificación de sistemas
- Planificación y control
- Vida artificial
- Aprendizaje y minería de datos
- Internet y Sistemas de Recuperación de Información, etc.



Winner of the Second Video Competition of the IEEE Computational Intelligence Society



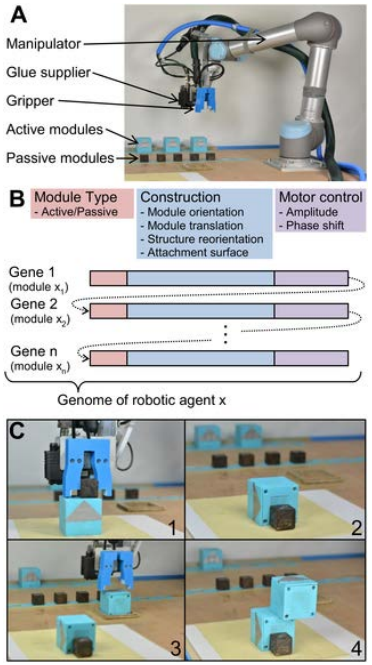
1st place

Team: **Evolutionary Computation - A Technology Inspired by Nature**

¿El límite?...

Se está investigando el proceso de evolución en los robots, construyendo un robot 'madre' que pueda diseñar, construir y evaluar sus propios 'hijos', y usar las conclusiones para mejorar la siguiente generación.

Todo ello sin intervención humana.

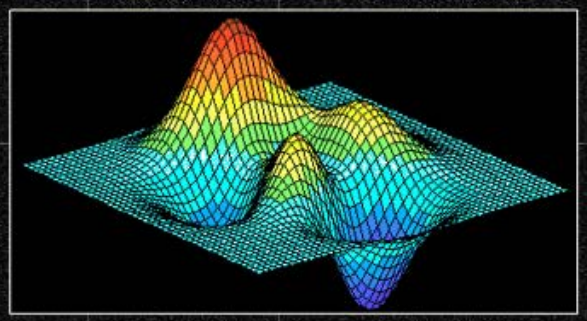


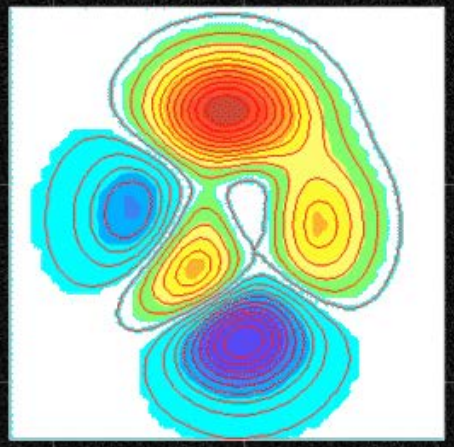
On the origin of (robot) species

Un ejemplo de Algoritmo Genético

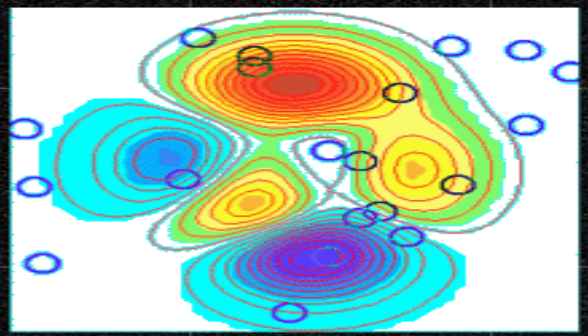
Ejemplo 1

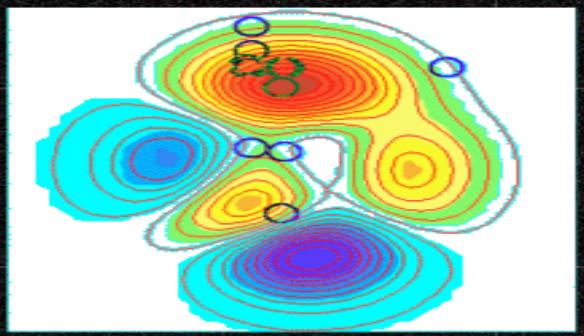
- Example: Find the max. of the "peaks" function
- z = f(x, y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) -1/3*exp(-(x+1)^2 - y^2).

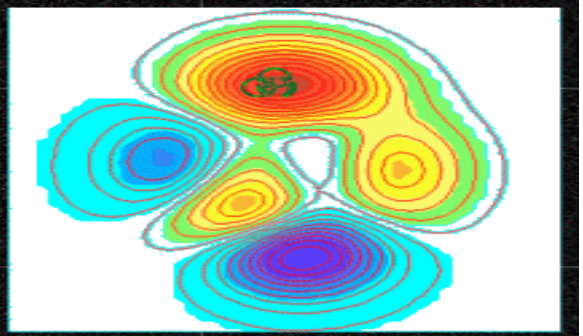




Source: Neuro-Fuzzy and Soft Computing, by J.-S. R. Jang, C.-T. Sun, E. Mizutani

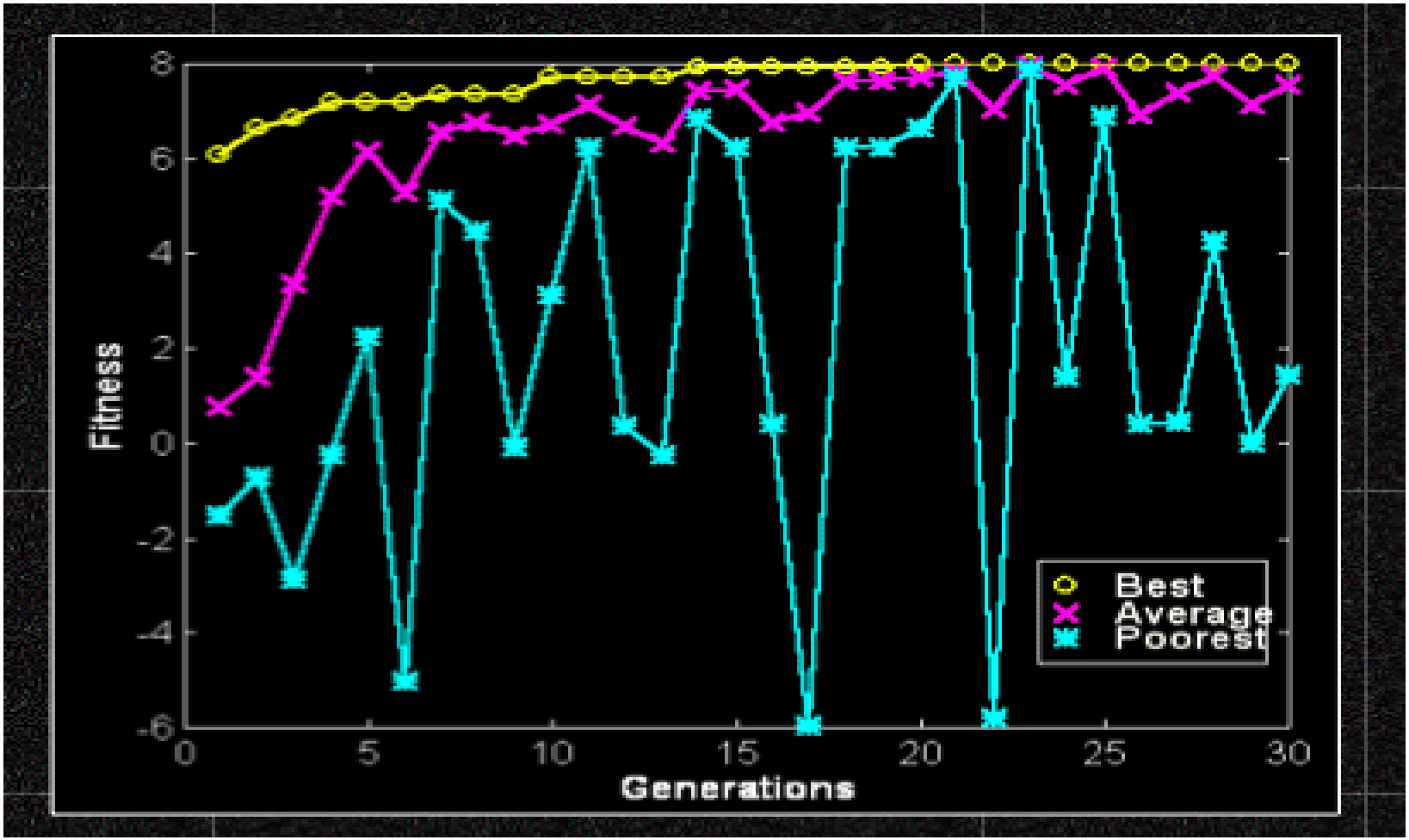




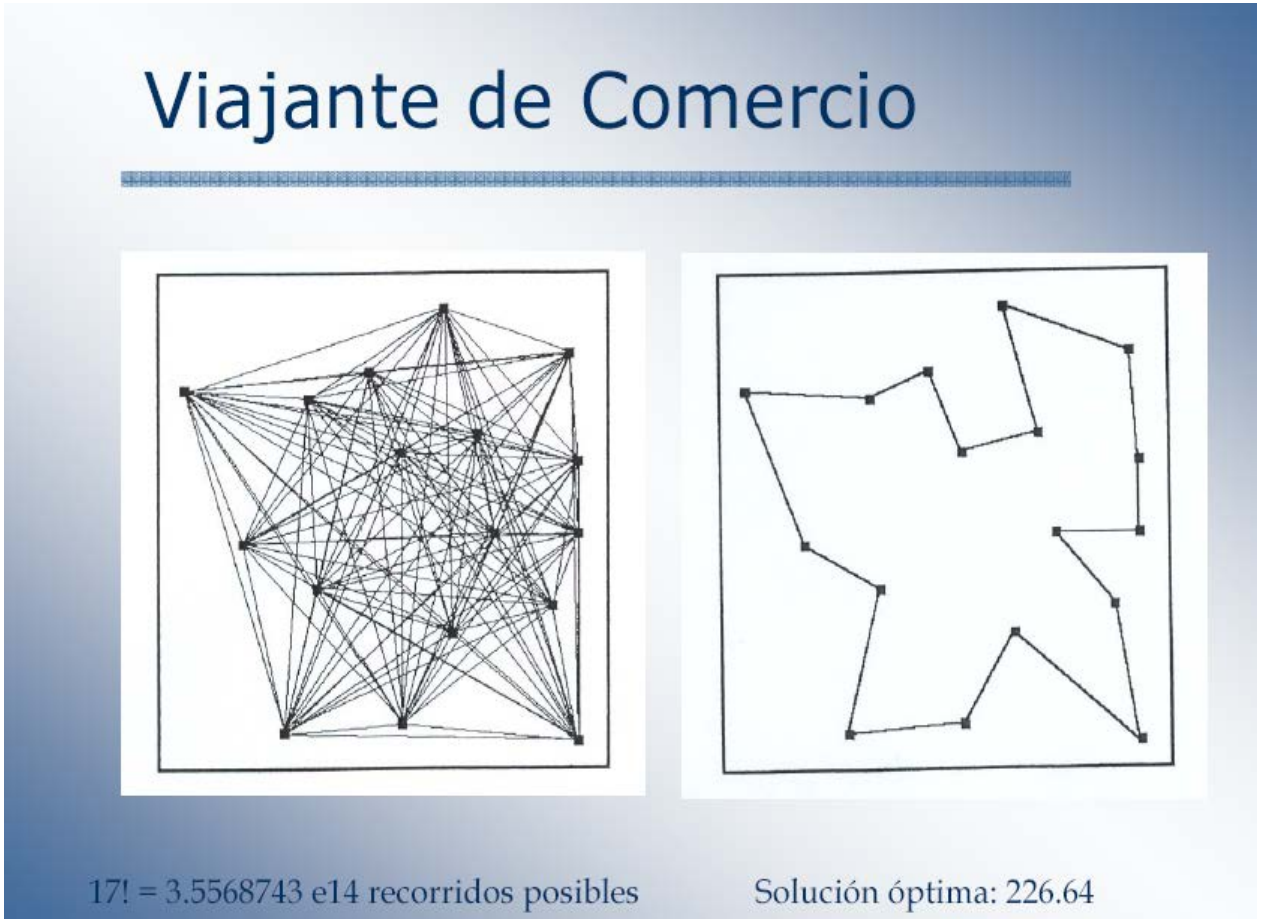


Initial population | 5th generation | 10th generation

La evolución de la adecuación (fitness)



Ejemplo 2



17 ciudades,

17! = 3.5568743 * 10^{14} Recorridos posibles

Con 1.000.000.000 recorridos/seg: 4 días de cómputo.

20 ciudades requerirían ……

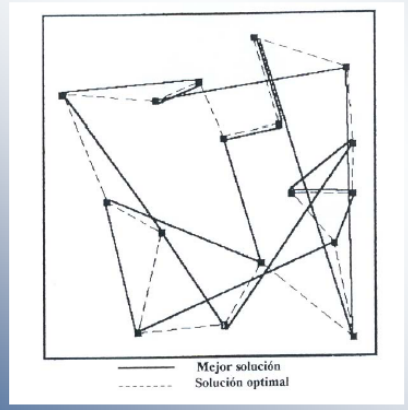
100 ciudades requerirían…..

20 ciudades requerirían 77 años,

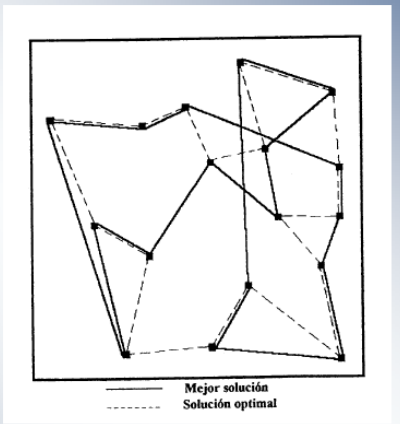
100 ciudades… 10^{135} millones de años

Iteración 0 → Iteración 25

Viajante de Comercio



Iteración: 0 Costo: 403.7

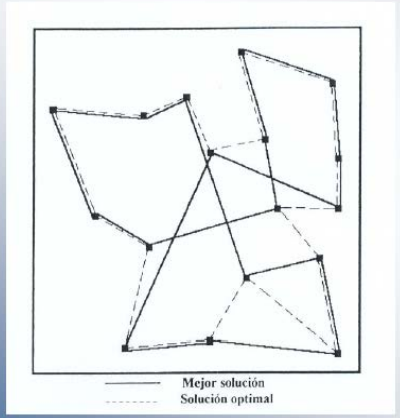


Iteración: 25 Costo: 303.86

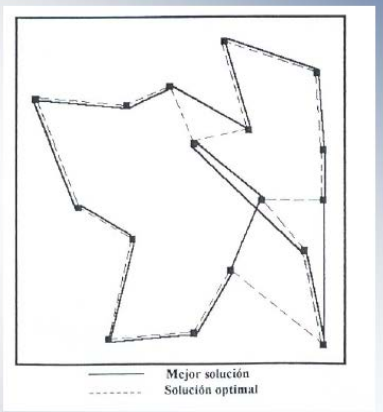
Solución óptima: 226.64

Iteración 50 → Iteración 100

Viajante de Comercio



Iteración: 50 Costo: 293,6

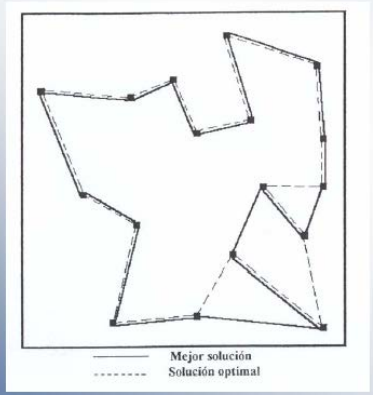


Iteración: 100 Costo: 256,55

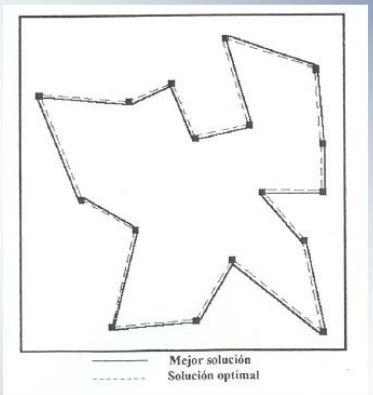
Solución óptima: 226,64

Iteración 200 → Iteración 400

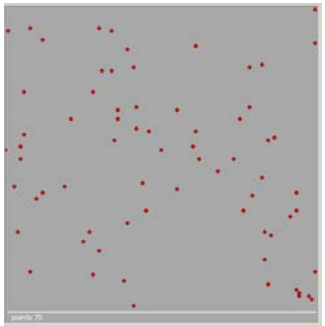
Viajante de Comercio

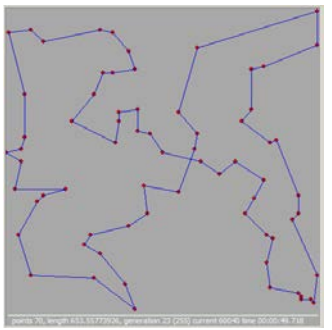


Iteración: 200 Costo: 231,4



Iteración: 250 Solución óptima: 226,64



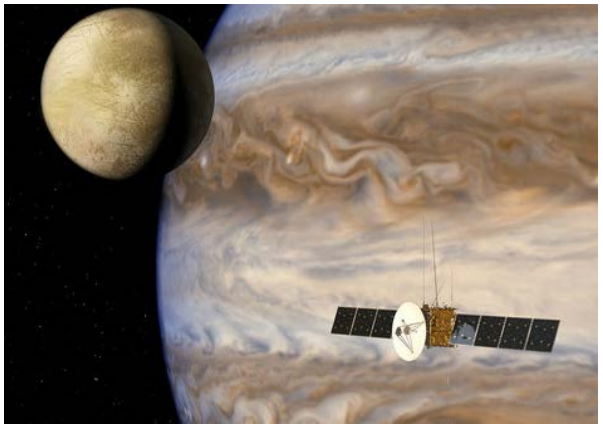


Aplicaciones: logística, recorrido señal en antenas, configuración, etc., etc.

Ejemplo 3: Planificación de rutas

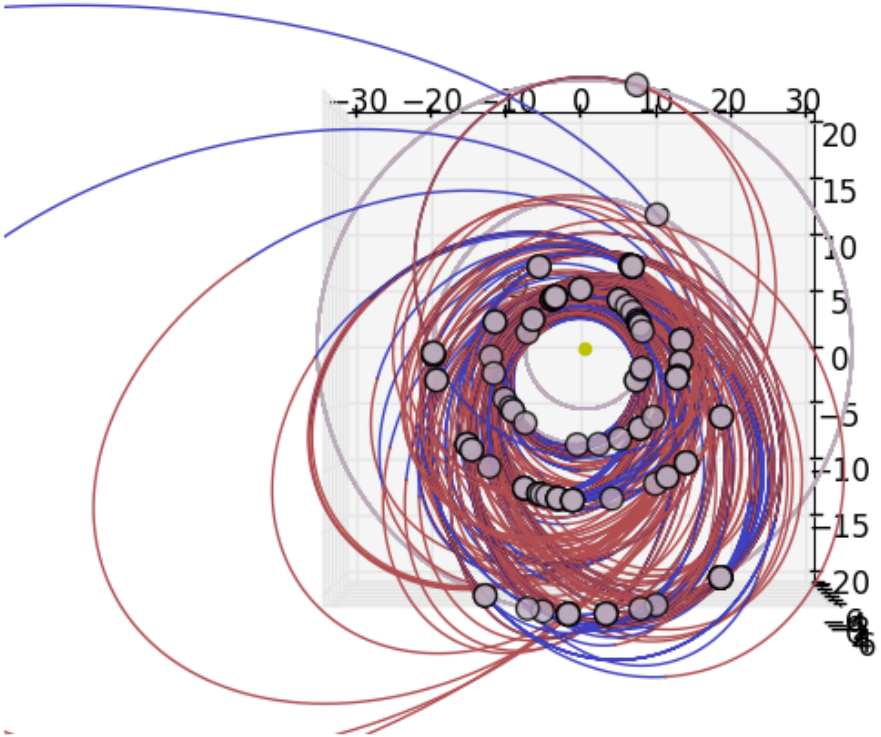
http://www.esa.int/Our_Activities/Technology/ESA_team_s_computer-evolved_trek_round_Jupiter_wins_science_prize

El grupo de conceptos avanzados de la ESA recibió el premio 'Humie' por la aplicación de la "computación evolutiva" en la planificación de la mejor ruta en torno a las principales lunas de Júpiter, logrando una eficacia más allá de los mejores esfuerzos humanos.



Descubiertas por Galileo, Júpiter tiene cuatro grandes lunas (Io, Europa, Calisto y Ganimedes), cada una con su propias características y girando en complejas e interrelacionadas órbitas en un mar gigante de gas.

Describiendo los satélites galileanos como balones de fútbol, con 32 caras de cada uno, el reto consistía en fotografiar tantas caras como sea posible, utilizando el menor combustible posible y minimizando la radiación desde Júpiter.



Conceptos principales de un Algoritmo Genético

1. Una **representación** adecuada (y simple) de las soluciones del problema (individuos). **Cromosomas, genes, genotipo y fenotipo**.

2. Una forma de crear una **población de soluciones iniciales** (individuos iniciales). A menudo, aleatoria.

3. Una **función de evaluación** capaz de medir la adecuación de cualquier solución (individuo) al problema. Hace el papel de 'heurística' para los procesos de selección y supervivencia. ***Fitness***.

4. Un conjunto de **operadores evolutivos** para combinar las soluciones existentes con el objetivo de obtener nuevas soluciones (nuevos individuos): ***selección, cruce, mutación y reemplazo*** de individuos. Guían el proceso de la búsqueda.

5. Conjunto de **parámetros**: tamaño de la población, número de iteraciones (generaciones), métodos de selección, cruce y mutación, probabilidades, etc.

Etapas de un Algoritmo Genético

0. Codificación de Soluciones: Fenotipo, Genotipo.

1. [Población Inicial]:

- Generar población aleatoria de ***n*** individuos: {x} *(soluciones del problema)*
- Evaluar la aptitud o *fitness* ***f(x)*** de cada individuo.

2. [Ciclo-Generacional]:

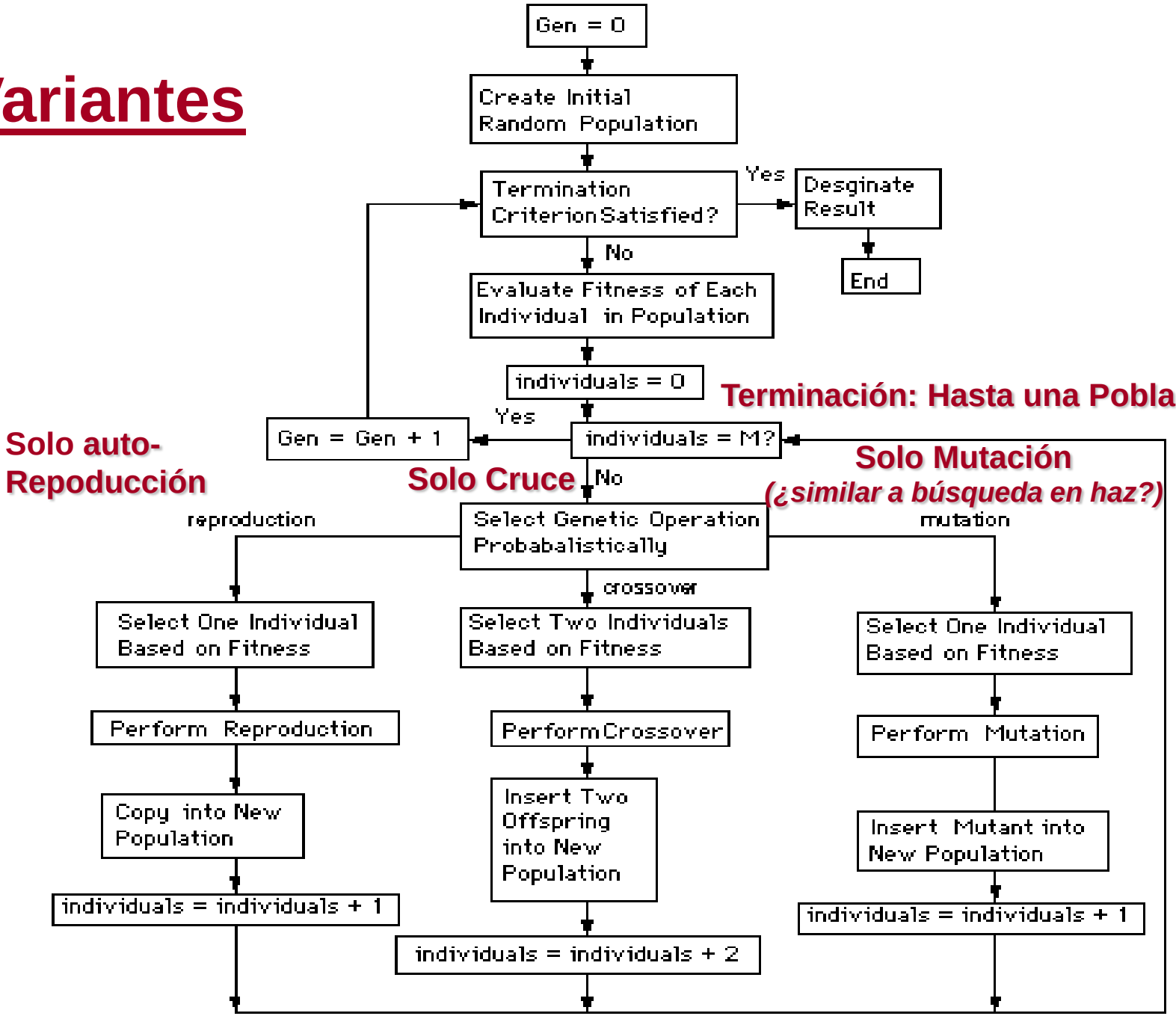
1. **[Selección]** Seleccionar dos padres de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
2. **[Cruce]** Combinar los genes de los dos padres para obtener descendientes.
3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo.
4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

3. Verificar condición de parada:

[Parada]: proporcionar como solución el **individuo con mejor valor de aptitud *f(x)*.**

[Ciclo]: Ir al paso **2**

Variantes


Gen = 0
Create Initial Random Population
Termination Criterion Satisfied?
Yes
Desginate Result
End
No
Evaluate Fitness of Each Individual in Population
individuals = 0
Terminación: Hasta una Población M
Yes
Gen = Gen + 1
individuals = M?
Solo auto-Repoducción
Solo Cruce
No
Solo Mutación
(¿similar a búsqueda en haz?)
reproduction
Select Genetic Operation Probabalistically
mutation
crossover
Select One Individual Based on Fitness
Select Two Individuals Based on Fitness
Select One Individual Based on Fitness
Perform Reproduction
PerformCrossover
Perform Mutation
Copy into New Population
Insert Two Offspring into New Population
Insert Mutant into New Population
individuals = individuals + 1
individuals = individuals + 2
individuals = individuals + 1

Otra variante: Algoritmos Genéticos Paralelos

Los algoritmos genéticos pueden implementarse en paralelo de manera natural

Hay diversos enfoques.

1) Enfoques de **grano grueso**: dividen la población en ***grupos de individuos*** llamados **demes**. Es la más típica.
   - Cada deme es asignado a un nodo de computación y se aplica un AG específico.
   - El cruce entre los demes ocurre con menor frecuencia que en el interior de ellos. La transferencia entre los demes se da por un proceso de *migración*.
   - Una de las ventajas de este enfoque es que reduce el fenómeno de *crowding* que es común en versiones no paralelas de los algoritmos genéticos:

     Crowding: *un individuo muy apto comienza a reproducirse rápidamente de tal forma que copias de este individuo, o bien de individuos muy parecidos, ocupan una fracción importante de la población. Este fenómeno reduce la diversidad de la población, haciendo la búsqueda mas lenta.*

2) La paralelización de **grano fino** asigna ***un procesador a cada miembro*** de la población. Las combinaciones se dan solo entre vecinos muy cercanos. Existen diferentes tipos de vecindad.

Diseño de un Algoritmo Genético

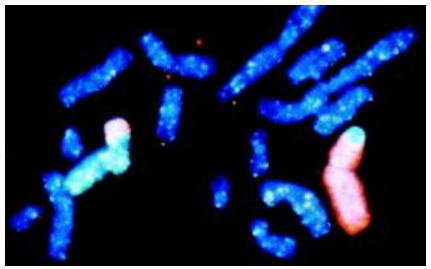
1. Diseñar una representación: correspondencia entre Genotipo y Fenotipo
2. Decidir cómo inicializar una población
3. Diseñar una forma de evaluar un individuo (fitness)
4. Decidir cómo seleccionar los individuos para ser padres
5. Diseñar un operador de cruce adecuado
6. Diseñar un operador de mutación adecuado
7. Decidir cómo reemplazar a los individuos
8. Decidir la condición de parada

Representación del individuo: una solución del problema

Evolución Genética

Individuo ≡ Cromosoma: cadena de ADN que se encuentra en el núcleo de las células.

Los cromosomas son responsables de la transmisión de información genética.



Gen: sección de ADN que codifica una cierta función bioquímica definida. Supone la unidad de herencia (**A**denina, **C**itosina, **G**uanina, **T**imina).

Genotipo: Información genética de un organismo.

Fenotipo: cualquier característica observable (externa) de un organismo (dependiente del genotipo más la influencia del medio/entorno).

Algoritmo Genético

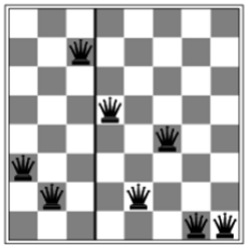
Individuo ≡ Cromosoma: estructura de datos que contiene una **secuencia de parámetros** que permite definir (o formar) una **solución** al problema.

Un cromosoma representa el **Genotipo** (representación interna de la solución):

Genotipo:

3	2	7	5	2	4	1	1

Fenotipo: representación externa de la solución



Gen: elementos del cromosoma que codifica el valor de un (o varios) parámetro, propiedad o aspecto del individuo.

Alelo: Valor (Valores) de un gen

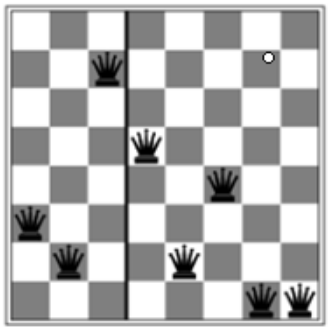
¡Es muy importante el diseño de los individuos (cada solución)!

Codificación de Soluciones en un AG: Solución ⇔ Cromosoma

- La codificación de los individuos (o solución) como un cromosoma (genotipo) es fundamental para la aplicación de los AG y es lo primero que hay que diseñar.
- La codificación debe poder representar todos los fenotipos (soluciones) posibles.

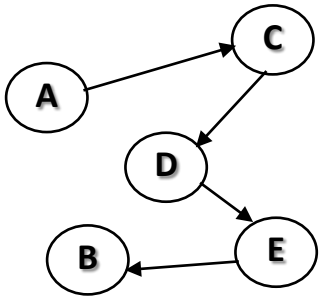
Correspondencia entre Genotipo y Fenotipo

Algunas veces, la obtención del Fenotipo a partir del Genotipo es un proceso obvio.



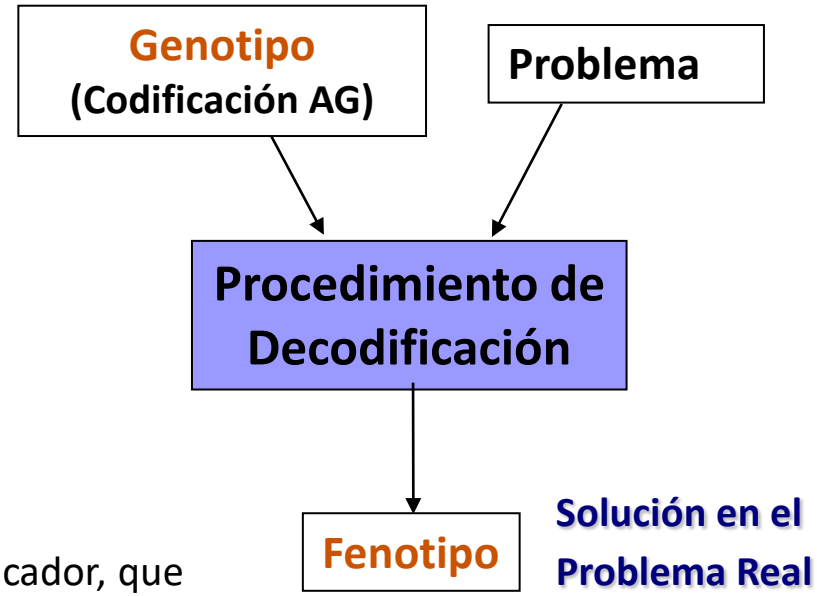
(6 7 2 4 7 5 8 8)

Fenotipo



Genotipo

(A C D E B)



En otras ocasiones, el Genotipo puede:

a) Ser un conjunto de parámetros para el proceso decodificador, que sobre los datos de un problema, obtendrá el fenotipo correspondiente.

b) Tener una estructura no linearizada (árbol, <atributo-valor>, etc.)

Estructura general de un Algoritmo Genético Simple

1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

Población Inicial

a) Se puede inicializar uniformemente de forma **aleatoria** en el espacio de búsqueda:

- Representación binaria: 0 o 1 con probabilidad 0.5
- Representación real: uniforme sobre un intervalo dado (para valores acotados)
- Representación de orden: Soluciones (individuos) factibles aleatorias (~ viajante)

b) Elegir la población inicial a partir de los resultados de una **heurística** previa

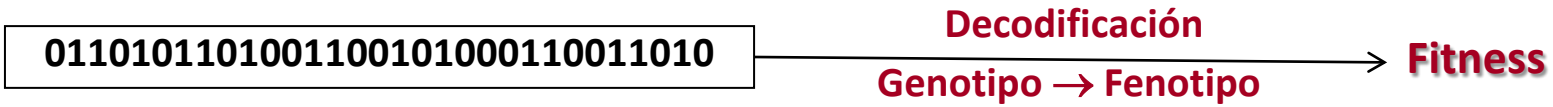
- Posible pérdida de la diversidad genética
- Posible preferencia no recuperable

Estructura general de un Algoritmo Genético Simple

1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

Aptitud

Evaluación de un individuo: **Función de Fitness o Aptitud**



- **Propósito**: diseñar una medida de la **calidad** de la solución candidata o cromosoma
- **Normalmente, este es el paso más costoso en las aplicaciones reales**
- Puede ser una subrutina, una caja negra, o cualquier proceso externo
- Se pueden utilizar funciones aproximadas para reducir el coste de evaluación

Manejo de restricciones:

¿Qué hacer si un individuo incumple alguna restricción del problema?:

Rechazo, Penalización, Reparación, Preservación.

- Si es un problema de factibilidad: típicamente se rechaza o se penaliza el fitness de la solución *(ejemplo: 8-reinas)*
- Si no es un problema de factibilidad: típicamente se repara la solución o se mantiene (preserva) la factibilidad *(ejemplo: viajante de comercio)*

Estructura general de un Algoritmo Genético Simple

1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

Estrategia de Selección

Selección de los elementos que se reproducen en cada ciclo generacional

- Debemos garantizar que los mejores individuos tengan una mayor posibilidad de ser padres (reproducirse) frente a los individuos menos buenos.

  Mientras más apto sea un organismo (valor de aptitud más alto), más veces será seleccionado para reproducirse.

- Debemos ser cuidadosos para dar una posibilidad de reproducirse a los individuos menos buenos. Éstos pueden incluir material genético útil en el proceso de reproducción.

- La **PRESIÓN SELECTIVA** afecta el proceso de búsqueda

  - Selección muy fuerte: organismos de *alto fitness* que son subóptimos pueden dominar la población, disminuyendo la diversidad necesaria para progresar.

  - Selección muy débil: evolución muy lenta.

- El número de padres que se seleccionan de la población (para cruzarse) es un parámetro de diseño *(relacionado con el reemplazo de la población)*.

Métodos de Selección

- **Selección elitista:** elección directa de los miembros más aptos de cada generación.

- **Selección proporcional:** los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza. Los peores individuos también pueden ser elegidos.
  - **Selección por rueda de ruleta.** Basado en el fitness del individuo y la media de grupo.

- **Selección jerárquica:** los individuos atraviesan múltiples rondas de selección en cada generación. Las primeras evaluaciones son más rápidas y menos discriminatorias (hay muchos individuos), las posteriores evaluaciones son más rigurosas.
  - **Selección por torneo:** grupos de individuos compiten entre ellos, sucesivamente.

- **Selección por rango: L**a selección se basa en el rango (posición) de cada individuo (en vez de su fitness). A cada individuo se le asigna un ranking basado en su aptitud, y la selección se basa en este ranking, en lugar de las diferencias absolutas en aptitud.

- **Selección generacional:** La descendencia de una generación es la siguiente generación de padres. No se conservan individuos entre generaciones.

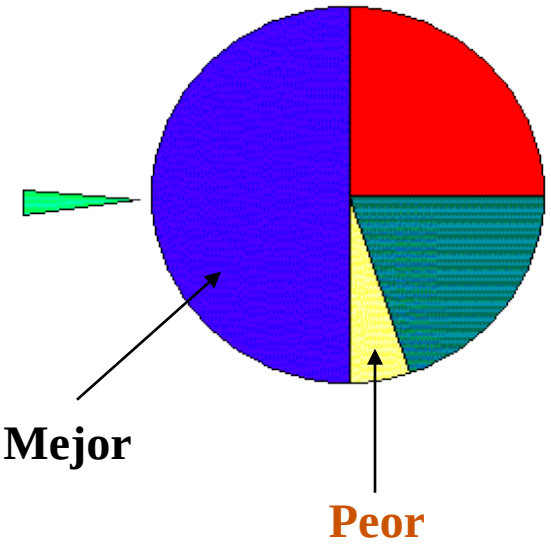
Selección Proporcional

A cada individuo$_i$ con fitness f_i , se le asigna una probabilidad de ser elegido basada en su f_i.

Selección por Rueda de Ruleta

A cada individuo de la población se le asigna una parte proporcional a su fitness *fi* en una ruleta, tal que la suma de todos los porcentajes sea la unidad.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$



Los mejores individuos (más adaptados) tienen:

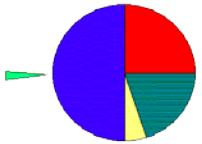
- Más posibilidades de ser elegidos
- Más espacio en la ruleta

Ejemplo Selección por Rueda de Ruleta

SUPERINDIVIDUO

Es un cromosoma que tiene una *adaptación* muy alta en comparación con el resto de los cromosomas de la población.

Individuo N°	Adaptación f_i	Prob. De Selección $p_i = f_i / \Sigma f_i$
1	200	0.797 (200/251)
2	30	0.119 (30/251)
3	10	0.040
4	8	0.032
5	2	0.008
6	1	0.004



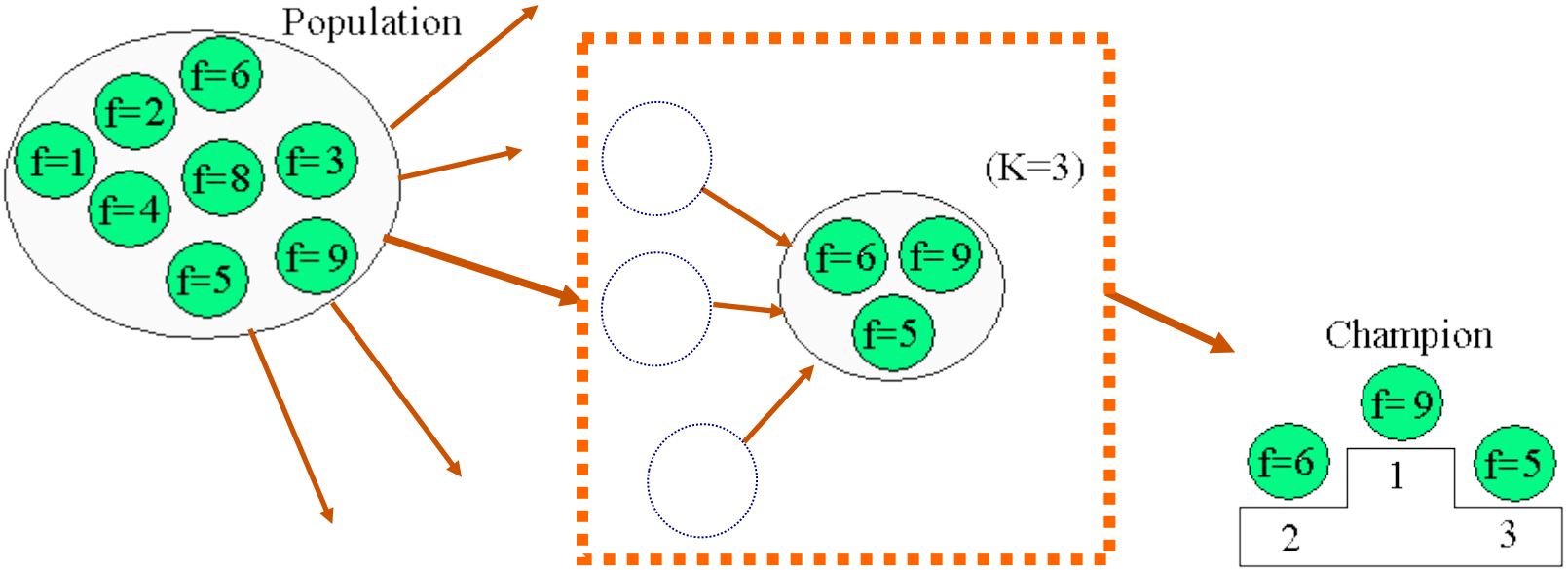
Es muy selectiva

Inconvenientes:

- Peligro de Convergencia Prematura (*premature convergence*) a un sub-óptimo debido a que ***individuos muy aptos dominan la población completa muy rápidamente***
- Baja Presión Selectiva (*selection pressure*) cuando los valores de aptitud son muy cercanos entre sí. Se produce ***estancamiento en el proceso de búsqueda***

Selección por Torneo:

- Se establecen grupos de *k* individuos (*k* es el parámetro del tamaño del torneo)
- Se selecciona el mejor de cada grupo: método elitista o método proporcional
- Se realizan sucesivos torneos, hasta que se seleccionan el número deseado de padres.



Variando el número (K) de individuos que participan en cada torneo se modifica la presión de selección.

- Cuando participan muchos individuos en cada torneo (K↑), la presión de selección es elevada y los peores individuos apenas tienen oportunidades de reproducción.
- Cuando el tamaño del torneo es reducido (K↓), la presión de selección disminuye y los peores individuos tienen más oportunidades de ser seleccionados.

Selección por Rango:

- La población se ordena de acuerdo a sus valores de la función fitness. La posición de cada individuo en la lista ordenada representa su **rango: r(x).**
- La **probabilidad** asignada a cada individuo depende de su **posición en el rango** y no de su valor objetivo
  - Solventa los problemas de la selección proporcional respecto a: **estancamiento** (cuando la presión selectiva es muy baja) o **convergencia prematura** (presión alta, súper-individuo)
  - El rango r(x) se usa para ponderar los individuos, aplicando una función h(x).

$$h(x) = \max - (\max - \min) * \frac{(r(x)-1)}{n-1}$$

$$P(x) = \frac{h(x)}{n}$$

Donde:

max+min=2

max ≥ min : max∈[1,2], min∈[0,1],

(max recomendado = 1.1)

↑max ⇒ Mayor separación P(x)

max=1, min=1: Igual P(x) para todos

Ejemplo (max = 1.1 → min = 0.9):

Fitness(x)	*Ranking(x)*	*h(x)*	*P(x)*
5	2	1	**0.33**
2	3	0.9	**0.30**
19	1	1.1	**0.37**
Σ		3	1

Aplicar Ruleta u otra técnica Proporcional

P(x) max=1	*P(x) max=2*	*Proporcional*
0.33	0.33	0.19
0.33	0.00	0.08
0.33	0.67	0.73

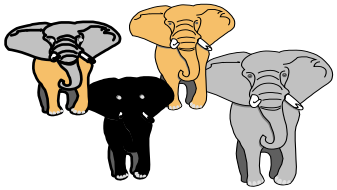
Estructura general de un Algoritmo Genético Simple

1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

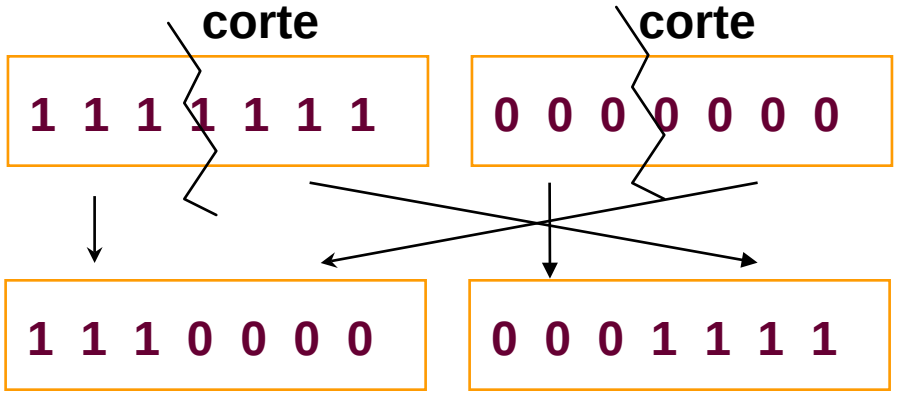
Cruce

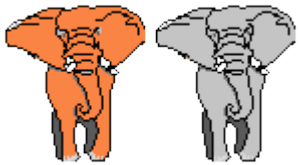
Naturaleza: proceso complejo que ocurre entre cromosomas homólogos. Los cromosomas se alinean, luego se fraccionan en ciertas partes y posteriormente intercambian fragmentos entre sí.

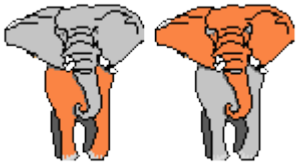
Computación Evolutiva: se simula la recombinación intercambiando segmentos de cadenas lineales de longitud fija (los cromosomas).

Población:  . . .

Cada cromosoma es cortado en ***n*** **puntos** y las porciones de cada cromosoma se recombinan (Ejemplo para ***n=1***)



padres 

hijos 

Cruce de 2 puntos:

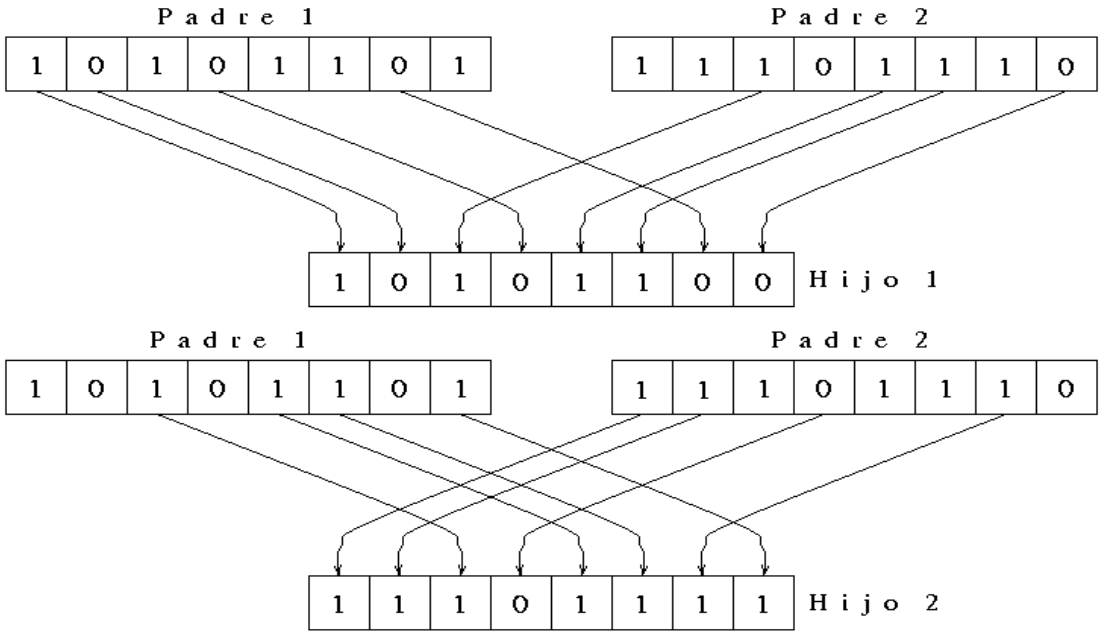
- Se seleccionan 2 puntos, y se intercambian los grupos de bits entre ellos

Padre1 = 10|1010|1010
Padre2 = 11|1000|1110

Hijo1 = 11|1010|1110
Hijo2 = 10|1000|1010

Cruce Uniforme:

- Se intercambian bits. Para cada bit del hijo se selecciona aleatoriamente de qué padre viene



Puntos de Cruce

- El cruce de un punto (n=1) presenta desventajas (por ejemplo, poca flexibilidad)
- No existe consenso en torno al uso de valores para $n \geq 3$.
- Asimismo, el incrementar el valor de n se asocia con un mayor efecto destructivo del cruce.
- En general, es aceptado que el **cruce de 2 puntos** es mejor que el cruce de un punto.

Se han realizado diversos análisis para comparar el desempeño de distintos operadores de recombinación, pero no hay evidencia concluyente de cuál es mejor.

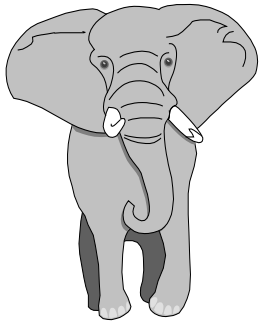
- La elección más acertada depende del problema a resolver.
- Sin embargo, parece haber consenso en cuanto a que la **recombinación de 2 puntos** y la **recombinación uniforme** son generalmente preferibles a la recombinación de 1 punto
- *Se pueden plantear otros tipos de cruce dependientes del problema, estructura no lineal del genotipo, etc.*

Estructura general de un Algoritmo Genético Simple

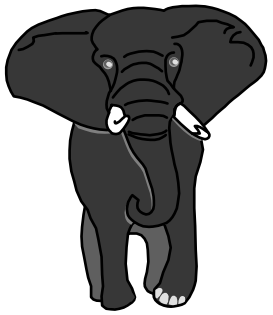
1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

Mutación

Antes **1 1 1 1 1 1 1**



Después **1 1 1 0 1 1 1**



Gen Mutado

La Mutación se aplica con probabilidad $\mathbf{p_{mut}}$ **para cada gen** de cada cromosoma.

La probabilidad de mutar (P_{mut}) es normalmente baja, generalmente en el rango [0.001..0.05]

Mutación por Inserción:

Se selecciona un valor en forma aleatoria y se le inserta en una posición arbitraria.

Ejemplo:

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

^

Si elegimos la posición 7 y movemos ese valor a la posición 2, tendríamos:

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

Mutación por Intercambio Recíproco:

Se seleccionan dos puntos al azar y se intercambian estos valores de posición.

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

^ <----------> ^

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

Ejemplo de Algoritmo Genético (Problema del Viajero de comercio)

Encontrar una ruta a través de una serie de ciudades que visitando todas las ciudades se minimice la distancia recorrida.

Lista de las ciudades a visitar

1) London 3) Dunedin 5) Beijing 7) Tokyo
2) Venice 4) Singapore 6) Phoenix 8) Victoria

Cromosomas *(orden en el que se recorren las ciudades)*

CityList1 (3 5 7 2 1 6 4 8)
CityList2 (2 5 7 6 8 1 3 4)

Cruce (2-puntos):

*Cromosoma-i (3 5 **7 2 1 6** 4 8)*
*Cromosoma-j (**5 8** 7 6 8 1 **3 4**)*
*Hijo k (**5 8 7 2 1 6 3 4**)*

***Mutación (intercambio)*:**

Antes: (5 8 7 2 1 6 3 4)
Después: (5 8 6 2 1 7 3 4)

Los individuos válidos deben visitar todas las ciudades:

- La operación de cruce debe obtener individuos válidos,
- Los individuos no válidos son eliminados, reparados o se les asocia un valor de fitness = -∞

Estructura general de un Algoritmo Genético Simple

1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

Reemplazo de la Población *(cambio generacional)*

Establece el criterio de los supervivientes en cada iteración / sustitución generacional.

- **Generacional**
  - **Toda la población es reemplazada en cada generación:** cada individuo sobrevive solo una generación y todos los hijos reemplazan a todos los padres.

- **Estado – Estacionario (Steady–State AG).** ***Típico.***
  - Solo unos **pocos individuos** son reemplazados en cada generación: típicamente se generan uno/dos hijos y solo uno/dos individuos (los peores) son reemplazados.

- **Gap Generacional**
  - **Criterio entre los dos esquemas anteriores: se define un porcentaje (gap) de los individuos que serán reemplazados cada generación, por evaluación del fitness:**

    Proporción = 1 (para generacional); Proporción = 1/|población| (para estacionario)

- **Reemplazo catastrofista:** cada cierto tiempo se elimina una gran cantidad de individuos:
  - **Empaquetado:** solo se mantiene un individuo de los que tienen igual fitness.
  - **Juicio final:** se destruye toda la población excepto el más avanzado. El resto de población se genera aleatoriamente.

Estructura general de un Algoritmo Genético Simple

1. **[Población Inicial]:** Generar población aleatoria de ***n*** cromosomas {x} *(soluciones apropiadas al problema)*
2. **[Aptitud]:** Evaluar la aptitud o *fitness* ***f(x)*** de cada cromosoma ***x*** en la población.
3. **[Nueva población]** Crear una nueva población repitiendo los pasos siguientes hasta completar la nueva población:
    1. **[Selección]** Seleccionar dos padres cromosomas de la población de acuerdo a su aptitud: Probabilidad de cruce ($\mathbf{p_c}$).
    2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
    3. **[Mutación]** Con una probabilidad de mutación ($\mathbf{p_{mut}}$), mutar cada gen de los cromosomas hijo
    4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.
4. **[Verificar]** Condición de parada → **STOP**, y proporcionar como solución el **cromosoma con mejor valor de aptitud *f(x)*.**
5. **[Ciclo]** Ir al paso **2**

Condición de Parada

- **Número** fijo de **generaciones** o **soluciones** generadas
- Tras un **tiempo de cómputo** fijo
- Considerando medidas de **diversidad en la población** ***(convergencia)***
- Cuando se alcance el **óptimo** (si es conocido), o un determinado valor de fitness.
- Tras **varias generaciones sin ninguna mejora** ***(convergencia)***

Entornos y/o Herramientas para desarrollar AG

En general, las herramientas genéricas no suelen ser muy útiles para desarrollar AG concretos:

- Codificación/decodificación/fitness: suelen ser muy dependientes del problema (suponer un individuo como una secuencia de bits sin estructura no suele bastar).
- Las operaciones (cruce, mutación, etc.) son dependientes de la codificación de los individuos, etc.

Existen sistemas/demos para diseñar AG, o evaluar parámetros en problemas tipo *(optimización de funciones matemáticas, TSP, etc.)*:

- **MATLAB** (ToolBox disponibles)**. *Global Optimization Toolbox*.**
- **Opt4J.** Entorno libre en Java. Disponible en: http://opt4j.sourceforge.net/
- **HeuristicLab** (original método de programación). Libre: http://dev.heuristiclab.com/
- **Mathematica**, **GoldenBerry-GA**, etc.

Así como librerías que implementan el código general de los AG:

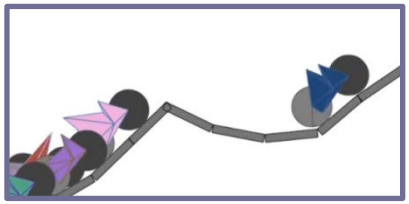
- **http://www.geneticprogramming.com/GPpages/software.html**
- **http://www.rubicite.com/Tutorials/GeneticAlgorithms.aspx ,**
- **http://people.sc.fsu.edu/~jburkardt/cpp_src/simple_ga/simple_ga.html**
- En **C++** (Open BEAGLE, MALLBA, Evolving Objects-EO, GALIB), ***Java*** (JGAP, JCLEC,ECJ, OSGiLiath), **Python** (DEAP), etc.

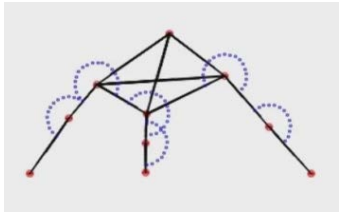
Estas utilidades pueden ayudar (no hay que implementar el algoritmo), pero la verdadera complejidad es el diseño de los individuos, codificación, decodificación y fitness y semántica de las operaciones (cruce y mutación) en cada problema concreto

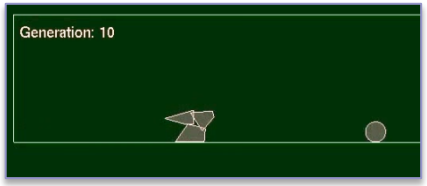
En resumen…

Los algoritmos genéticos constituyen una de las metaheurísticas más aplicadas.

- Existen muy diversas aplicaciones reales
- Existe una amplia base de trabajo en el área

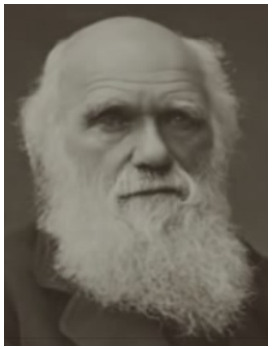


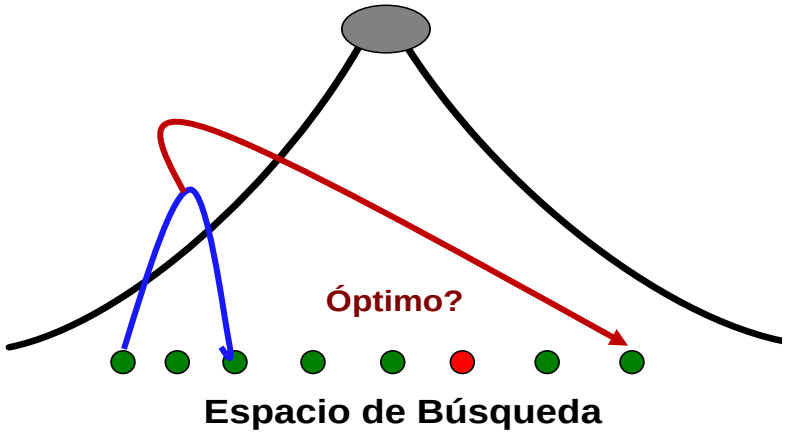


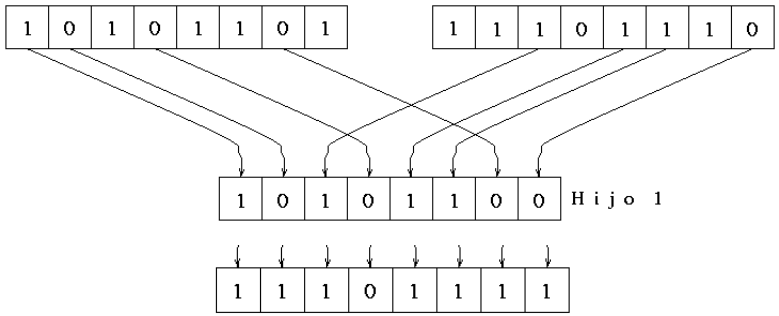


La metaheurística, fundamentalmente, se basa en:

1) Explotación de las características de buenas soluciones (selección/cruce) a nuevas soluciones ≈ Proceso mejora local
2) Cierta exploración de nuevas soluciones variantes (mutación)



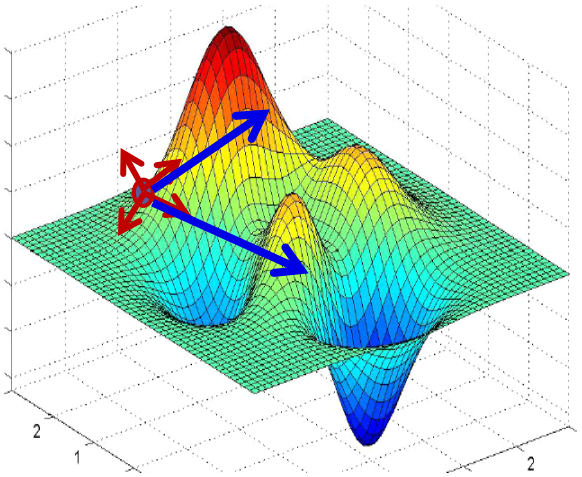




Exploración *vs.* Explotación en Algoritmos Evolutivos

La aplicabilidad de los AG proviene en gran parte en su capacidad de combinar un proceso de búsqueda global (exploración) con un proceso de búsqueda local (explotación).

- La exploración está relacionada con mayor diversidad de la población, a fin de explorar un amplio espacio de estados no priorizando un alto fitness.
- La explotación está relacionada con la focalización en individuos de alto fitness, priorizando su importancia.



La exploración / explotación en AG es controlada por los parámetros de las operaciones:

- Mutación: Introduce innovación en la búsqueda.
  - Salto a una zona de búsqueda distante/local.
  - Mayor mutación implica mayor exploración (búsqueda global). Menor mutación implica mayor explotación.
- Cruce: Recombinación de soluciones existentes
  - Relacionado con la exploración: mayor nº puntos de cruces implica mayor exploración. Menor nº cruces implica mayor explotación.
- Selección: dirige la búsqueda hacia zonas prometedoras del espacio de búsqueda.
  - Relacionado con la explotación (presión selectiva). A mayor presión selectiva mayor explotación (menor presión, mayor exploración).

Conclusiones

Los Algoritmos Genéticos están inspirados en la evolución *(darwiniana: supervivencia del mejor)* **que sucede en la naturaleza, con el fin de ir mejorando soluciones.**

Hay muchas guías o recomendaciones para diseñar un AG:

- **Es muy importante la codificación genética:** debe capturar la información relevante de la solución y hacerla fácilmente identificable.
- **Tamaño de la población**: debe de ser suficiente para garantizar la diversidad de las soluciones.
- **Condición de terminación:** lo más habitual es que la condición de terminación sea la convergencia del algoritmo genético o un número prefijado de generaciones.
- **Lenguaje de Representación:** lo ideal es utilizar alfabetos con ***baja cardinalidad*** (es decir, con pocas letras) como el binario.
- **Función de Aptitud:** debe hacer una presión adecuada de mejora y significar realmente una solución mejor para el problema dado.
- ***No fiarse de la autoridad central (operadores que 'controlan' a toda la población)*:**

  La Naturaleza actúa de forma distribuida y segmentada. Por tanto, se debe minimizar la necesidad de operadores que "vean" a toda la población. Por ejemplo, en vez de comparar el fitness de un individuo con todos los demás, se puede comparar solo con los *vecinos*, es decir, aquellos que estén, de alguna forma, situados cerca de él

  → más parecido a la naturaleza, donde los recursos son compartidos por los individuos locales.

- **Detractores (falta de criterio o razón subyacente para la optimización) y Defensores (proceso evolutivo, aplicaciones con éxito).**
  - ¿Es suficiente la información genética para capturar y progresar en los aspectos relevantes de la solución?
  - ¿Es suficiente una evolución basada solo en los genes?
  - O más bien, ¿los individuos evolucionan por aprendizaje, comunicación y competición entre ellos, por 'entes' de información no exclusivamente genéticos?

- **Muchas aplicaciones:**

  Acústica, Ingeniería aeroespacial, Astronomía y astrofísica, Química, Ingeniería eléctrica, Mercados financieros, Juegos, Geofísica, Ingeniería de materiales, Militares, Biología molecular, Reconocimiento de patrones y minería de datos, Robótica, Diseño de rutas y horarios, etc.

- **Diversa información disponible en la Web:**

  www.genetic-programming.org www.geneticprogramming.com

Somewhere, something went wrong
(Anonymous)

Ejercicio

Resolver el siguiente problema de “cifras y letras”:

Solución:

((10*100) +8) / (9-5) = 252

Codificación: Genes del individuo

?

Diseñar un algoritmo genético para encontrar la secuencia de operaciones matemáticas (+,-,*,/) para, a partir de los números dados, aproximarse tanto como sea posible a 252