ENXEÑARÍA DO COÑECEMENTO

4º Grao Enxeñaría Informática

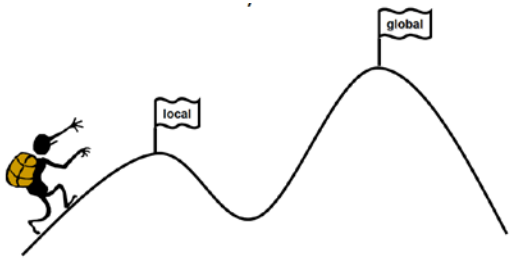
Curso 2016-17
5. Computación evolutiva

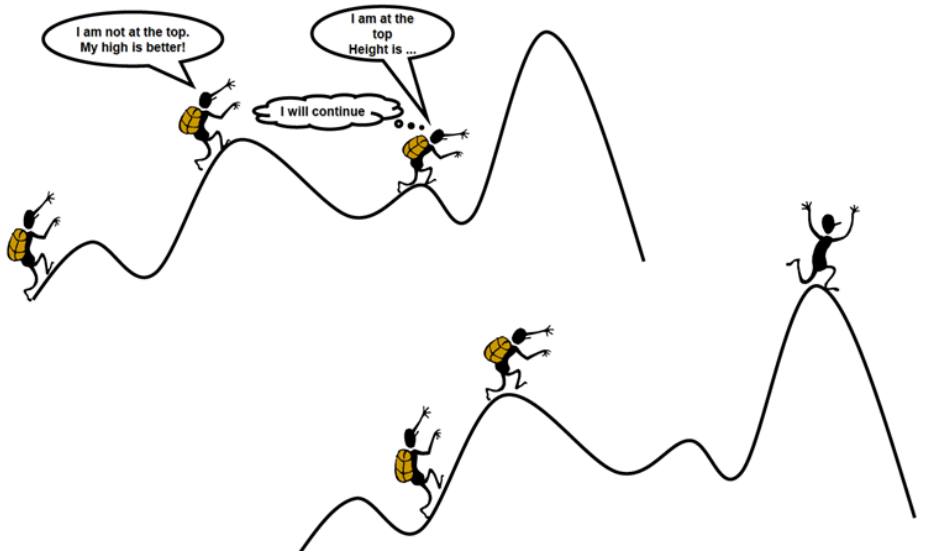
Alberto J. Bugarín Diz
Departamento de Electrónica e Computación
Universidade de Santiago de Compostela
alberto.bugarin.diz@usc.es

5. Contexto

•4. Metaheurísticas: introdución e clasificación. Metaheurísticas basadas en traxectorias: simulated annealing, tabu search.

•5. Computación evolutiva. Algoritmos xenéticos: representación, operadores, selección e remprazo. Aplicacións en problemas de busca.





- **Clasificación: MH basadas en…**
  - **métodos constructivos: GRASP, ACO**
  - **De búsqueda o trayectorias: recorren el espacio de soluciones mediante una transformación iterativa de una solución de partida (búsqueda local):** *Enfriamiento Simulado, Búsqueda Tabú, …*
  - **poblaciones:** evolución de conjuntos de soluciones o poblaciones, habitualmente con procedimientos aleatorios. Ej.: Computación evolutiva, PSO, …

5. Computación Evolutiva (CE)

- Objetivos

  - Describir que es un algoritmo evolutivo, explicando sus componentes y los conceptos básicos

  - Relacionar con otras técnicas de optimización global

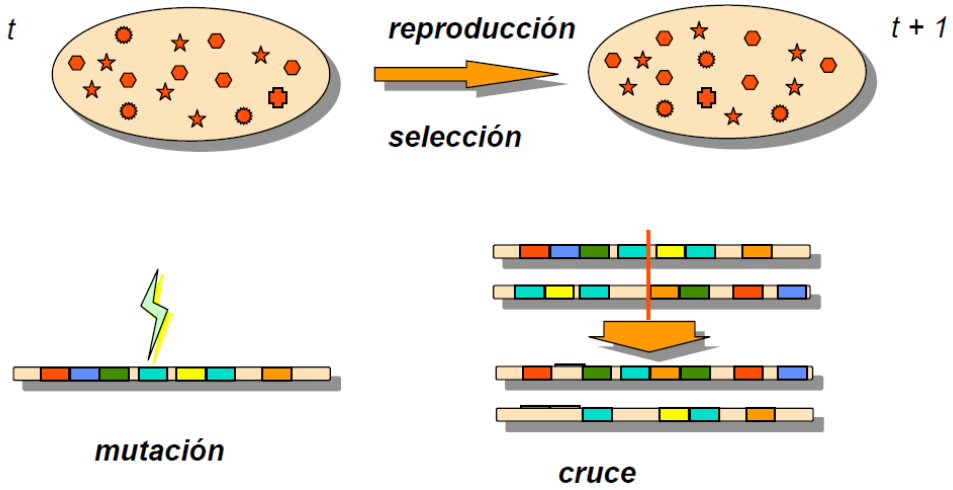
- ¿Que es un algoritmo evolutivo (AE)? Metáfora biológica

  – Diferentes variantes, una idea base común: dada una población de individuos con recursos limitados, la competencia por los recursos produce la **selección natural**. Supervivencia de los que mejor se adaptan al medio.

  – Evolución: mejora en la adaptación de la población al medio

5. Computación Evolutiva (CE)

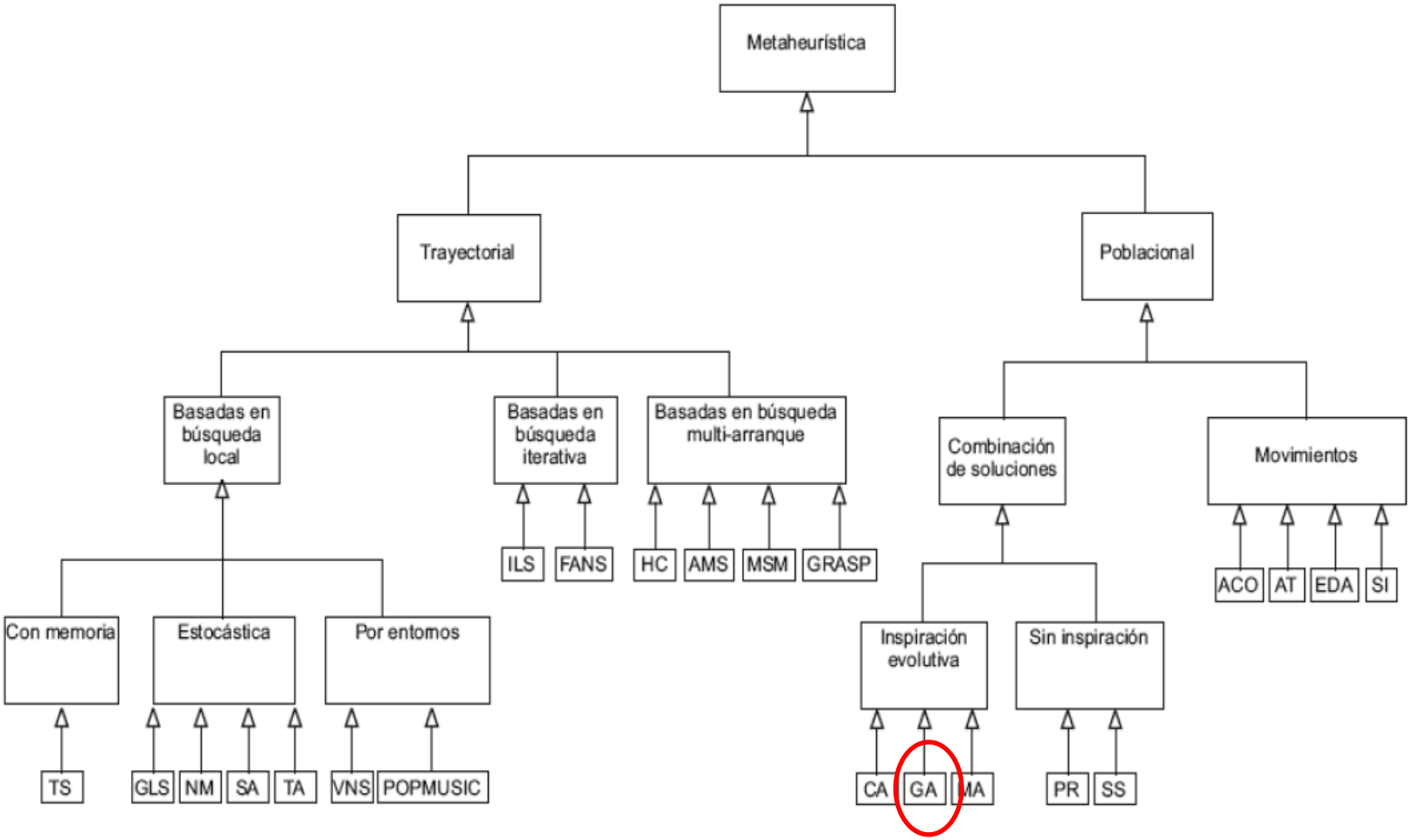
- CE: formada por modelos de evolución basados en poblaciones de elementos que representan soluciones a un problema
- Consiste en simular la evolución de esa población, dando lugar a una técnica de optimización probabilística. ¿Cómo se simula?



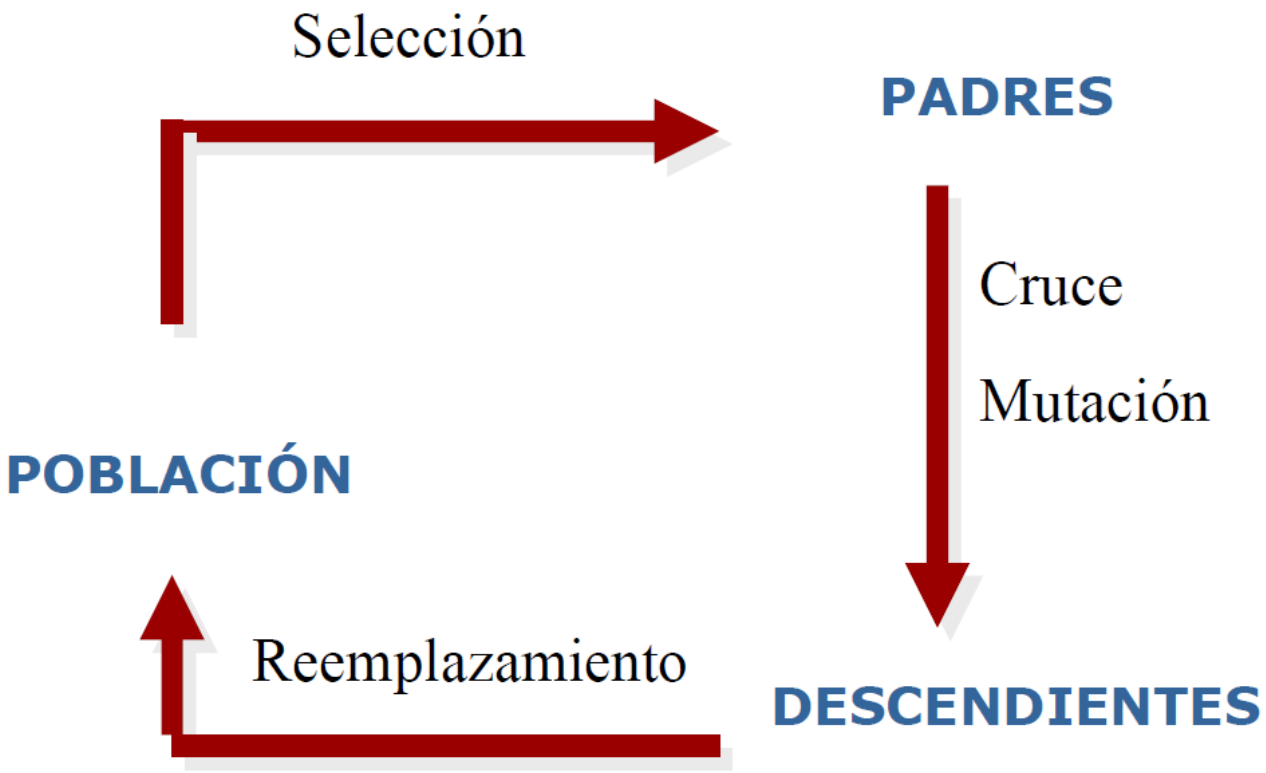
- Es una **meta-heurística**: parametrización de un procedimientos general que conduce a soluciones sub-óptimas (de una forma eficiente)

5. Computación Evolutiva (CE)

- Taxonomía (Duarte, 2004)

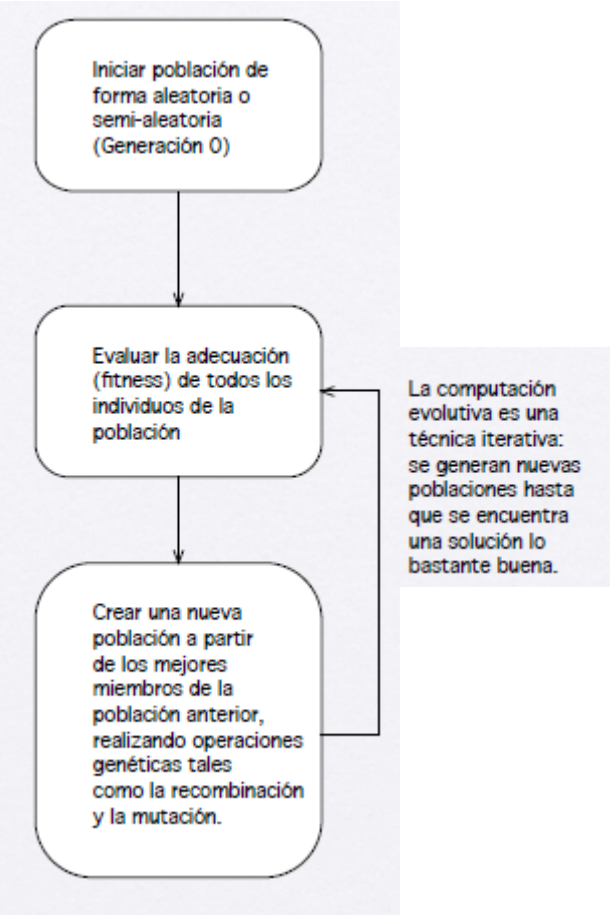


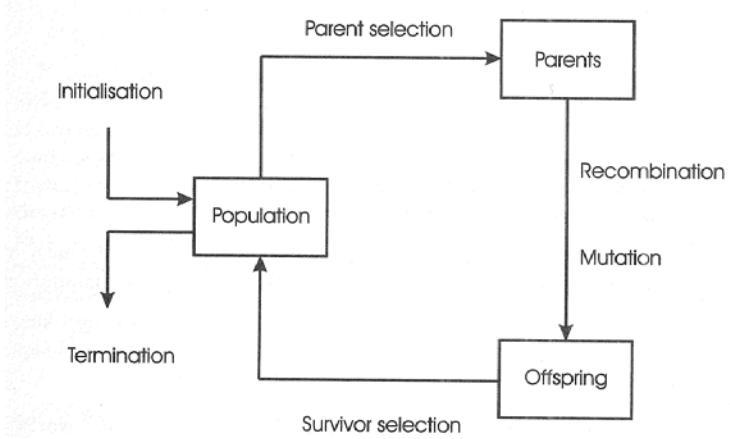
- Ciclo evolutivo



5. Computación Evolutiva (CE)

- Problema de optimización: la **calidad** de las soluciones (***fitness***) se mide con la función **objetivo** o de **coste**.





Procedimiento Algoritmo Genético

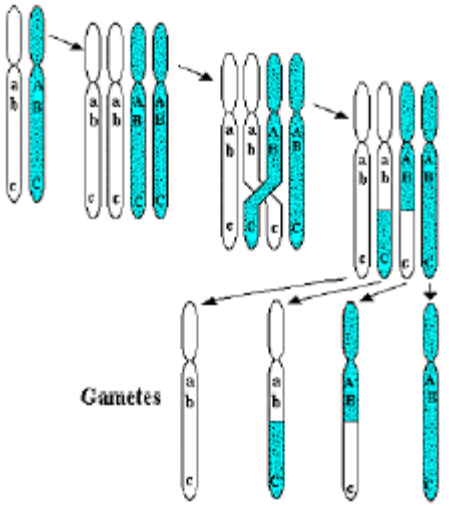
```
Inicio (1)
   t = 0;
   inicializar P(t);
   evaluar P(t);
   Mientras (no se cumpla la condición de parada) hacer
   Inicio(2)
        t = t + 1
        seleccionar P(t) desde P(t-1)
        recombinar P(t)
        mutación P(t)
        evaluar P(t)
   Final(2)
Final(1)
```

5. Computación Evolutiva (CE)

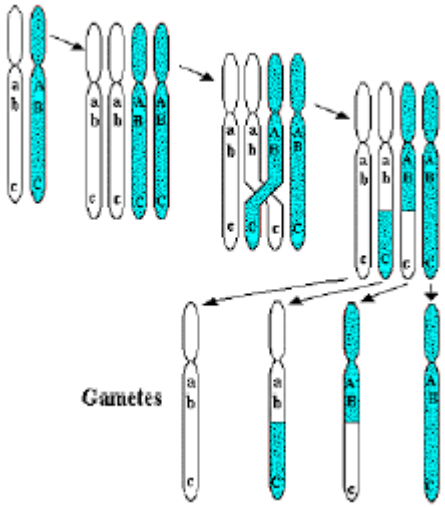
- Estrategia evolutiva:
  - Conjunto aleatorio de soluciones posibles: **población inicial**
  - Evaluamos la calidad (*fitness*) de las soluciones, como medida de su "adaptación al medio"
  - Escogemos una selección de los mejores como base para a siguiente generación:
    - **recombinación**: aplicado a los "padres" produce "hijos"
    - **mutación**: aplicado a un individuo, produce otro nuevo
  - Evaluamos la *fitness* de la descendencia. Sus elementos **compiten** con la población anterior para formar parte de la nueva generación.
  - Se itera el proceso hasta encontrar un candidato con la suficiente calidad, que sería una **solución**. O hasta llegar a un límite.

5. Computación Evolutiva (CE)

- Base de la estrategia:
  - Operadores que introducen **diversidad** en la población: **recombinación** y **mutación**
  - Selección: incrementa la **calidad media** de las soluciones en una población
- Aplicación combinada de ambas estrategias: mejora los valores de *fitness* en poblaciones consecutivas.



- **Naturaleza aleatoria** de la MH por varios motivos:
  - selección: **probabilidad** de escoger individuos con *fitness* baja
    - en esto, es similar a TS, SA
  - recombinación: selección aleatoria de los fragmentos
  - mutación: selección aleatoria del fragmento



- Soluciones candidatas: codificación numérica
  - entera
  - real

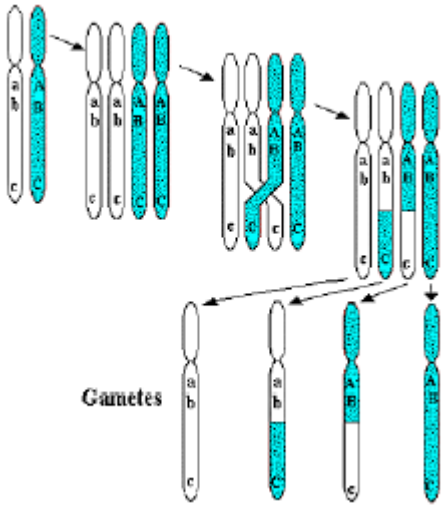
(**2.1**, -0.3, 4.6)

(9.1, **0.2, 7.2**)

↓

(**2.1,0.2, 7.2**)

(9.1,-0.3, 4.6)



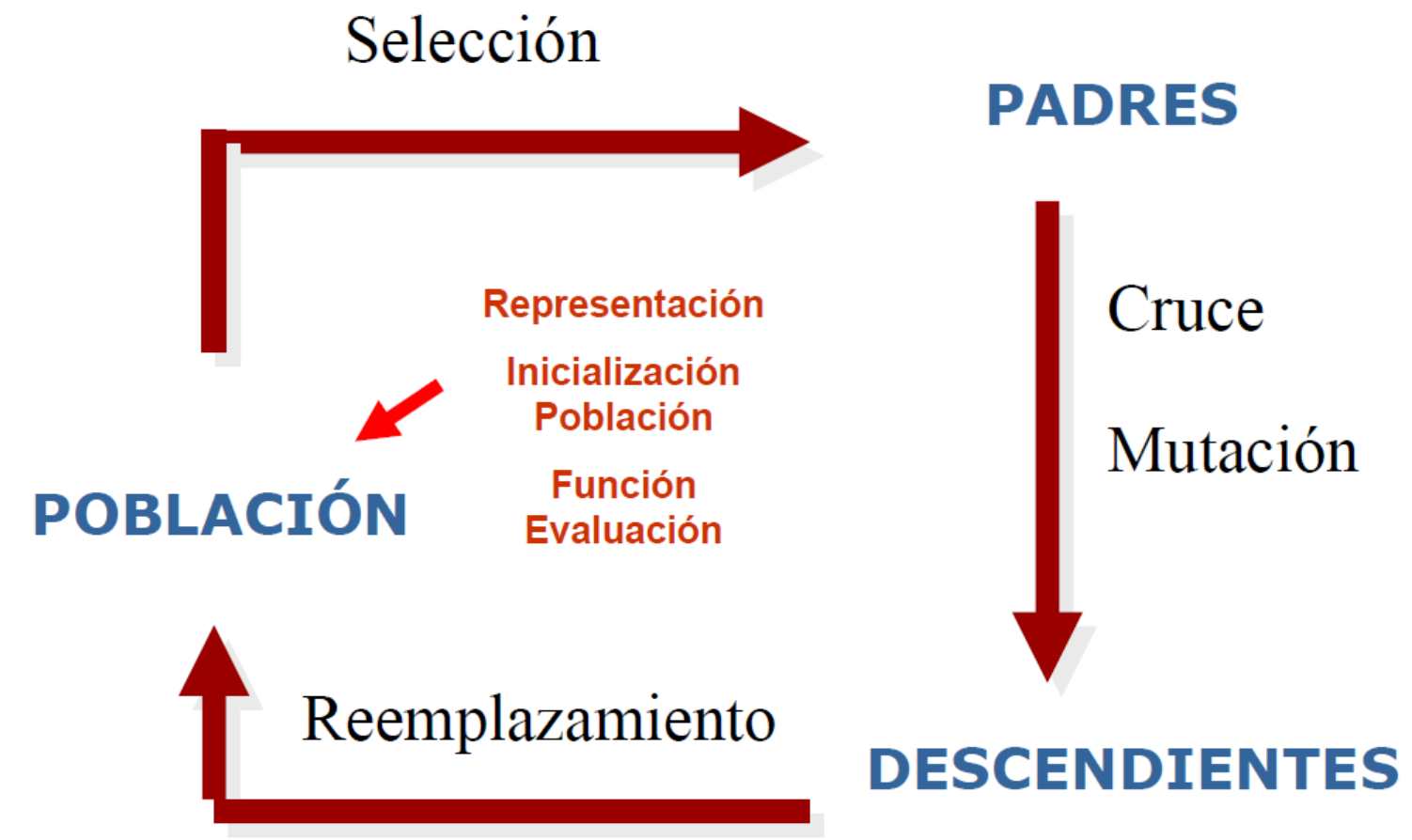
5. Computación Evolutiva (CE)

- Tipos de algoritmos evolutivos: comparten la estrategia general. Se diferencian en:
  - Operadores
  - Cómo incorporar las nuevas soluciones a la población

- Múltiples modelos de evolución de poblaciones:
  - Algoritmos genéticos: soluciones cadenas (*n* variables)
  - Programación genética: soluciones árboles

¿CÓMO SE CONSTRUYE UN AG?

RESUMEN



PROCESO ITERATIVO + CRITERIO DE PARADA

5. Computación Evolutiva (CE)

- Los AE siguen una estructura de ***generar soluciones y probar*** en el proceso de busca de la solución.
- La función de *fitness* representa una estimación **heurística** de la calidad de la solución.
- El proceso de busca va dirigido por los operadores: variación y selección
- ¿Cuáles son las componentes de un AE?
  - 1 representación: definición de los individuos
  - 2 función de *fitness*
  - 3 población
  - 4 mecanismo de selección
  - 5 operadores de variación: cruce y mutación
  - 6 mecanismo de reemplazo generacional

- Por lo tanto, los pasos en el diseño son...

DEPENDE DEL PROBLEMA

- **Diseñar una representación**
- **Decidir cómo inicializar una población**
- **Diseñar una correspondencia entre genotipo y fenotipo**
- **Diseñar una forma de evaluar un individuo**

COMPONENTES DEL ALGORITMO

- **Diseñar un operador de mutación adecuado**
- **Diseñar un operador de cruce adecuado**
- **Decidir cómo seleccionar los individuos para ser padres**
- **Decidir cómo reemplazar a los individuos**
- **Decidir la condición de parada**

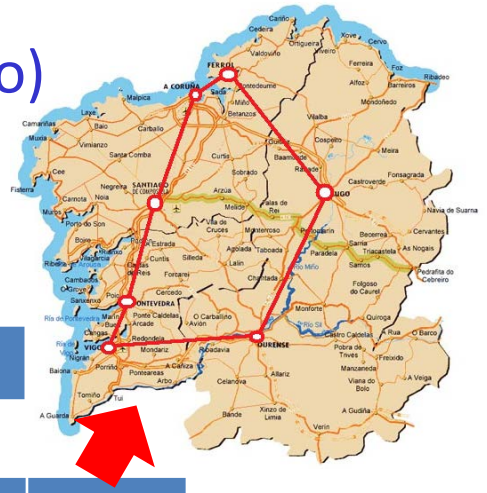
5.1 CE: representación

- 1. Representación:
  - modelado de la solución al problema: “real” -> “AE”
  - mecanismos para codificar la solución al problema (**fenotipos**) en forma de una cadena de genes (**genotipos**)
  - escogida la representación, se definen operadores **compatibles** con ella
  - **cromosoma** o **genotipo**: vector con la solución
  - **gen**: elemento mínimo del cromosoma (atributo)
  - **alelo**: valor de cada gen
- Ejemplo: TSP
- Genotipo

1	2	3	4	5	6

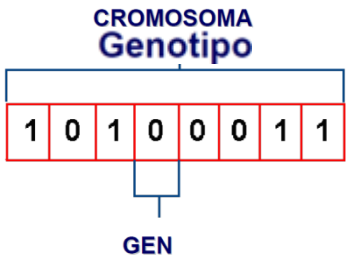
- Fenotipo

0	1	2	3	4	5	6	0



5.1 CE: representación

- Codificación binaria
  - representación de cada individuo mediante codificación discreta: secuencia de bits
  - cada gen es un bit. La codificación de cada alelo depende de la aplicación

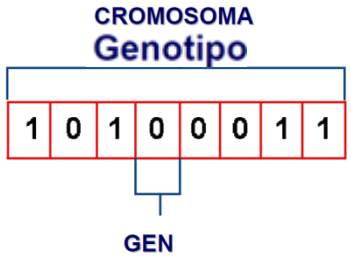


Fenotipo

- **Entero**
- **Número real**
- **secuencia**
- ...
- **Cualquier otra?**

5.1 CE: representación

- Codificación binaria



El fenotipo pueden ser números enteros

Fenotipo:

= 163

$1*2^7 + 0*2^6 + 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =$

128 + 32 + 2 + 1 = 163

Ejemplo: un número entre 2.5 y 20.5 utilizando 8 dígitos binarios

Fenotipo:

= 13.9609

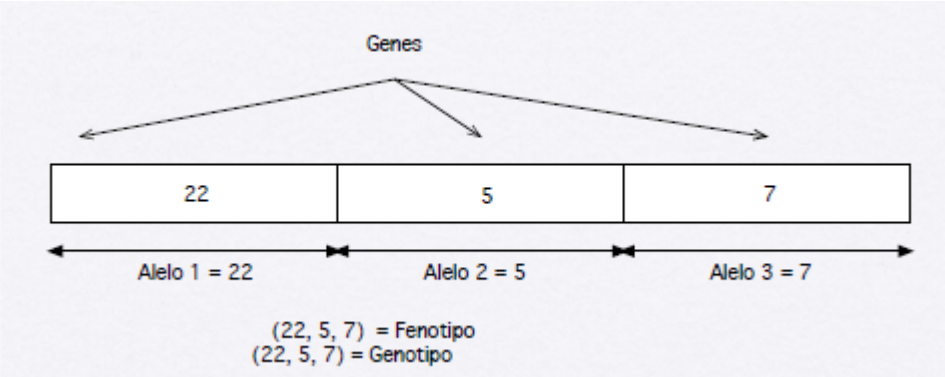
$$x = 2.5 + \frac{163}{256}(20.5 - 2.5) = 13.9609$$

5.1 CE: representación

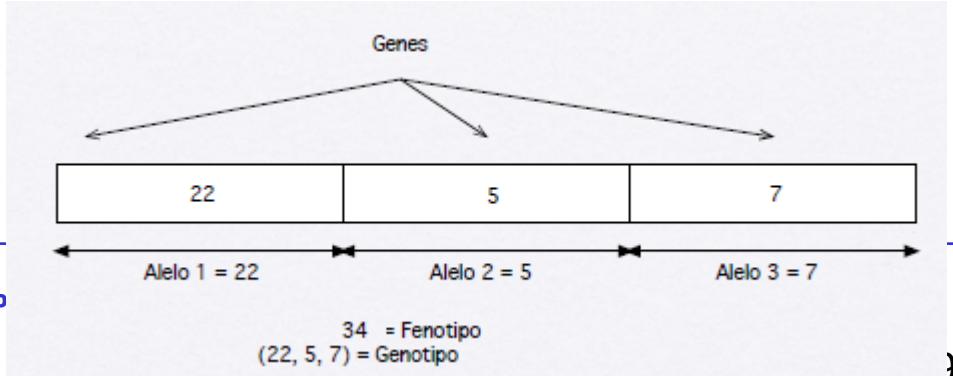
- Codificación real
  - cada individuo: secuencia de números reales
  - el fenotipo (solución) no tiene porqué coincidir con el genotipo (individuo). Puede ser el resultado de evaluarlo con una función determinada

Ejemplos:

- fenotipo **coincide** con genotipo



- fenotipo como **función** (suma) del genotipo



5.1 CE: representación

- Representación de orden
  - cada individuo se representa como una **permutación** de valores
  - precisan operadores especiales (con **reparación**), para asegurar que el resultado de cada operación (e.g. recombinación, mutación) sigue siendo una permutación

Ejemplo: TSP

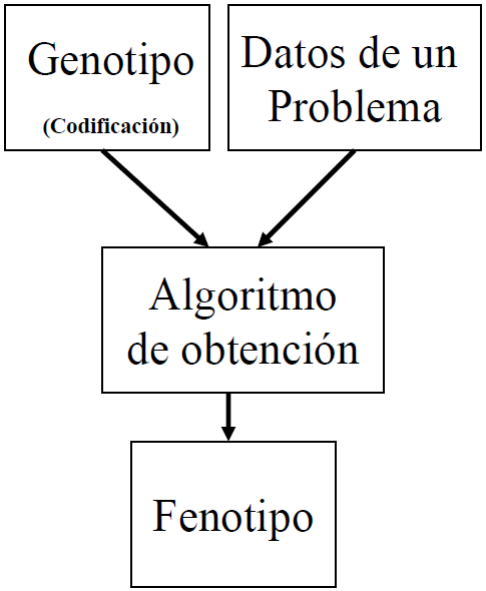
7	3	6	8	2	4	1	5

5.1 CE: representación

- Paso de genotipo a fenotipo: de la representación a la solución
  - Paso obvio a veces. Ejemplo: TSP

7	3	6	8	2	4	1	5

  - Paso complejo a veces: el genotipo puede un conjunto de parámetros de entrada de otro algoritmo que es el que obtiene la solución final. Ejemplo: simulación numérica, visualización 3D, sistemas basado en conocimiento (reglas), …



5.2 CE: inicialización

- Población
  - representa las posibles soluciones
  - es la unidad (elemento mínimo) de la evolución
  - Criterio de diversidad: medida del número de diferentes soluciones presentes en la población
- Inicialización de la población:
  - aleatoria (uniforme en el rango de cada parámetro), con el objetivo de producir soluciones repartidas uniformemente en el **espacio de búsqueda**.
    - binaria: 0 ó 1 (p=0.5)
    - real: distribución uniforme sobre un intervalo [a, b]
  - Por criterios heurísticos
  - Combinación de ambas

5.3 *Fitness* de los individuos

- Función de calidad (*fitness*): representa los requisitos a los que debe adaptarse la población
  - Cuantifica la mejora de la población
  - Etapa de mayor coste: 95% CPU
  - Tipos:
    - Función simple: resultado numérico, real, ...
    - Procedimiento
    - Simulador
    - Cualquier proceso externo: experimentos, ...
  - En otros casos: vector, ordenable o no
  - En los problemas (típicos) de optimización: **función objetivo**.
  - Si hay **múltiples objetivos** a satisfacer, debe alcanzarse una solución de **compromiso** entre todos los objetivos (media –ponderada-, máximo, mínimo, ...)

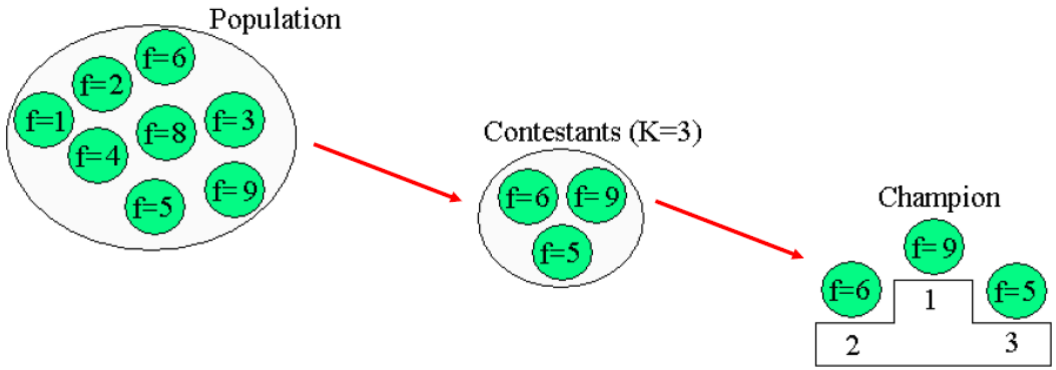
5.4 CE: selección

- Mecanismo de selección
  - Trata de distinguir entre los individuos de una población de acuerdo con su calidad
  - Se denomina "padre"/"madre" al individuo seleccionado para formar parte de la **descendencia** (***offspring***, candidatos a permanecer en la nueva generación)
  - La selección es probabilística: los individuos de mayor calidad tienen mas probabilidades de permanecer, pero también tienen opciones a permanecer los de menor calidad.
    - diversidad, como alternativa al "*local trapping*"

  – Idea de **presión selectiva**: determina el grado en el que el proceso está dirigido por los mejores individuos

5.4 CE: selección

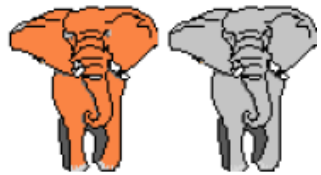
- Selección aleatoria
- Selección por torneo
  - Seleccionar el mejor de *k* individuos escogidos aleatoriamente:
    torneo **binario**: *k*=2; **ternario**: *k*=3
  - Ejemplo: torneo con *k*=3

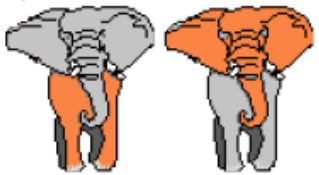


- Orden lineal: población **ordenada** por *fitness*. Se asocia una **probabilidad de selección** que depende de ese **orden**.
- Ruleta: probabilidad de selección **proporcional al *fitness*** de cada individuo

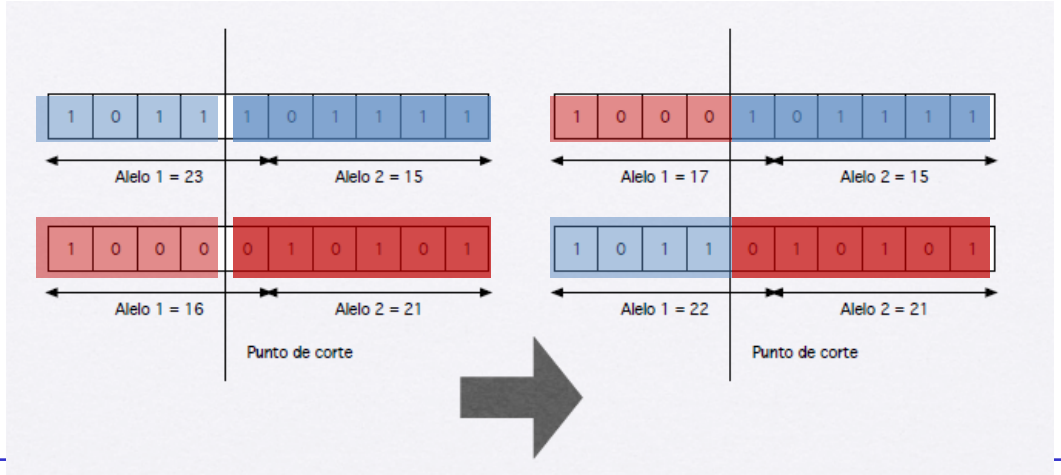
5.5 CE: operadores de variación. Cruce

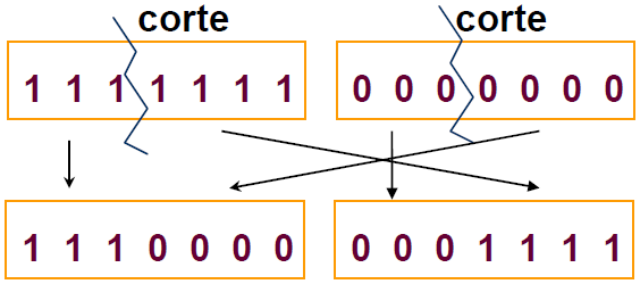
- Operadores de cruce: mezcla de los genotipos de **dos** padres.
  - Debe diseñarse de forma **consistente con la representación**, para dar lugar a individuos válidos
  - **Probabilidad alta** de darse: $P_C \in [0.6, 0.9]$. En caso de no darse, el resultado son los propios padres





- Algunos ejemplos: punto/s de cruce **aleatorio**
  - –Caso binario: *one-point random crossover*





5.5 CE: operadores de variación. Cruce

•caso de representación real: "interpolación" o combinación aritmética convexa con parámetro α.

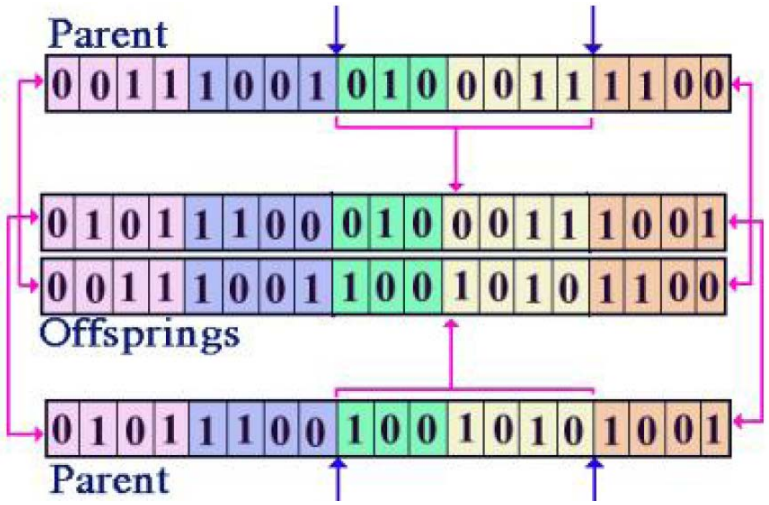
Cruce de (22,10,4) y (12,6,9)

$\alpha = 0.1$

$\alpha(22, 10, 4) + (1 - \alpha)(12, 6, 9) = (13, 6.4, 8.5)$

$(1 - \alpha)(22, 10, 4) + \alpha(12, 6, 9) = (21, 9.6, 5.8)$

• Cruce de dos puntos



5.5 CE: operadores de variación. Cruce

•caso de representación real: "interpolación" o combinación aritmética convexa con parámetro α.

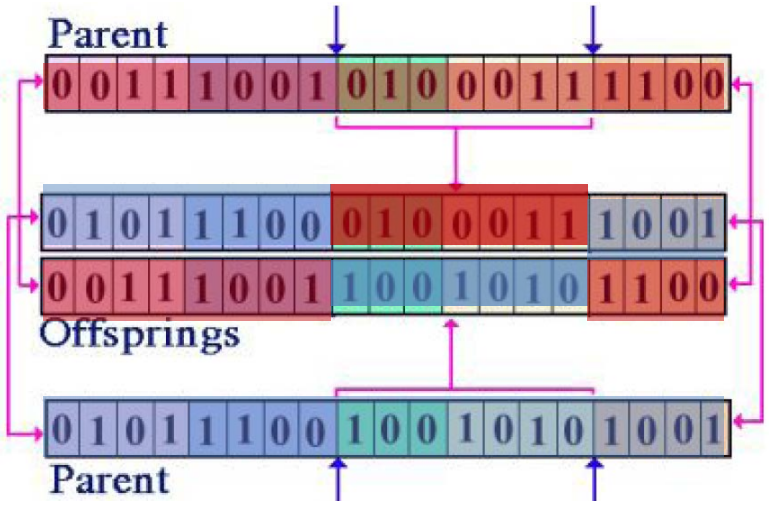
Cruce de (22,10,4) y (12,6,9)

$\alpha = 0.1$

$\alpha(22, 10, 4) + (1 - \alpha)(12, 6, 9) = (13, 6.4, 8.5)$

$(1 - \alpha)(22, 10, 4) + \alpha(12, 6, 9) = (21, 9.6, 5.8)$

• Cruce de dos puntos



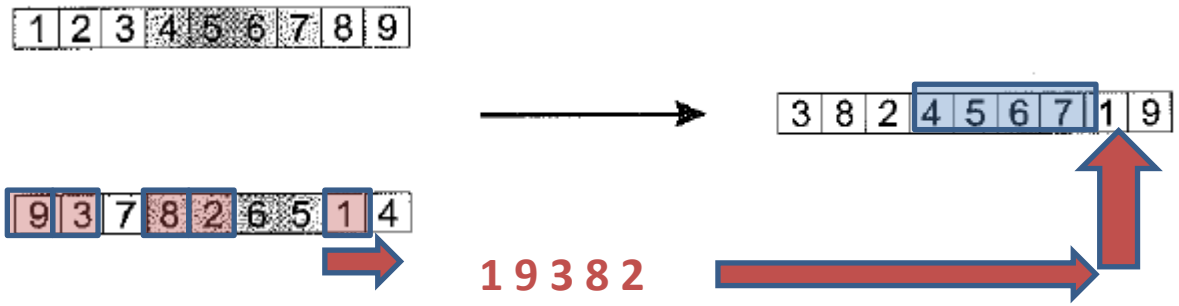
5.5 CE: operadores de variación. Cruce

• Representación de orden: deben utilizarse operaciones que incluyan **mecanismos de reparación** que mantengan la representación (sigan siendo permutaciones)

– Ejemplo: *order crossover* (OX)

• Copiar un segmento aleatorio del P1 en H1

• Completar H1 con el resto de alelos en el orden que aparecen en P2, a partir del segundo punto de corte



5.5 CE: operadores de variación. Cruce

- Cruce para representación de orden PMX (*partially mapped crossover*)

  - Se elige una subcadena central y se establece una correspondencia por posición entre las asignaciones contenidas en ellas
  - Cada hijo contiene la subcadena central de uno de los padres y el mayor número posible de asignaciones en las posiciones definidas por el otro padre. Cuando se forma un ciclo, se sigue la correspondencia fijada para incluir una asignación nueva

    $Padre_1$ = (1 2 3 | 4 5 6 7 | 8 9)
    $Padre_2$ = (4 5 3 | 1 8 7 6 | 9 2)
    $Hijo'_1$ = (* * * | 1 8 7 6 | * *)
    $Hijo'_2$ = (* * * | 4 5 6 7 | * *)

    Correspondencias: (1-4, 8-5, 7-6, 6-7)

    $Hijo_1$ = (1-4 2 3 | 1 8 7 6 | 8-5 9) = (4 2 3 | 1 8 7 6 | 5 9)
    $Hijo_2$ = (4-1 5-8 3 | 4 5 6 7 | 9 2) = (1 8 3 | 4 5 6 7 | 9 2)

5.5 CE: operadores de variación. Cruce

- Cruce para representación de orden CX (cycle crossover)

  - Partiendo de la asignación i del primer padre, CX toma la siguiente asignación j del segundo padre como aquella en la misma posición que i, y sitúa j en la misma posición que ocupa en el primer padre

    $Padre_1 = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$

    $Padre_2 = (2\ 4\ 6\ 8\ 7\ 5\ 3\ 1)$

  - Aleatoriamente podemos escoger 1 ó 2 para la primera posición, supongamos que escogemos 1. Esto implica escoger 8 en la posición 8, lo cual supone escoger 4 en la posición 4 y, por tanto, 2 en la posición 2

    $(1\ *\ *\ *\ *\ *\ *\ *) \rightarrow (1\ *\ *\ *\ *\ *\ *\ 8) \rightarrow (1\ *\ *\ 4\ *\ *\ *\ 8) \rightarrow (1\ 2\ *\ 4\ *\ *\ *\ 8)$

  - Se ha formado un ciclo. Se escoge aleatoriamente una ciudad entre 3 ó 6 para la tercera posición, supongamos que escogemos 6

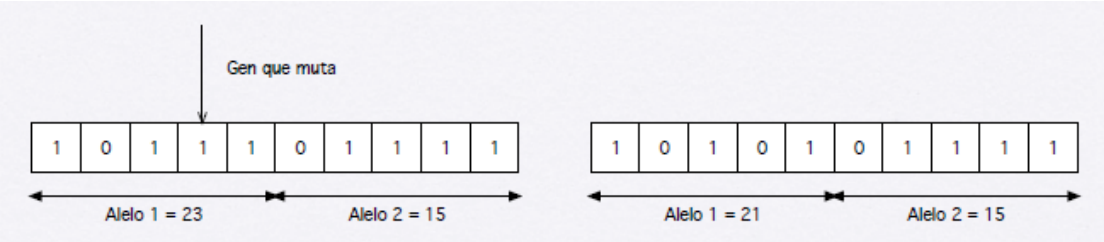
    Hijo = $(1\ 2\ 6\ 4\ *\ *\ *\ 8)$

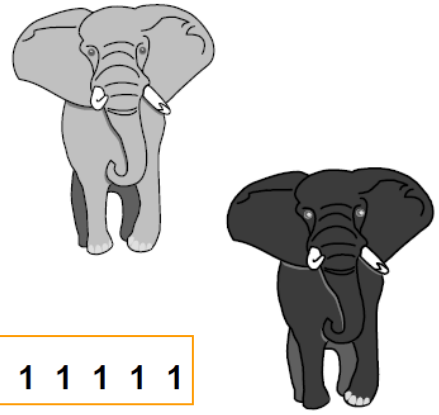
  - Esto implica escoger obligatoriamente la ciudad 5 en la posición 6, lo que implica 7 en la posición 5 y 3 en la posición 7

    Hijo = $(1\ 2\ 6\ 4\ 7\ 5\ 3\ 8)$

5.5 CE: operadores de variación. Mutación

- Operadores de mutación: operación **unaria** aplicada a un genotipo. Produce un descendente ligeramente modificado.
  - Debe permitir alcanzar **cualquier parte del espacio** de busca
  - El tamaño de la mutación debe ser controlado
  - Debe dar lugar a genotipos válidos
  - Se aplica con probabilidad p_m muy baja **sobre cada gen**, tras haber aplicado previamente el operador de cruce
- Ejemplo binario:





antes 1 1 1 1 1 1 1

después 1 1 1 0 1 1 1

gen mutado

5.5 CE: operadores de variación. Mutación

- Operadores de mutación para **representación entera**:
  - *Random resetting*: se escoge aleatoriamente con probabilidad p_m un valor de entre los posibles (dentro del rango válido)
- Operadores de mutación para **representación de orden**:
  - Intercambio recíproco: entre dos valores de la permutación, seleccionados aleatoriamente

  - Inserción: se seleccionan dos valores y se inserta uno al lado del otro

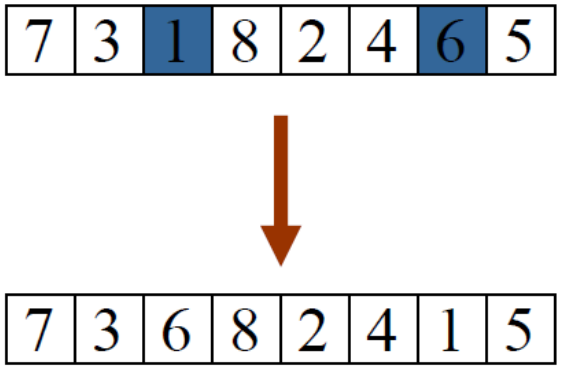
  - Mezcla: se barajan los valores en un rango

5.5 CE: operadores de variación. Mutación

- Ejemplo de representación real: modificación aleatoria, típicamente siguiendo una gaussiana normal N(0,σi)
  - media 0
  - σ desviación típica

$$x'_i = x_i + N(0,\sigma_i)$$

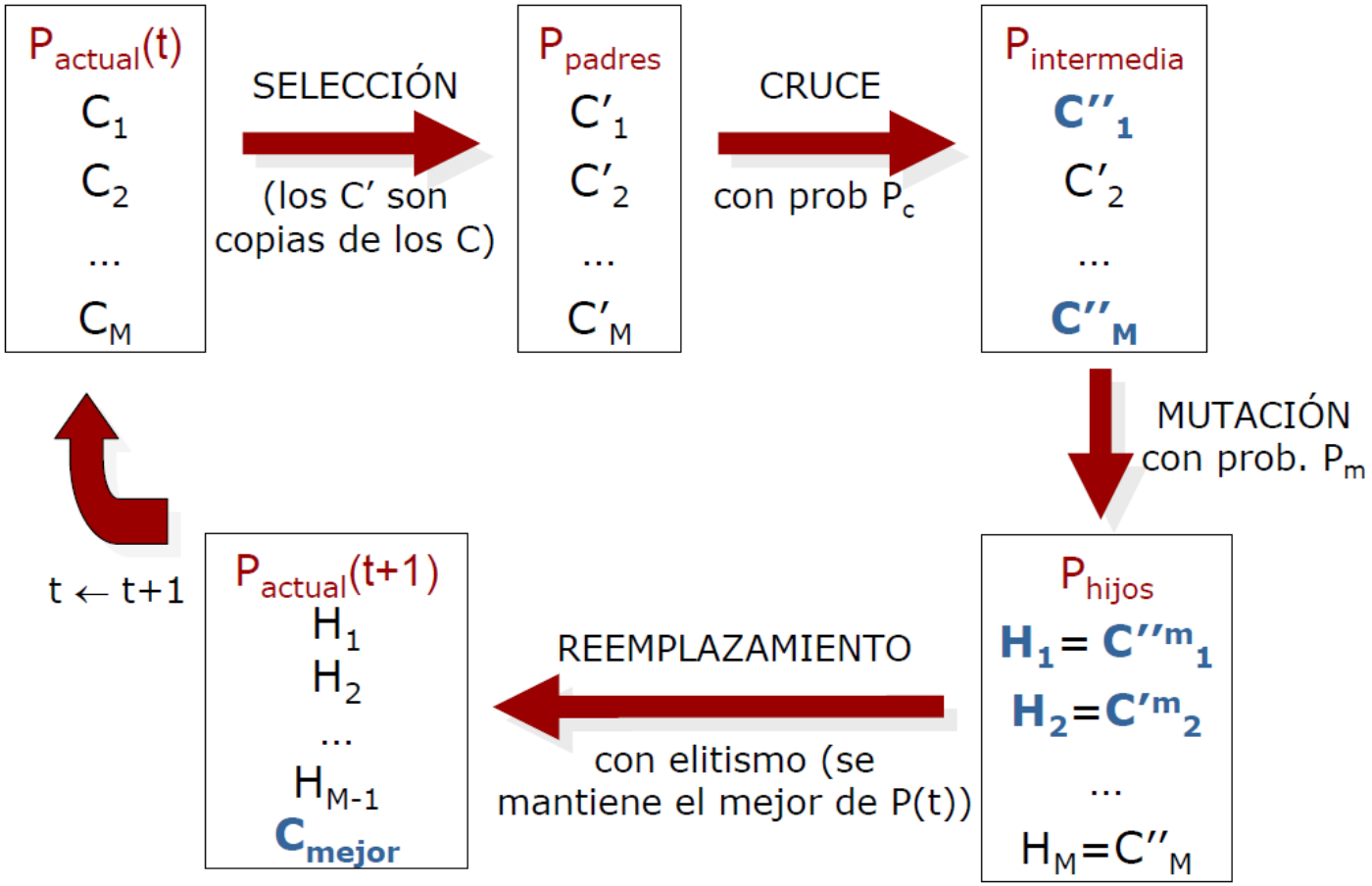
- Caso de orden: intercambio de genes



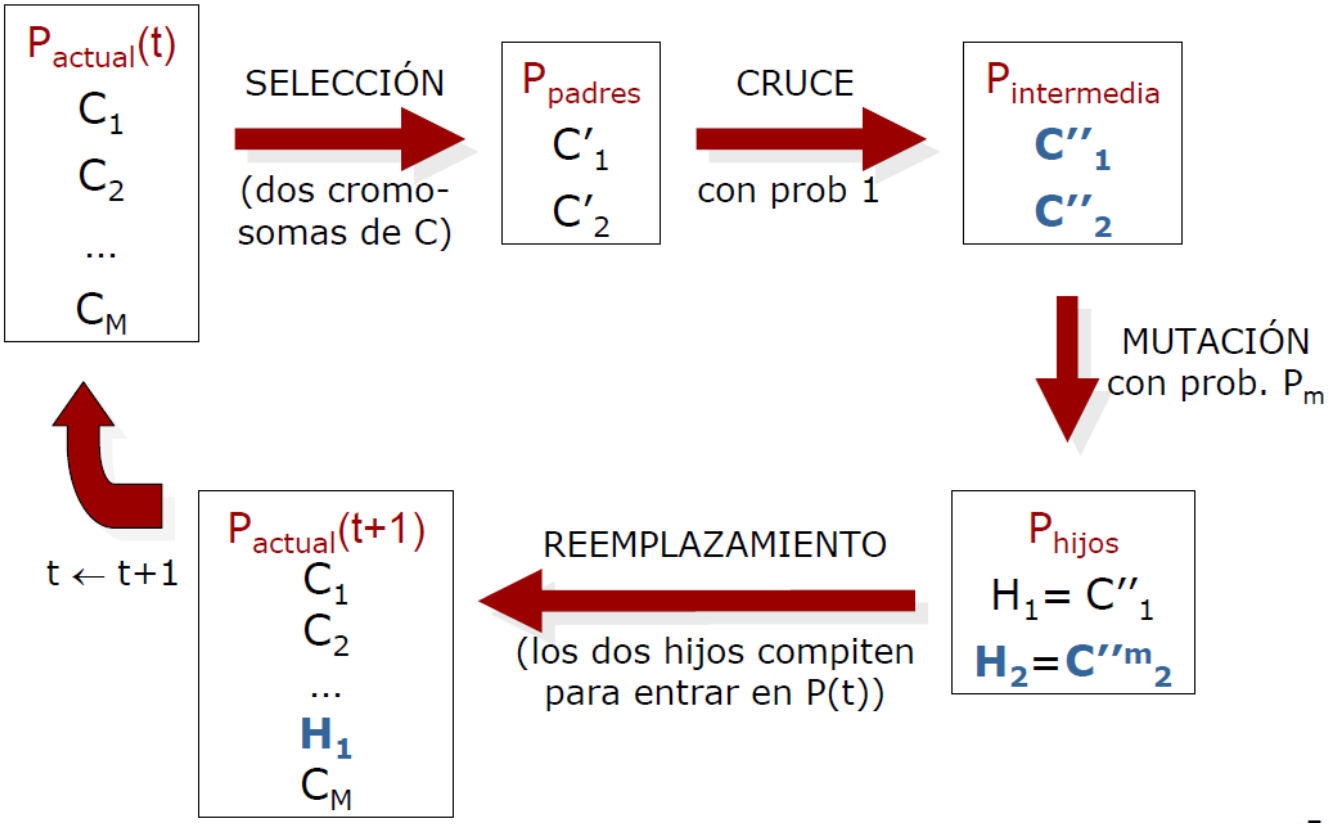
5.6 CE: mecanismos de reemplazo

- Reemplazo: es el mecanismo que determina la composición de la nueva generación. Establece que individuos se mantienen en la población y cuales son sustituidos por la descendencia
  - Establece la supervivencia de la descendencia para la nueva generación
  - Es similar a la selección, pero en otra etapa: después de producirse la descendencia: torneo, lineal, aleatorio, ruleta, ...
- Inserción de la descendencia:
  - **modelo generacional**: la descendencia sustituye completamente a la generación anterior
  - **modelo permanente**: los descendientes se insertan en la población actual, reemplazando a otros individuos.
- Puede contemplarse **elitismo**, para no reemplazar a los mejores individuos de la solución actual.

modelo generacional

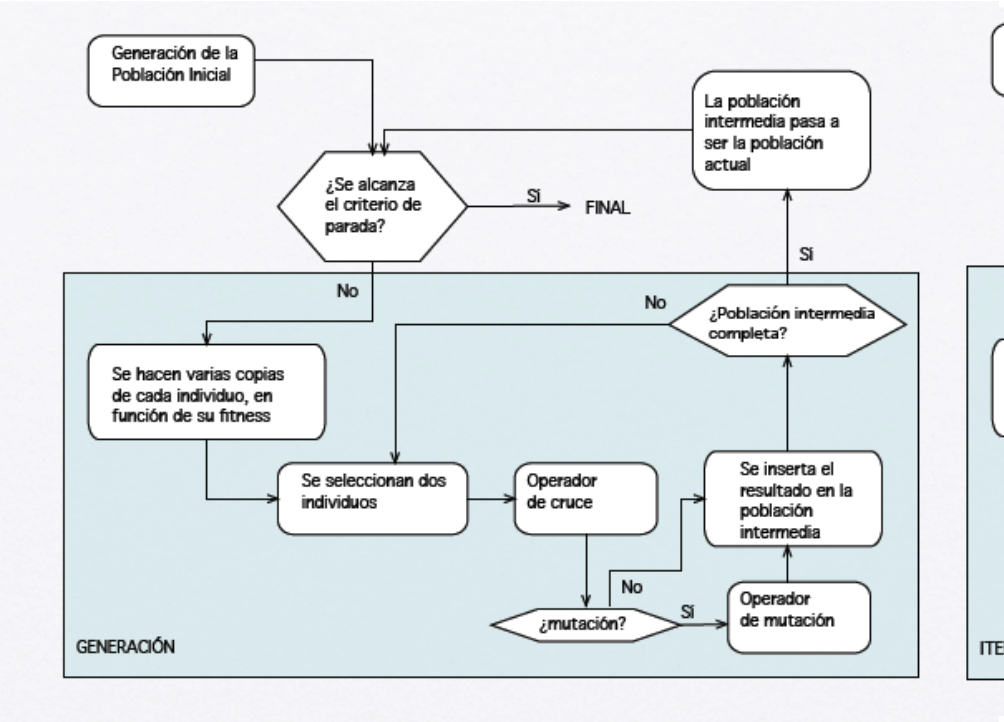


modelo permanente

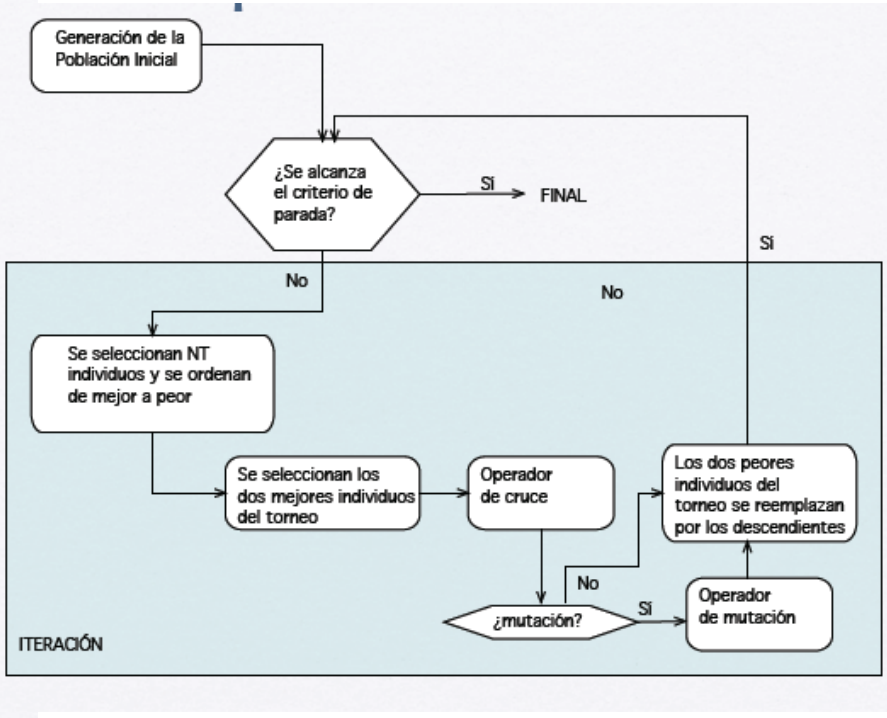


5.6 CE: mecanismos de reemplazo

Generacional



Permanente



5.6 CE: mecanismos de reemplazo

- En el modelo **permanente** hay diferentes estrategias de reemplazo:
  - Reemplazar al *peor de la población* (RW). Produce **alta presión selectiva**.
  - *Torneo Restringido* (RTS): reemplaza al mas parecido de entre *w* (*w*=3, …). Mantiene una **cierta diversidad**.
  - *Peor entre semejantes* (WAMS): reemplaza el peor cromosoma del conjunto de *w* (*w*=3, …) padres mas parecidos al descendiente (seleccionados de toda la población). Busca **equilibrio entre diversidad y presión selectiva**.
  - Algoritmo de *Crowding Determinístico* (DC): el hijo reemplaza al padre mas parecido. **Mantiene diversidad**.

5.7 CE: condición de parada

- **Condición de parada**: consiste en fijar criterios para finalizar la ejecución del algoritmo
  - Estrategias típicas:
    - cuando se alcanza una solución con una precisión prefijada (idealmente, la solución óptima
    - cuando los recursos de CPU son limitados: se establece un número de iteraciones máximo
    - también cuando no hay una mejora significativa entre poblaciones sucesivas
    - la diversidad es insuficiente

- Recordemos siempre que no está asegurada la solución óptima: puede haber óptimos locales

5.8.CE: diseño y consideraciones

- Experimentación:
  - NUNCA se deben obtener conclusiones con una ejecución ÚNICA
  - SIEMPRE se deben realizar varias ejecuciones independientes y proporcionar como resultados
    - valores estadísticos: media y desviación típica
    - análisis de significancia: tests estadísticos
  - Los ejemplos deben ser representativos del caso real, no demasiado triviales ni simples

- Diversidad:
  - Alta presión selectiva = falta de diversidad = todos los individuos parecidos → **convergencia prematura** a los vecinos más próximos
  - Solución:
    - introducir mecanismos de diversidad
    - **reinicialización** del algoritmo

5.8.CE: diseño y consideraciones

- **Exploración** vs **explotación**:

  - **Exploración**: zonas desconocidas del espacio de búsqueda.
    exceso de exploración: busca aleatoria, falta de convergencia

  - **Explotación**: mejora local de los individuos
    exceso de explotación: busca local y convergencia a un óptimo local

5.9.CE: ejemplos

- Ejercicio 1: problema de las 8 reinas
  - Espacio de busca P: todas las posibles configuraciones de reinas p
  - Función de *fitness*: número de reinas atacadas.
    - 0: solución
    - inverso: $1/(q(p)+\varepsilon)$
  - Genotipos: vector de índices {1, 2, 3, 4, 5, 6, 7, 8}:
    - evita automáticamente restricciones de fila/columna
  - Operadores:
    - Mutación: intercambio de dos posiciones aleatorias
    - ¿Cruce? Dos padres P1, P2
      - punto aleatorio *i*
      - corte en dos segmentos P11-P12, P21-P22
      - Descendencia: F11=P11, F21=P21. Completar resto de P2→F1 y de P1→F2

5.9.CE: ejemplos

- Ejercicio 1 (cont.): problema de las ocho reinas
  - Mecanismo de selección: 2 padres mejores de 5 aleatorios
  - Manejo de la población: 2 padres→2 hijos. Población de *n*+2, de los que se seleccionan los *n* mejores. Eliminar los dos peores

- Ejercicio 2: problema de la mochila {0, 1}.
  - N objetos con peso *p_i* y valor *v_i*.

- Ejercicio 3: problema del viajante de comercio (TSP)

Un posible planteamiento (distinto del propuesto en la práctica)

Representación de orden

(3 5 1 13 6 15 8 2 17 11 14 4 7 9 10 12 16)

17 ciudades

Objetivo: Suma de la distancia entre las ciudades.

Población: 61 cromosomas - Elitismo

Cruce: OX (P_c = 0,6)

Mutación: Inversión de una lista (P_m = 0,01 – cromosoma)

- Ejercicio 3: problema del viajante de comercio (TSP)

1. **Generación de la solución inicial**: aleatoria o greedy
2. **Esquema de representación**: Representación de orden mediante permutación {1,…,n}
3. **Selección**: torneo binario
4. **Enfoque:** generacional con elitismo (1 individuo)
5. **Operador de cruce**: Operador OX.
6. **Operador de mutación**: intercambio de 2 genes
7. **Función objetivo (minimización)**: La distancia recorrida por el viajante.

$$C(S) = \sum_{i=1}^{n-1} (D[S[i], S[i+1]]) + D[S[n], S[1]]$$

5. Bibliografía

- Referencias básicas:
  - Francisco Herrera (Univ. Granada).
    - Algorítmica (Tema 7): algoritmos genéticos (transparencias). http://sci2s.ugr.es/graduateCourses/Algoritmica
    - Metaheurísticas (Tema 5): Metaheurísticas basadas en poblaciones. http://sci2s.ugr.es/graduateCourses/Metaheuristicas
  - A.E. Eiben and J.E. Smith, Introduction to Evolutionary Computing, Springer, 2007. ISBN: 978-3-540-40184-1.
    - Cap. 2: What is an evolutionary algorithm?
    - Cap. 3: Genetic algorithms
- J.T. Palma, R. Marín. Inteligencia Artificial: técnicas, métodos y aplicaciones. Ed. McGraw-Hill, 2008. Capítulo 11 “Computación evolutiva”.
  - 11.1: Introducción
  - 11.2: Un algoritmo genético simple
  - 11.3: Fundamentos de los algoritmos genéticos
  - 11.4.1-11.4.4: Diseño de algoritmos evolutivos (excepto de 11.4.5 en adelante)

5. Bibliografía

- C. Reeves. Genetic Algorithms. Handbook of Metaheuristics. Kluwer Academics. (2003). Capítulo 3, 55-82.

- Mas material avanzado:
  - Francisco Herrera (Univ. Granada): Asignatura Bioinformática. http://sci2s.ugr.es/graduateCourses/Algoritmica.