BIOINFORMÁTICA
2013 - 2014

PARTE I. INTRODUCCIÓN

- Tema 1. Computación Basada en Modelos Naturales

PARTE II. MODELOS BASADOS EN ADAPTACIÓN SOCIAL (Swarm Intelligence)

- Tema 2. Introducción a los Modelos Basados en Adaptación Social
- Tema 3. Optimización Basada en Colonias de Hormigas
- Tema 4. Optimización Basada en Nubes de Partículas (Particle Swarm)

PARTE III. COMPUTACÍON EVOLUTIVA

- Tema 5. Introducción a la Computación Evolutiva
- Tema 6. Algoritmos Genéticos I. Conceptos Básicos
- Tema 7. Algoritmos Genéticos II. Diversidad y Convergencia
- Tema 8. Algoritmos Genéticos III. Múltiples Soluciones en Problemas Multimodales
- Tema 9. Estrategias de Evolución y Programación Evolutiva
- Tema 10. Algoritmos Basados en Evolución Diferencial (Diferential Evolution – DE)
- Tema 11. Modelos de Evolución Basados en Estimación de Distribuciones (EDA)
- Tema 12. Algoritmos Evolutivos para Problemas Multiobjetivo
- Tema 13. Programación Genética
- Tema 14. Modelos Evolutivos de Aprendizaje

PARTE IV. OTROS MODELOS DE COMPUTACIÓN BIOINSPIRADOS

- Tema 15. Sistemas Inmunológicos Artificiales
- Tema 16. Otros Modelos de Computación Natural/Bioinspirados

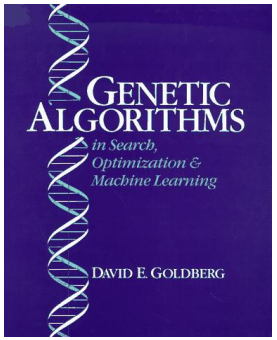
BIOINFORMÁTICA

TEMA 6: ALGORITMOS GENÉTICOS I: CONCEPTOS BÁSICOS

1. INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS
2. MODELOS: GENERACIONAL vs ESTACIONARIO
3. ¿CÓMO SE CONSTRUYE UN AG?
4. SOBRE SU UTILIZACIÓN
5. EJEMPLO: VIAJANTE DE COMERCIO
6. CONCLUSIONES

BIBLIOGRAFÍA

D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, 1989.



Clásico en AG

Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer Verlag, 1996.

T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Institute of Physics Publishers, 1997.

A.E. Eiben, J.E. Smith. Introduction to Evolutionary Computing. Springer, 2003.

1. INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS

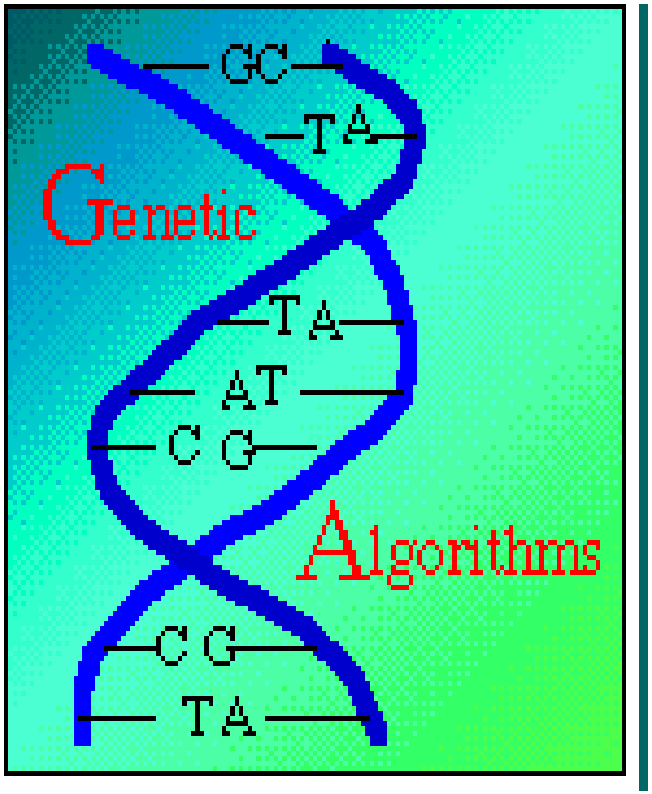
- ¿QUÉ ES UN ALGORITMO GENÉTICO?
- LOS INGREDIENTES
- El CICLO DE LA EVOLUCIÓN
- ESTRUCTURA DE UN ALGORITMO GENÉTICO

¿Qué es un Algoritmo Genético?

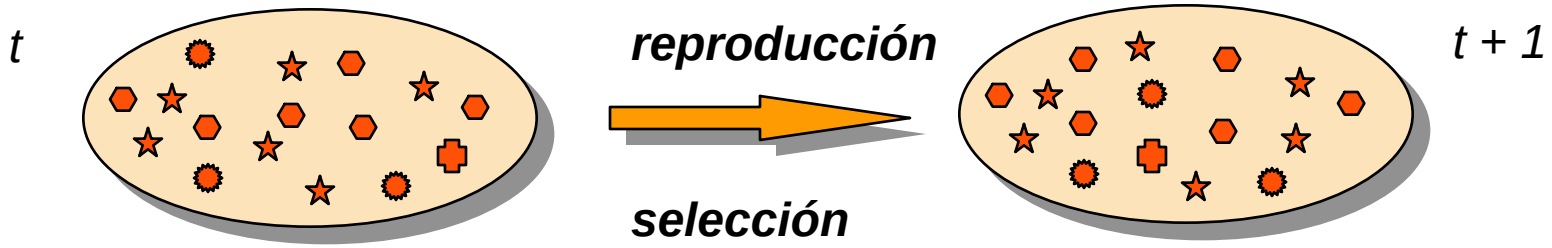
Los Algoritmos Genéticos

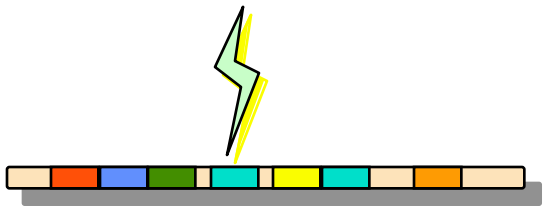
son algoritmos de optimización
búsqueda
y aprendizaje
inspirados en los procesos de

Evolución Natural
y
Evolución Genética

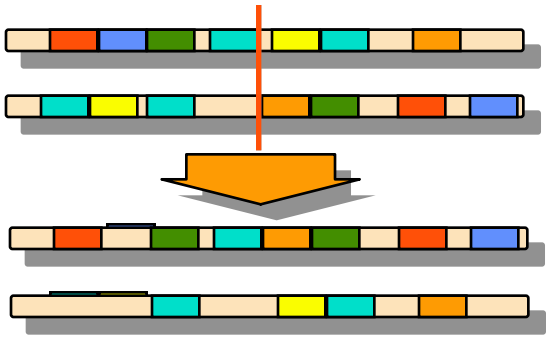


Los Ingredientes



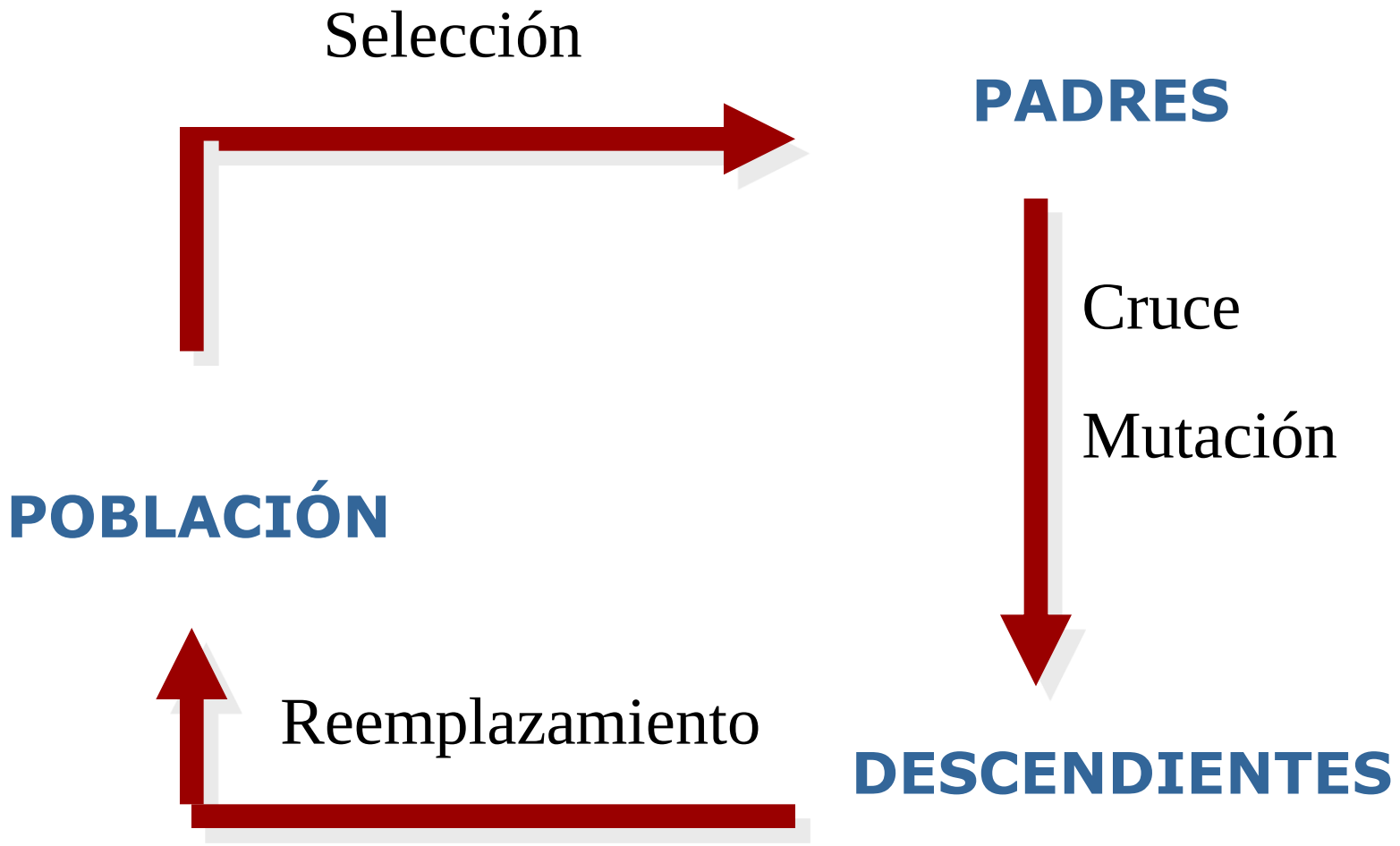


mutación



cruce

El ciclo de la Evolución



Estructura de un Algoritmo Genético

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)
```

2. MODELOS: GENERACIONAL vs ESTACIONARIO

Modelo generacional. Durante cada iteración se crea una población completa con nuevos individuos.

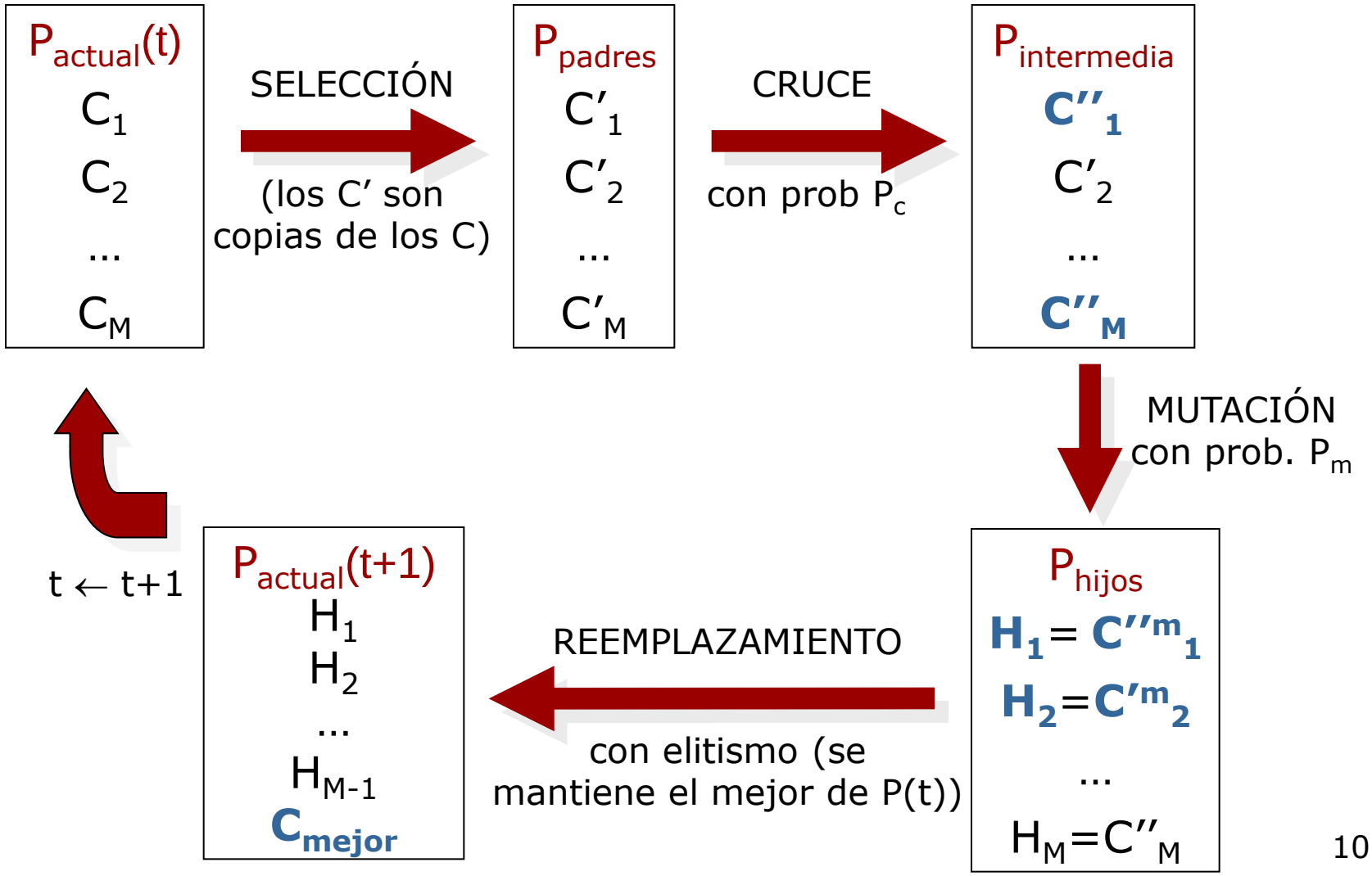
La nueva población reemplaza directamente a la antigua.

Modelo estacionario. Durante cada iteración se escogen dos padres de la población (diferentes mecanismos de muestreo) y se les aplican los operadores genéticos.

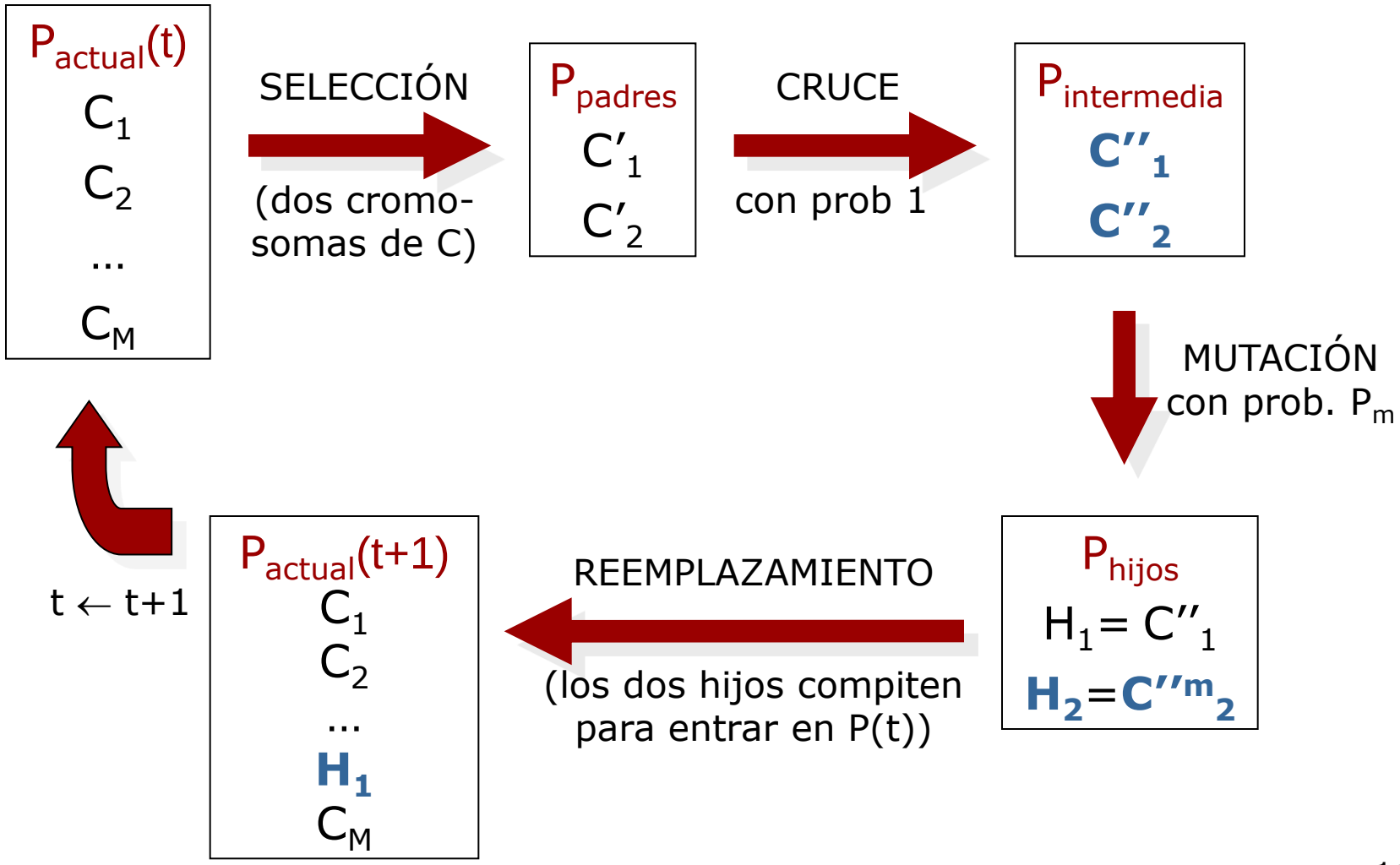
El/los descendiente/s reemplazan a uno/dos cromosoma/s de la población inicial.

El modelo estacionario es elitista. Además produce una presión selectiva alta (convergencia rápida) cuando se reemplazan los peores cromosomas de la población.

Modelo Generacional



Modelo Estacionario



3. ¿CÓMO SE CONSTRUYE UN AG?

Los pasos para construir un Algoritmo Genético

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

DEPENDE DEL PROBLEMA

COMPONENTES DEL ALGORITMO

Representación

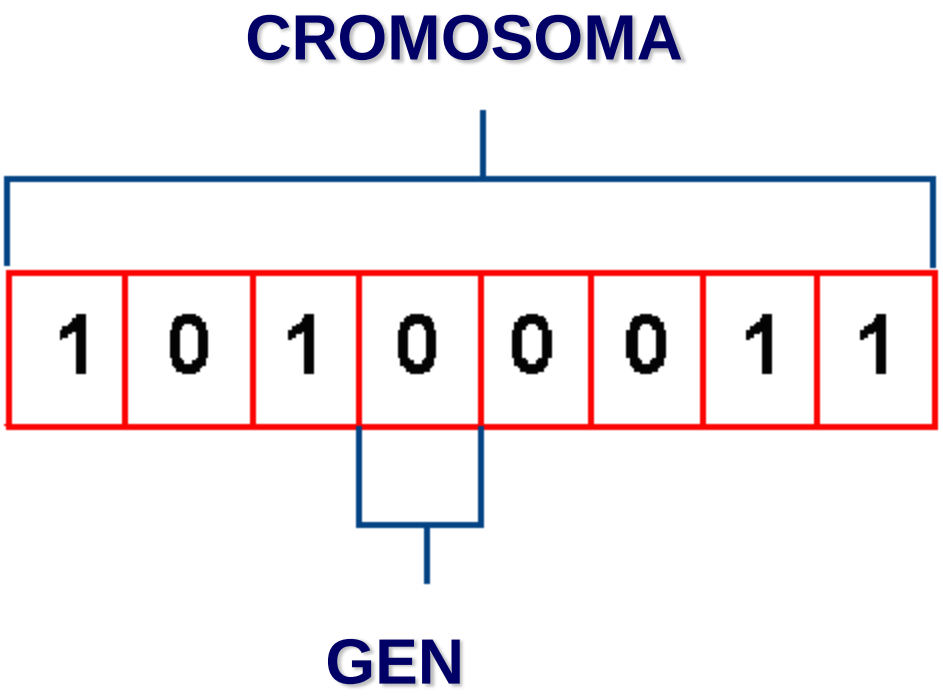
Debemos disponer de un mecanismo para codificar un individuo como un genotipo.

Existen muchas maneras de hacer esto y se ha de elegir la más relevante para el problema en cuestión.

Una vez elegida una representación, hemos de tener en mente como los genotipos (codificación) serán evaluados y qué operadores genéticos hay que utilizar.

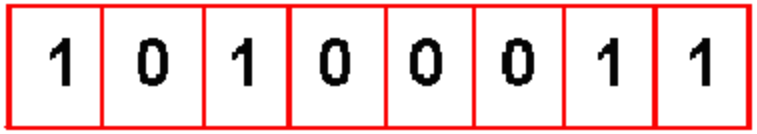
Ejemplo: Representación binaria

- La representación de un individuo se puede hacer mediante una codificación discreta, y en particular binaria.



Ejemplo: Representación binaria

8 bits Genotipo



Fenotipo

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

Ejemplo: Representación Real

- Una forma natural de codificar una solución es utilizando valores reales como genes

- Muchas aplicaciones tienen esta forma natural de codificación

Ejemplo: Representación Real

- Los individuos se representan como vectores de valores reales:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x_i \in R$$

- La función de evaluación asocia a un vector un valor real de evaluación:

$$f : R^n \rightarrow R$$

Ejemplo: Representación de orden

- Los individuos se representan como permutaciones.

7	3	6	8	2	4	1	5

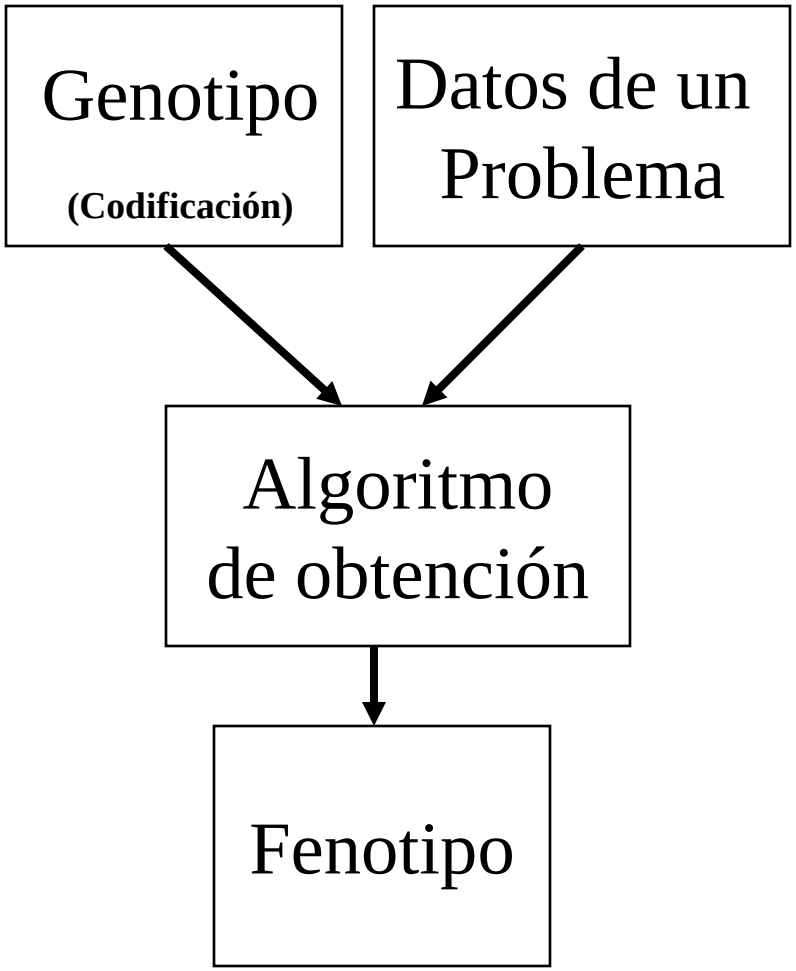
- Se utilizan para problemas de secuenciación.
- Ejemplo famoso: Viajante de Comercio, donde cada ciudad tiene asignado un único número entre 1 y n.
- Necesita operadores especiales para garantizar que el resultado de aplicar un operador sigue siendo una permutación.

Inicialización

- Uniforme sobre el espacio de búsqueda … (si es posible)
  - Cadena binaria: 0 ó 1 con probabilidad 0.5
  - Representación real: uniforme sobre un intervalo dado (para valores acotados)
- Elegir la población a partir de los resultados de una heurística previa.

Correspondencia entre Genotipo y Fenotipo

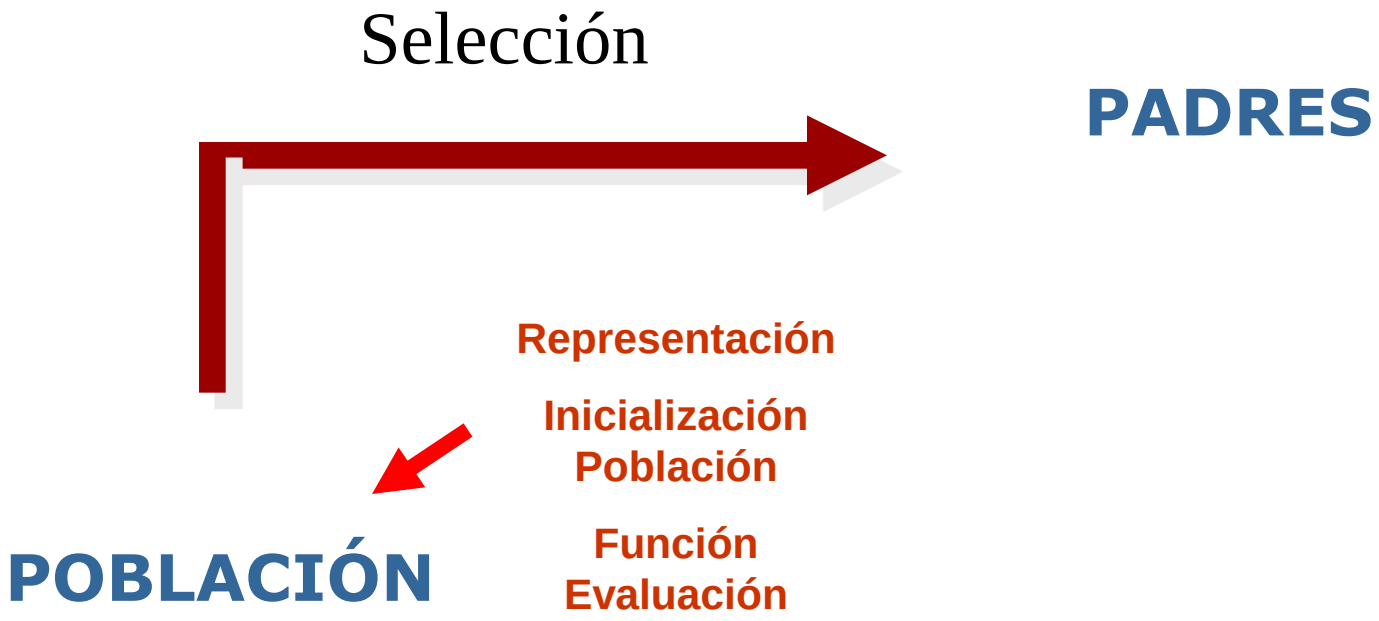
- Algunas veces la obtención del fenotipo a partir del genotipo es un proceso obvio.
- En otras ocasiones el genotipo puede ser un conjunto de parámetros para algún algoritmo, el cual trabaja sobre los datos de un problema para obtener un fenotipo



Evaluación de un individuo

- Este es el paso más costoso para una aplicación real
- Puede ser una subrutina, un simulador, o cualquier proceso externo (ej. Experimentos en un robot, ...)
- Se pueden utilizar funciones aproximadas para reducir el costo de evaluación.
- Cuando hay restricciones, éstas se pueden introducir en el costo como penalización.
- Con múltiples objetivos se busca una solución de compromiso.

¿CÓMO SE CONSTRUYE UN AG?



Estrategia de Selección

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

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

Esta idea nos define la **presión selectiva** que determina en qué grado la reproducción está dirigida por los mejores individuos.

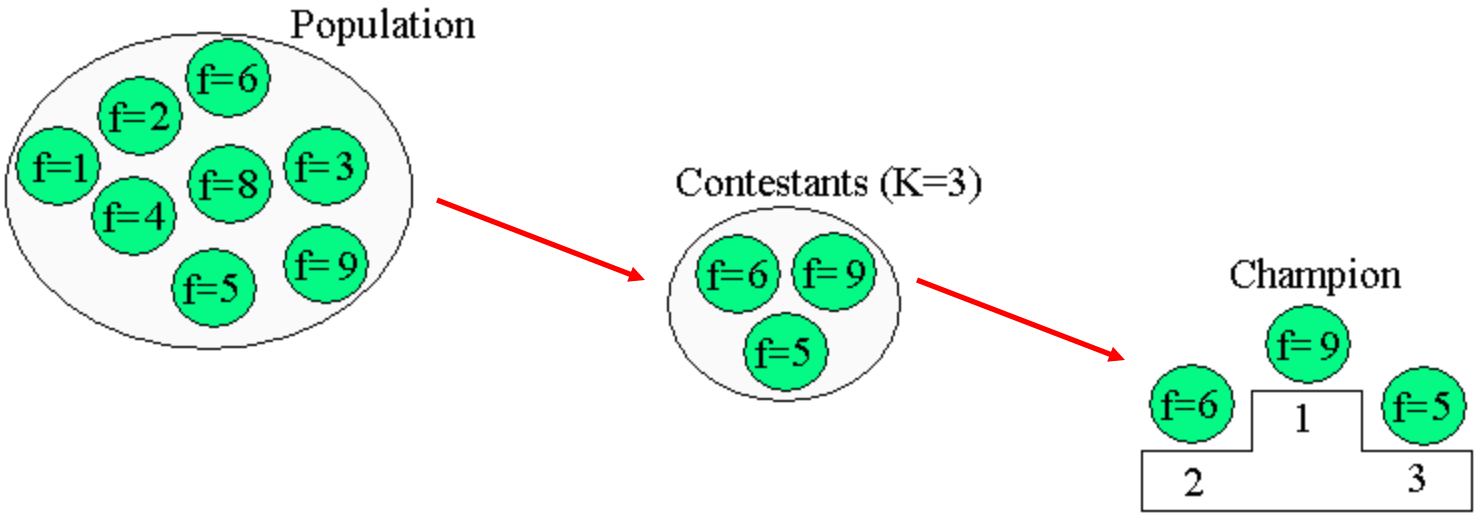
Estrategia de Selección

Selección por torneo

Para cada padre a seleccionar:

- Escoger aleatoriamente k individuos, con reemplazamiento
- Seleccionar el mejor de ellos

k se denomina tamaño del torneo. A mayor k, mayor presión selectiva y viceversa.

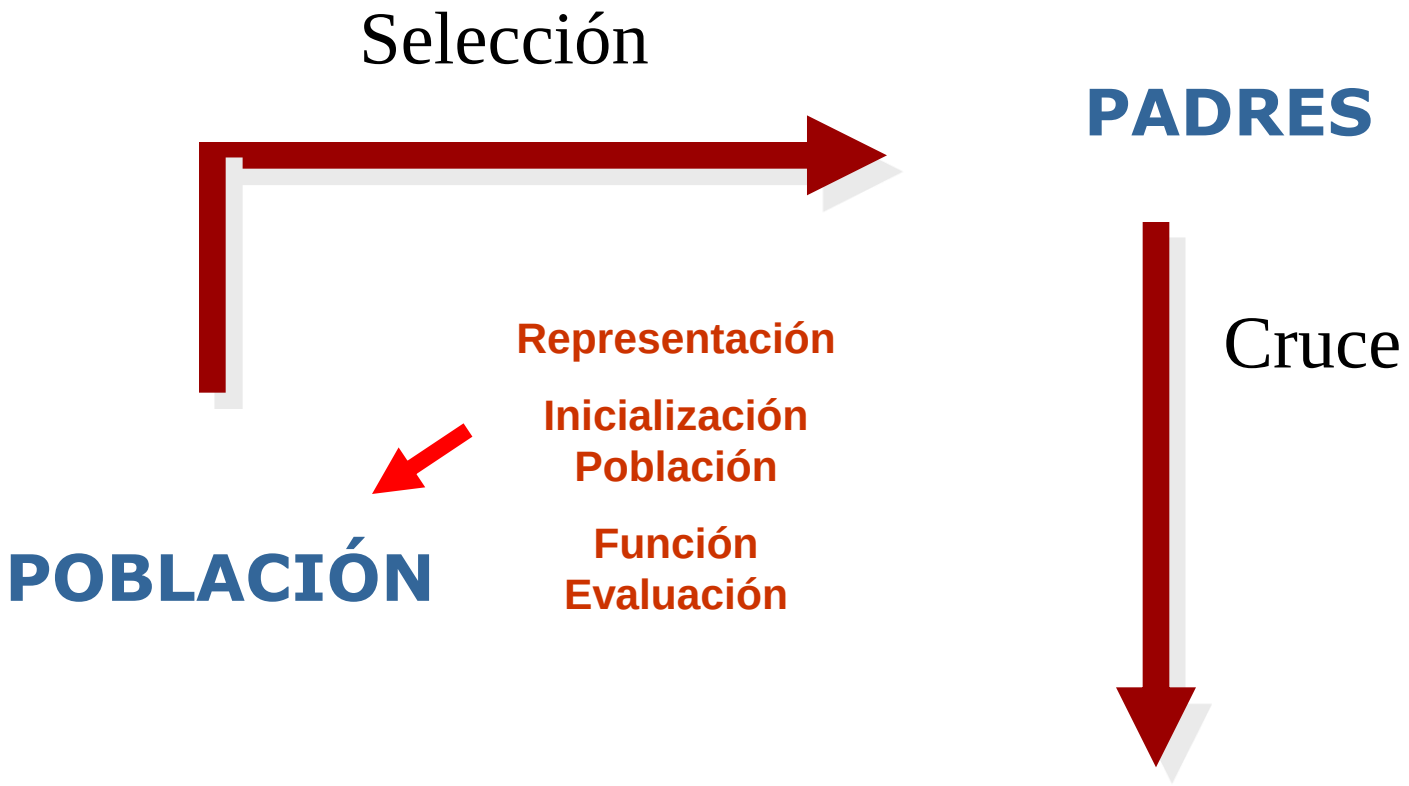


Estrategia de Selección

Algunos esquemas de selección

- Selección por Torneo (TS): Escoge al individuo de mejor fitness de entre Nts individuos seleccionados aleatoriamente (Nts=2,3,…).
- Orden Lineal (LR): La población se ordena en función de su fitness y se asocia una probabilidad de selección a cada individuo que depende de su orden.
- Selección Aleatoria (RS).
- Emparejamiento Variado Inverso (NAM): Un padre lo escoge aleatoriamente, para el otro selecciona Nnam padres y escoge el más lejano al primer (Nnam=3,5, ….). Está orientado a generar diversidad.
- Selección por Ruleta: Se asigna una probabilidad de selección proporcional al valor del fitness del cromosoma. (Modelo clásico)

¿CÓMO SE CONSTRUYE UN AG?



Operador de Cruce

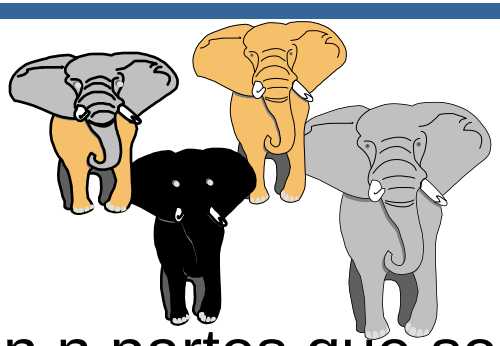
Podríamos tener uno o más operadores de cruce para nuestra representación.

Algunos aspectos importantes a tener en cuenta son:

- Los hijos deberían heredar algunas características de **cada** padre. Si éste no es el caso, entonces estamos ante un operador de mutación.
- Se debe diseñar de acuerdo a la representación.
- La recombinación debe producir cromosomas válidos.
- Se utiliza con una probabilidad alta de actuación sobre cada pareja de padres a cruzar (Pc entre 0.6 y 0.9), si no actua los padres son los descendientes del proceso de recombinación de la pareja.

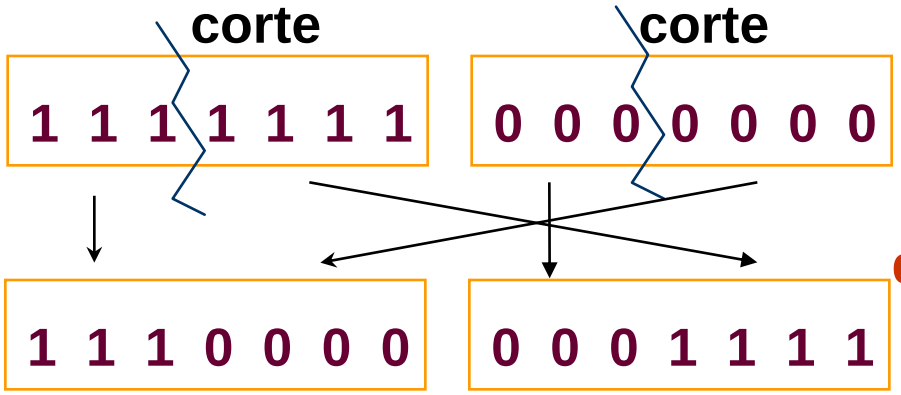
Ejemplo: Operador de cruce para representación binaria

Población:

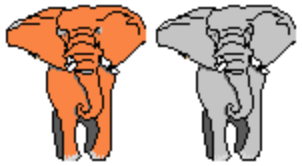


. . .

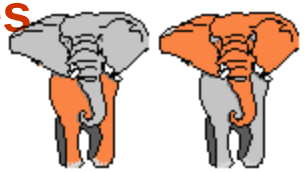
Cada cromosoma se corta en n partes que son recombinadas. (Ejemplo para n = 1).



padres

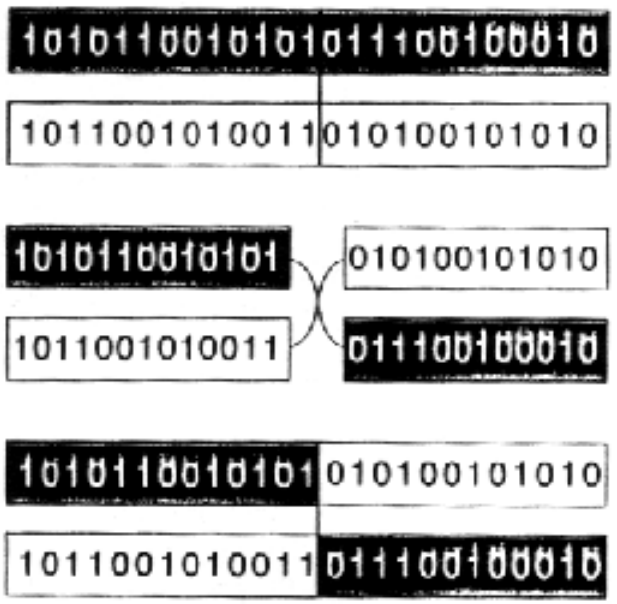


descendientes

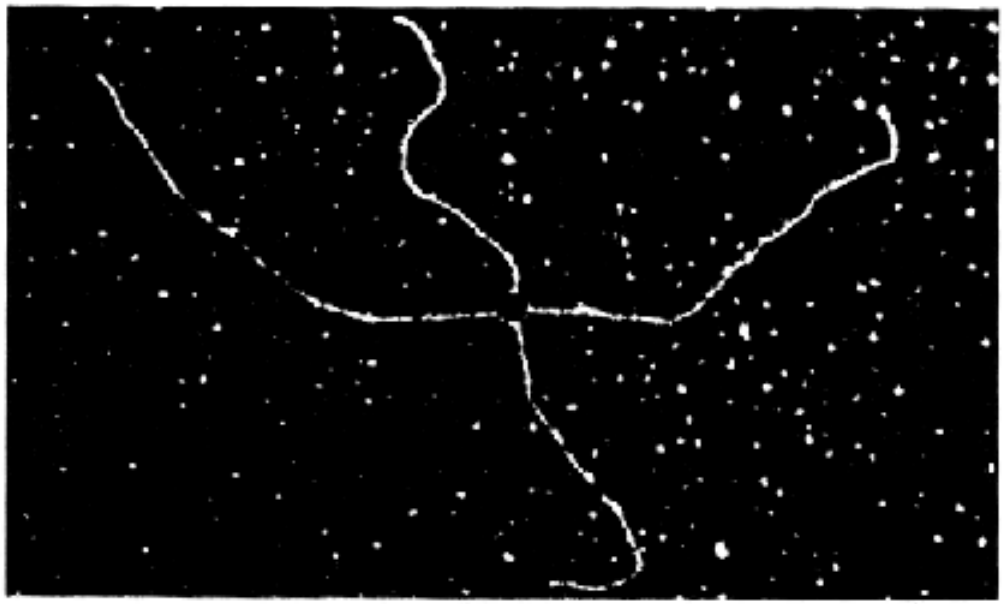


Operador de Cruce

Imagen clásica (John Holland) que introduce el operador de cruce

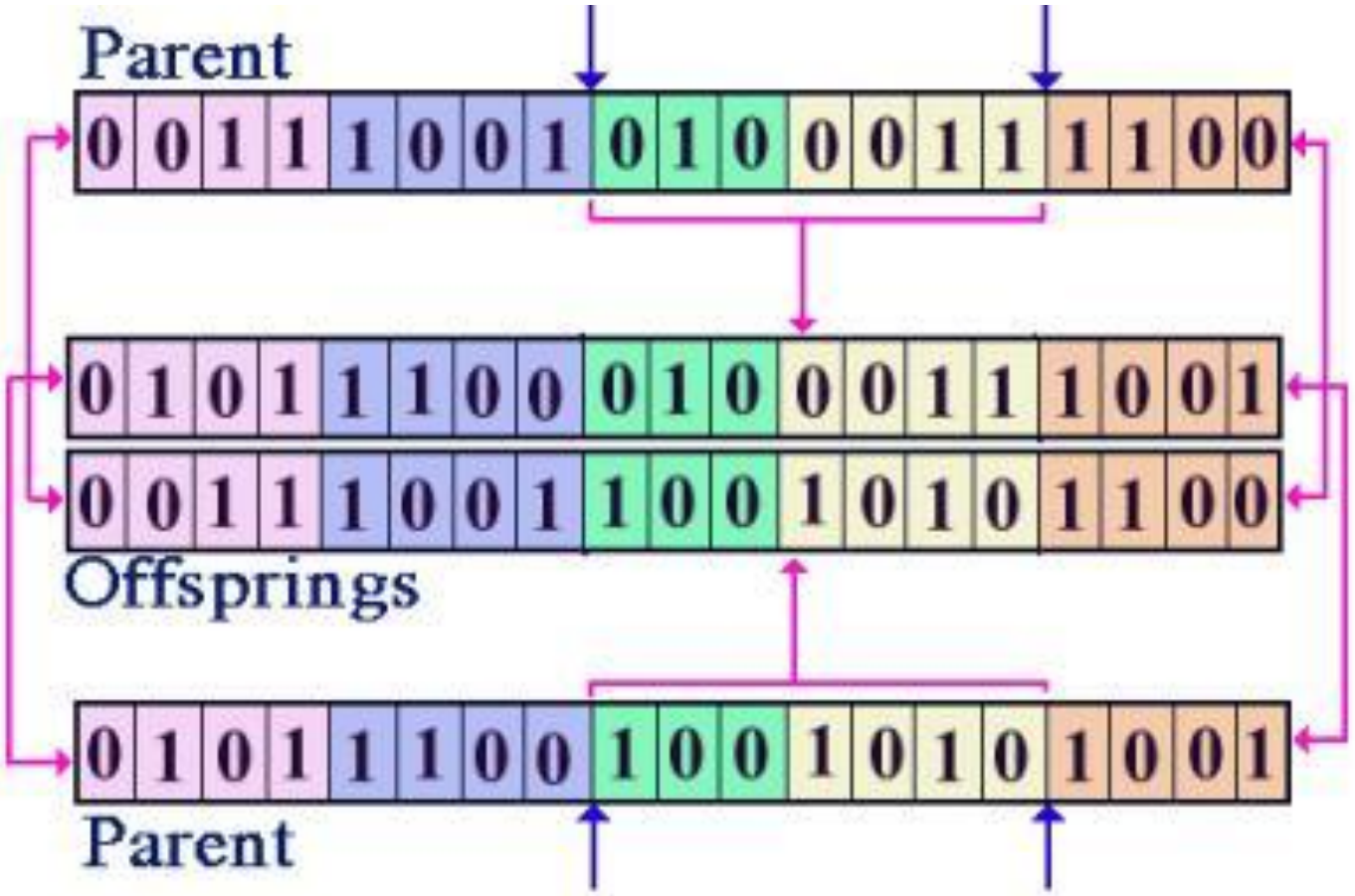


CROSSOVER is the fundamental mechanism of genetic rearrangement for both real organisms and genetic algorithms.



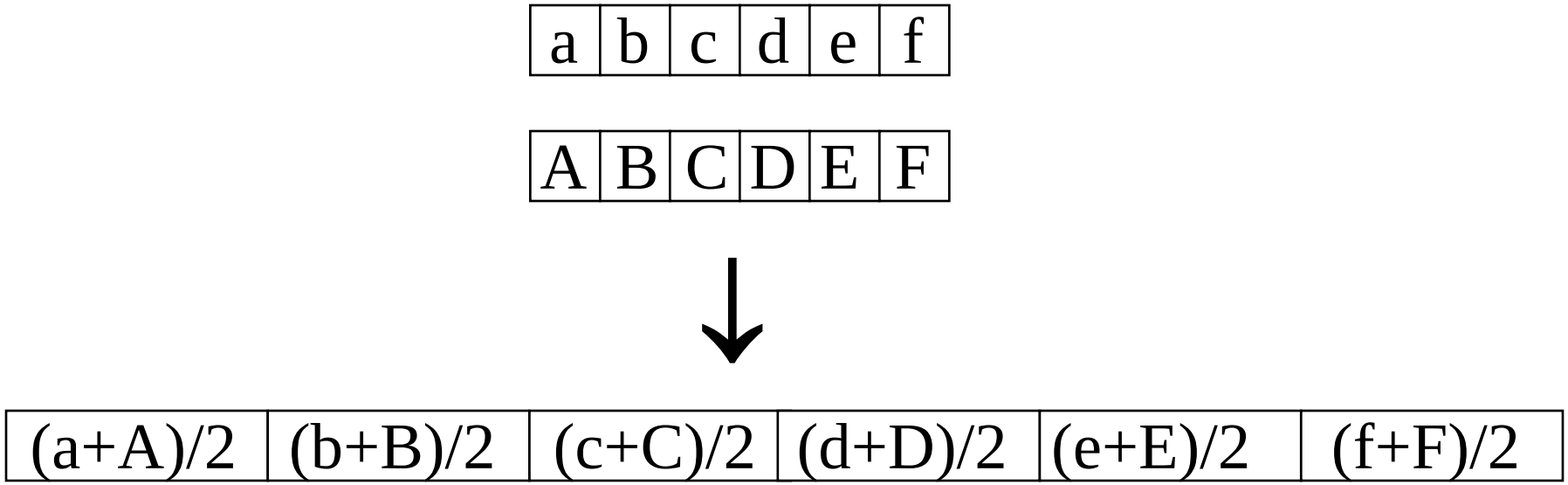
Chromosomes line up and then swap the portions of their genetic code beyond the crossover point.

Ejemplo: Operador de cruce en dos puntos para representación binaria



Ejemplo: Operador de cruce para representación real

Recombinación aritmética (cruce aritmético):



Existe muchos operadores específicos para la codificación real.

Ejemplo: Operador de cruce para representación real: **BLX-α**

- Dados 2 cromosomas

$$C_1 = (c_{11},\ldots, c_{1n}) \text{ y } C_2 = (c_{21},\ldots, c_{2n}) ,$$

- BLX- α genera dos descendientes

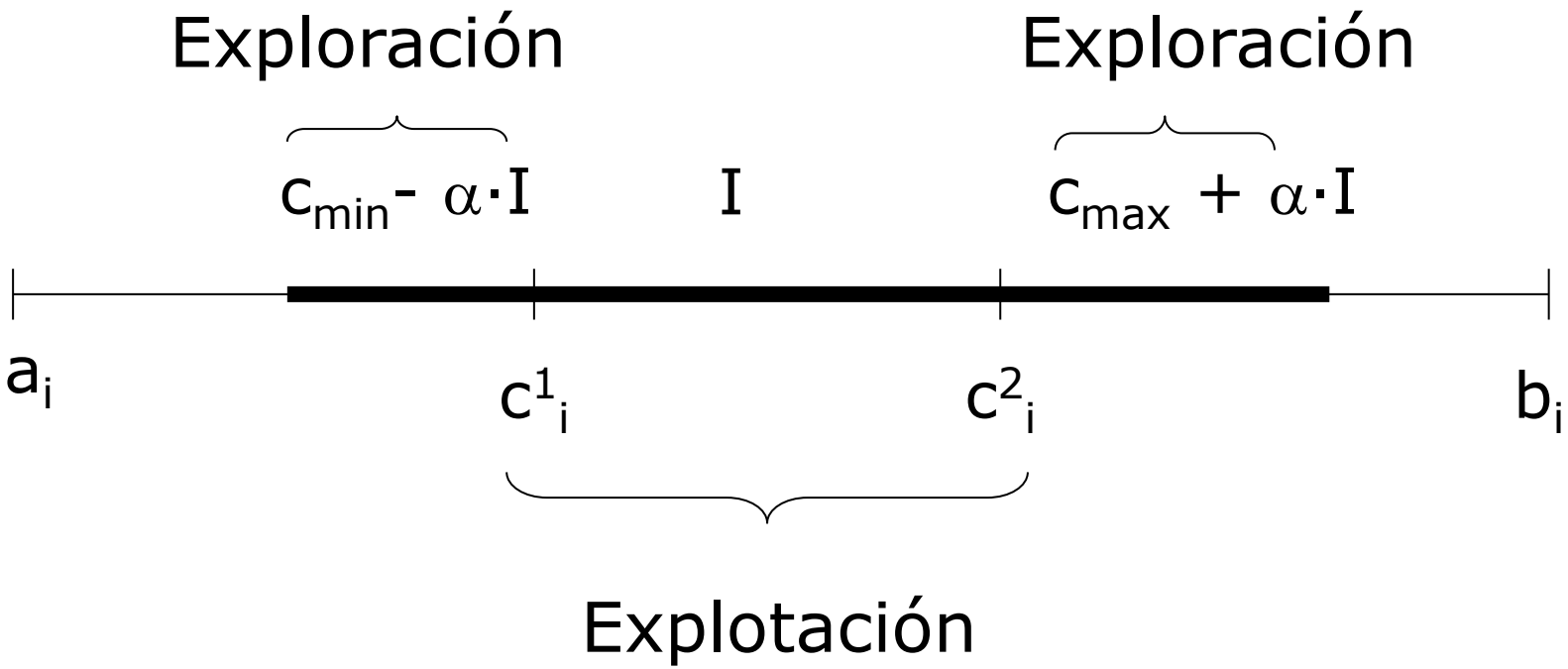
$$H_k = (h_{k1},\ldots, h_{ki},\ldots, h_{kn}) , k = 1,2$$

- donde h_{ki} se genera aleatoriamente en el intervalo:

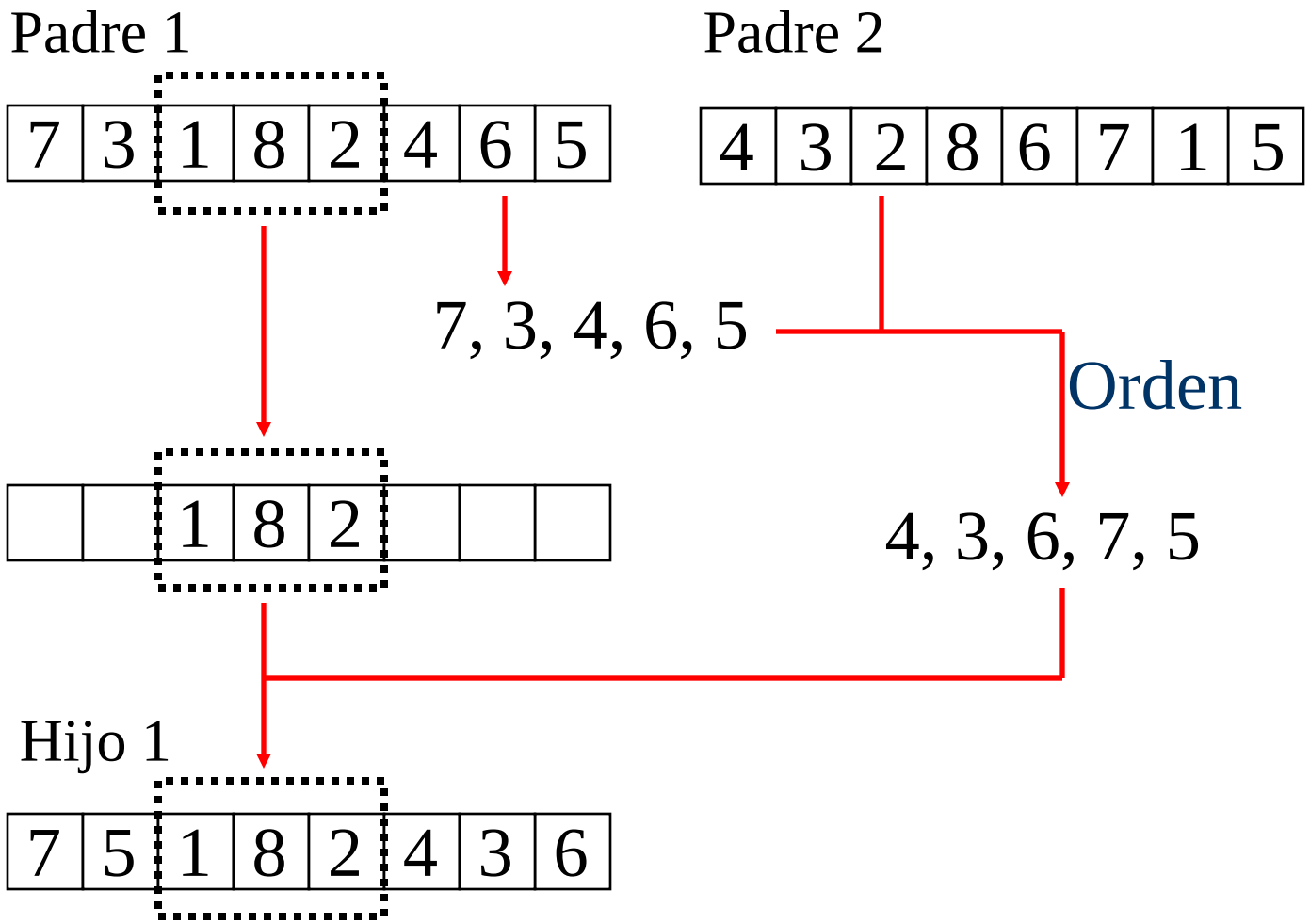
$$[C_{min} - I\cdot\alpha, C_{max} + I\cdot\alpha]$$

  - $C_{max} = \max \{c_{1i} , c_{2i}\}$
  - $C_{min} = \min \{c_{1i} , c_{2i}\}$
  - $I = C_{max} - C_{min} , \alpha \in [0,1]$

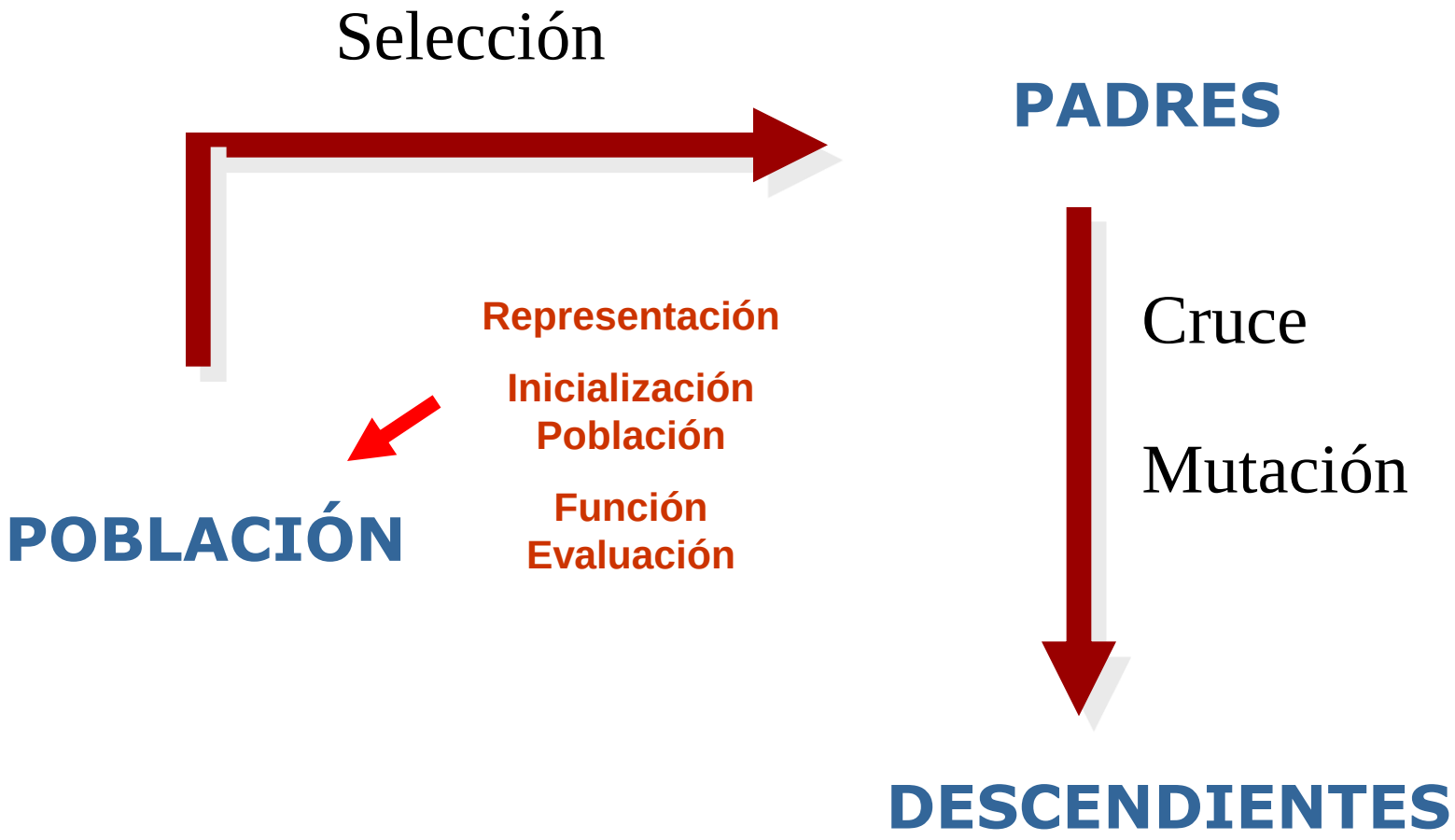
Ejemplo: Operador de cruce para representación real: **BLX-α**



Ejemplo: Operador de cruce para representación de orden



¿CÓMO SE CONSTRUYE UN AG?



Operador de mutación

Podemos tener uno o más operadores de mutación para nuestra representación.

Algunos aspectos importantes a tener en cuenta son:

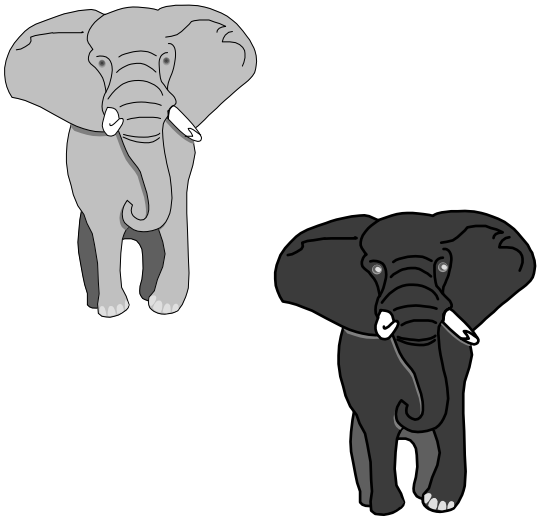
- Debe permitir alcanzar cualquier parte del espacio de búsqueda.
- El tamaño de la mutación debe ser controlado.
- Debe producir cromosomas válidos.
- Se aplica con una probabilidad muy baja de actuación sobre cada descendiente obtenido tras aplicar el operador de cruce (incluidos los descendientes que coinciden con los padres porque el operador de cruce no actúa).

Ejemplo: Mutación para representación discreta binaria

antes 1 1 1 1 1 1 1

después 1 1 1 0 1 1 1

gen mutado



La mutación ocurre con una probabiliad p_m para cada gen

Ejemplo: Mutación para representación real

Perturbación de los valores mediante un valor aleatorio.

Frecuentemente, mediante una distribución Gaussiana/normal $N(0,\sigma)$, donde

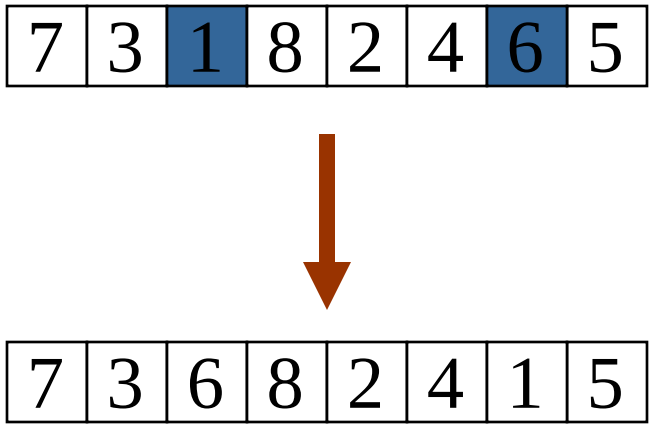
- 0 es la media
- σ es la desviación típica

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

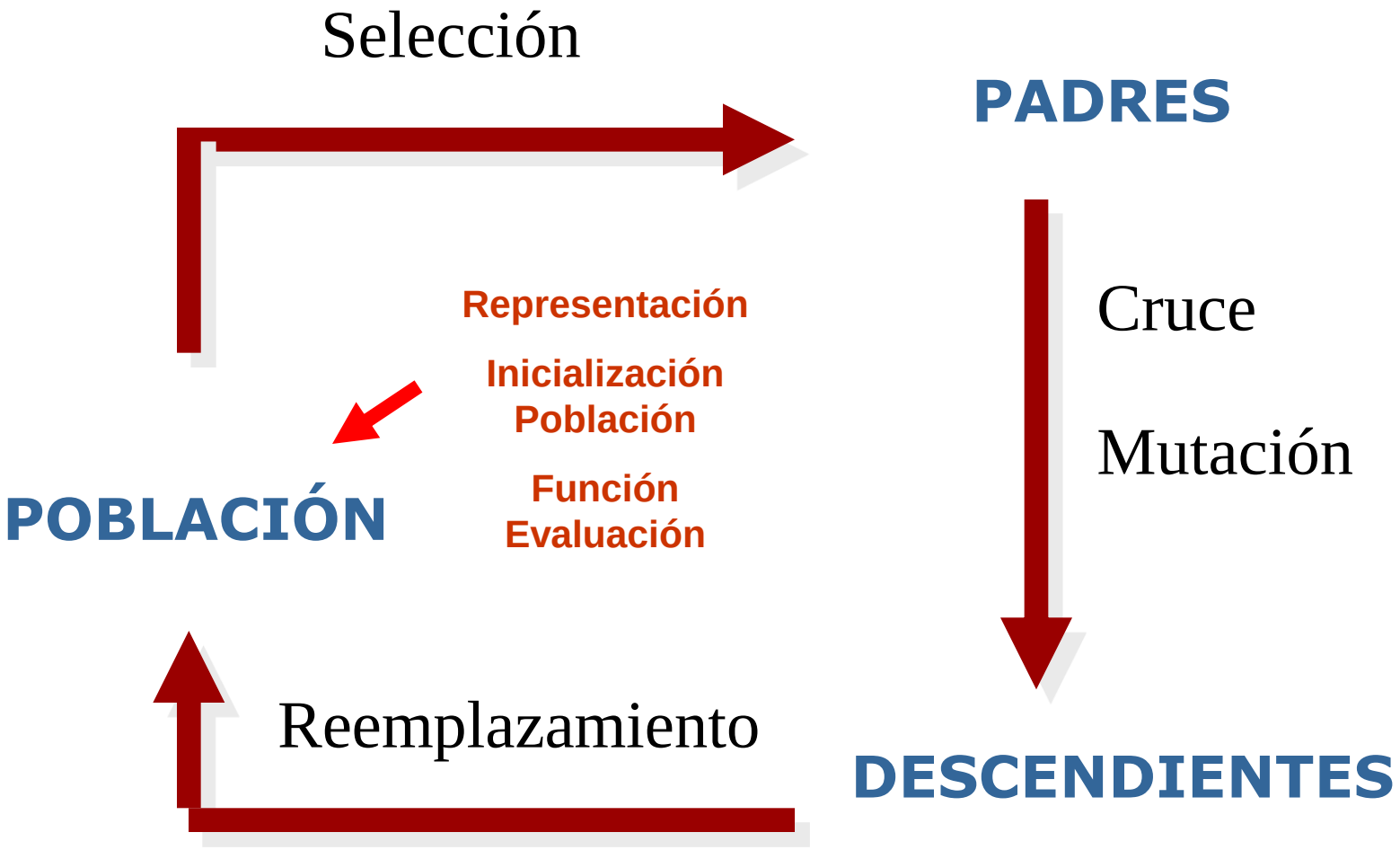
para cada parámetro.

Ejemplo: Mutación para representación de orden

Seleción aleatoria de dos genes e intercambio de ambos.



¿CÓMO SE CONSTRUYE UN AG?



Estrategia de Reemplazamiento

La presión selectiva se ve también afectada por la forma en que los cromosomas de la población son reemplazados por los nuevos descendientes.

Podemos utilizar métodos de reemplazamiento aleatorios, o determinísticos.

Podemos decidir no reemplazar al mejor cromosoma de la población: **Elitismo**

(el uso del Elitismo es aconsejado en los modelos generacionales para no perder la mejor solución encontrada).

Un modelo con alto grado de elitismo consiste en utilizar una población intermedia con todos los padres (N) y todos los descendientes y seleccionar los N mejores. Esto se combina con otras componentes con alto grado de diversidad. Será objetivo de estudio del Tema 7.

Estrategia de Reemplazamiento

Algunas estrategias de reemplazo para AG estacionarios

Cuando se considera un modelo estacionario (en el que se reemplazan solo uno o dos padres, frente al modelo generacional en el que se reemplaza la población completa), nos encontramos con diferentes propuestas.

A continuación presentamos algunas posibilidades:

- Reemplazar al peor de la población (RW). Genera alta presión selectiva.
- Torneo Restringido (RTS): Se reemplaza al mas parecido de entre *w (w=3, …)*. Mantiene una cierta diversidad.
- Peor entre semejantes (WAMS): Se reemplaza el peor cromosoma del conjunto de los *w (w=3, …)* padres más parecidos al descendiente generado (seleccionados de toda la población). Busca equilibrio entre diversidad y presión selectiva.
- Algoritmo de Crowding Determinístico (DC): El hijo reemplaza a su padre más parecido. Mantiene diversidad.

Estudio comparativo de algunos modelos estacionarios

- Para cada combinación selección-reemplazo se ha definido un AG.

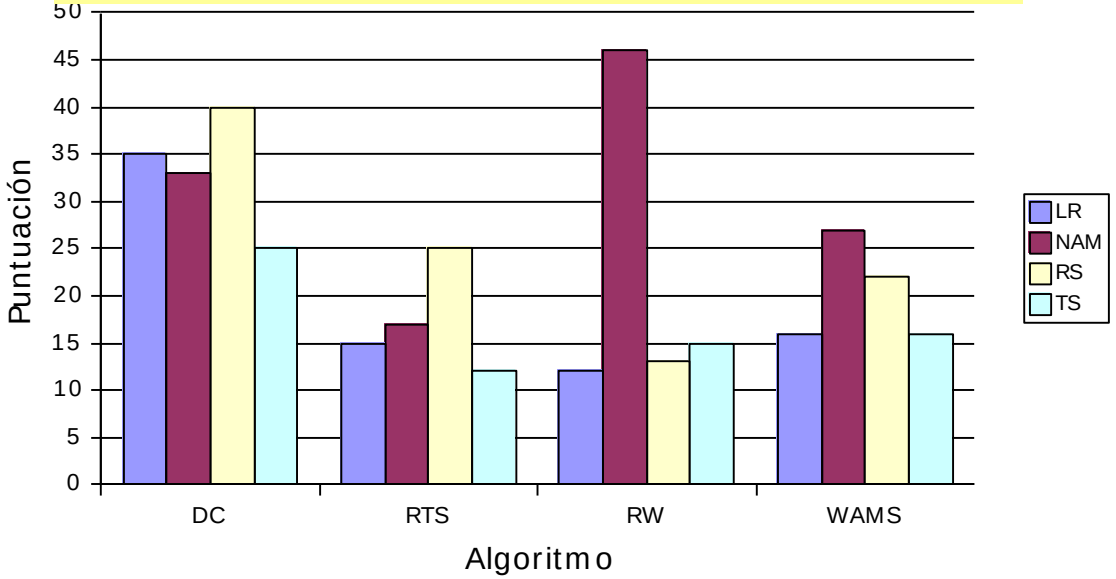
- Las características comunes de todos ellos son:
  - Cruce aplicado es BLX-0.5.
  - Mutación no uniforme aplicado con pMut = 1/8 (ind.)
  - Tamaño de la población de 60 individuos.
  - 100000 evaluaciones por ejecución

Estudio comparativo de algunos modelos estacionarios

- Se evalúan sobre un conjunto de 13 problemas (2 reales y 11 funciones clásicas de distinta dificultad).
- Para cada AG se muestra una puntuación obtenida según el criterio:
  - Para cada función se ordenan los AGs en función de la media para 50 ejecuciones.
  - Se aplica el *t-test de Student con p=0.05.*
  - *Para cada función se asigna a cada algoritmo una puntuación. 5 al mejor, 4 al siguiente, ... Los equivalentes entre sí (mediante t-student) reciben igual puntuación.*
  - *Se suman los resultados para las 13 funciones.*

Estudio comparativo de algunos modelos estacionarios

Resultados de las 16 combinaciones



Selección por Torneo (TS)
Orden Lineal (LR)
Selección Aleatoria (RS).
Emparejamiento Variado Inverso (NAM)

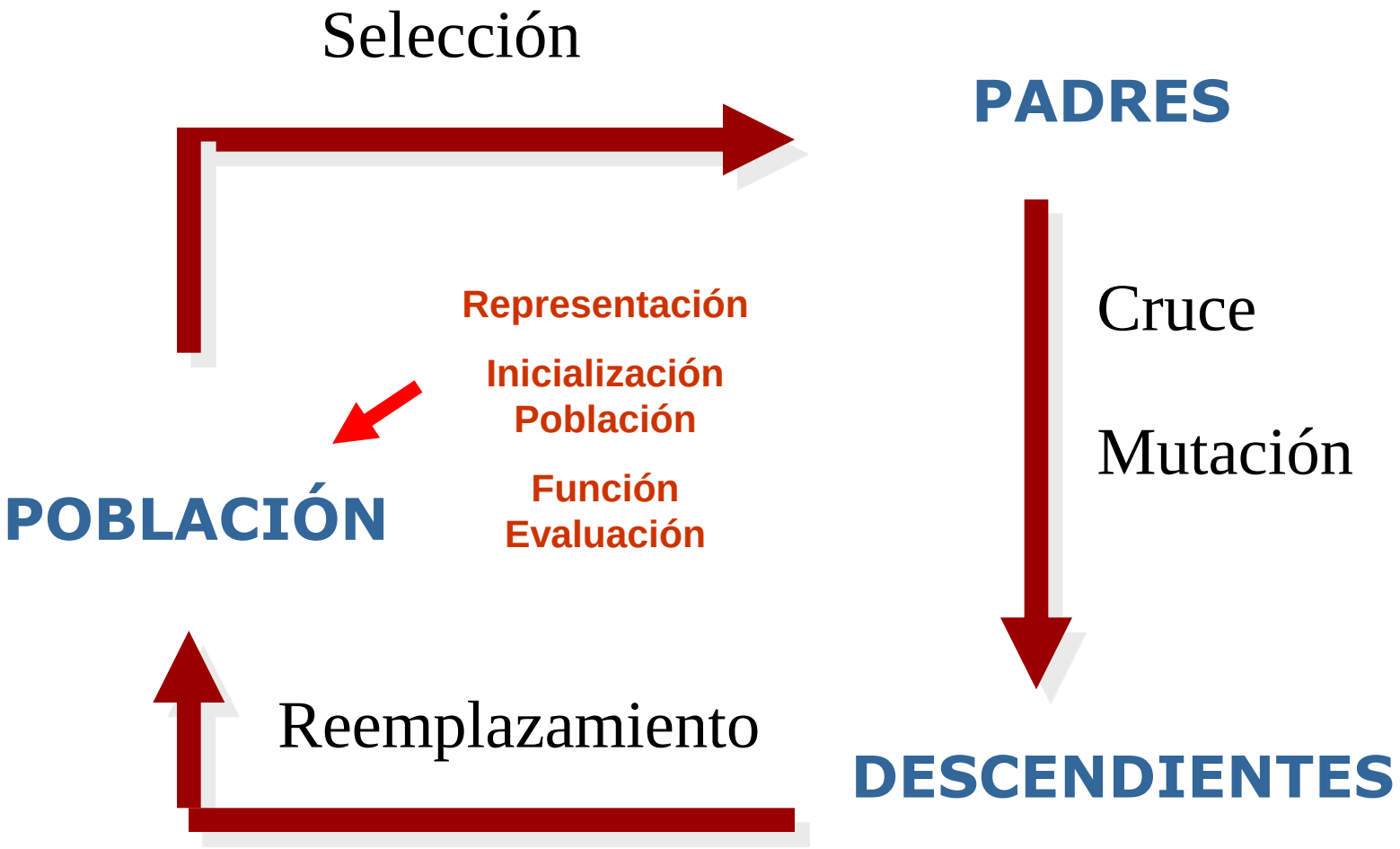
Reemplazar al peor de la población (RW)
Torneo Restringido (RTS)
Peor entre semejantes (WAMS)
Algoritmo de Crowding Determinístico (DC)

Criterio de Parada

- Cuando se alcanza el óptimo!
- Recursos limitados de CPU:

  Fijar el máximo número de evaluaciones
- Límite sobre la paciencia del usuario: Después de algunas iteraciones sin mejora.

¿CÓMO SE CONSTRUYE UN AG?
RESUMEN



PROCESO ITERATIVO + CRITERIO DE PARADA

4. SOBRE SU UTILIZACIÓN

- Nunca se deben sacar conclusiones de una única ejecución
  - **utilizar medidas estadísticas (medias, medianas, …)**
  - **con un número suficiente de ejecuciones independientes**
- No se debe ajustar/chequear la actuación de un algoritmo sobre ejemplos simples si se desea trabajar con casos reales.
- Existe una comentario genérico en el uso de los Algoritmos no determinísticos:

  "Se puede obtener lo que se desea en una experimentación de acuerdo a la dificultad de los casos utilizados"

  (se encuentran propuestas en las que basta encontrar un caso adecuado para un algoritmo para afirmar que es muy bueno, pero esta afirmación no puede ser extensible a otros casos, es el error en el que incurren algunos autores)

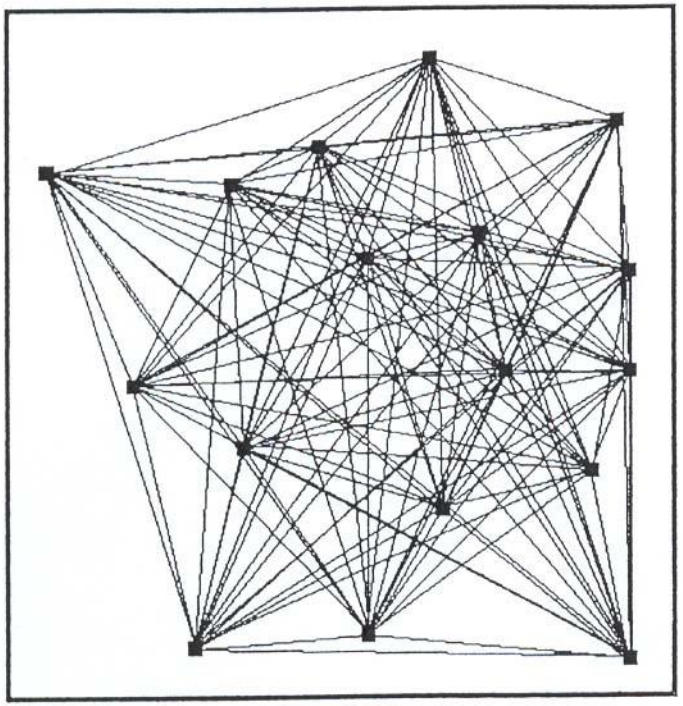
5. EJEMPLO: VIAJANTE DE COMERCIO

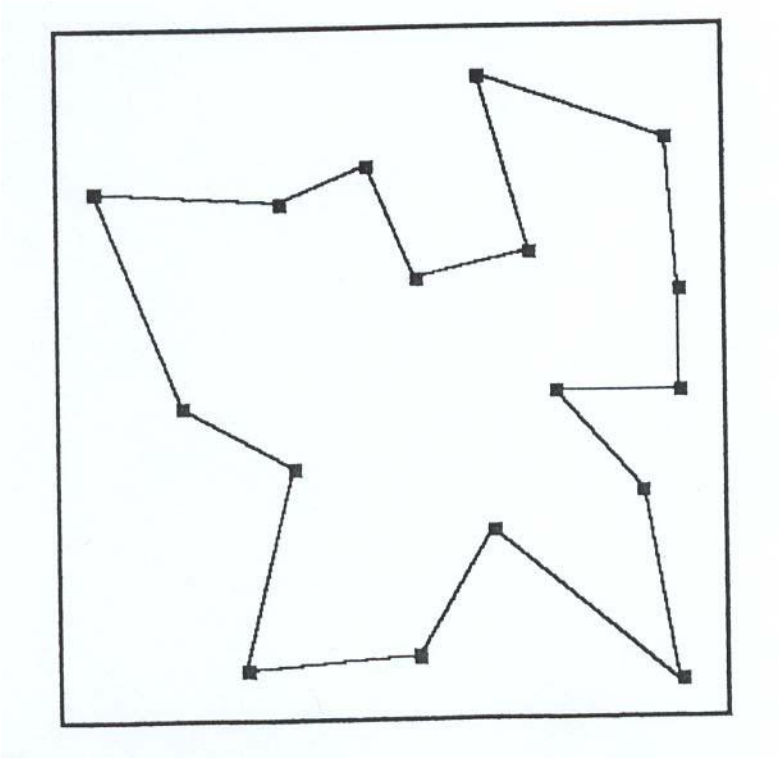
Representación de orden

(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)

Viajante de Comercio

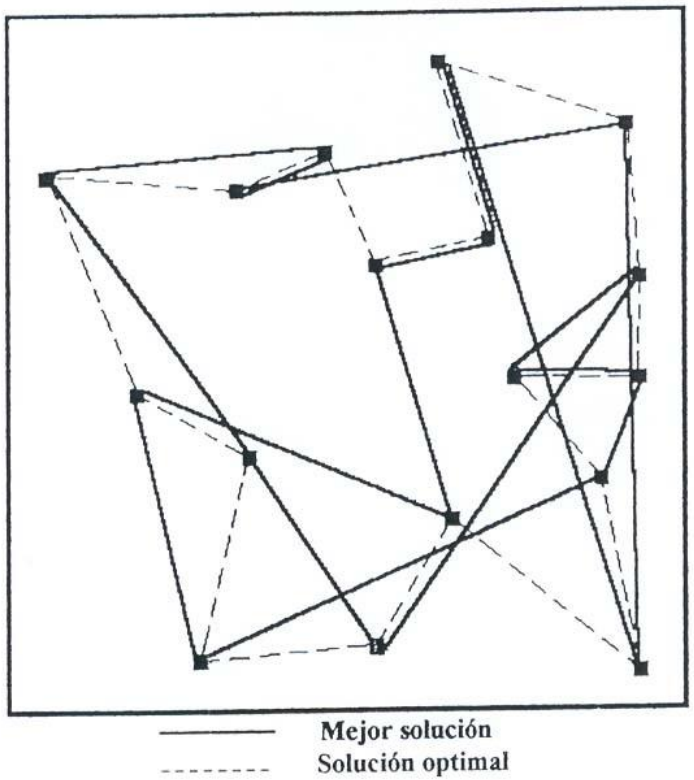




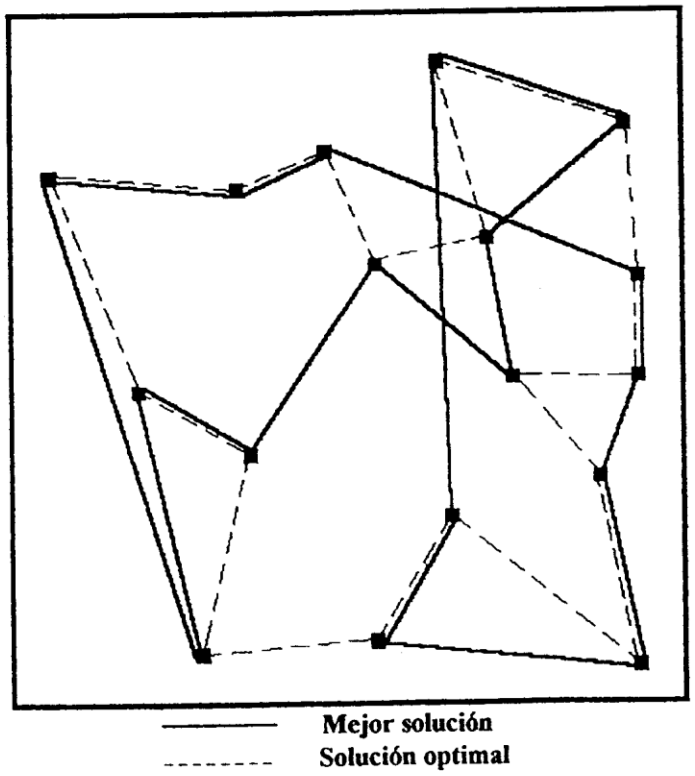
17! = 3.5568743 e14 recorridos posibles

Solución óptima: 226.64

Viajante de Comercio



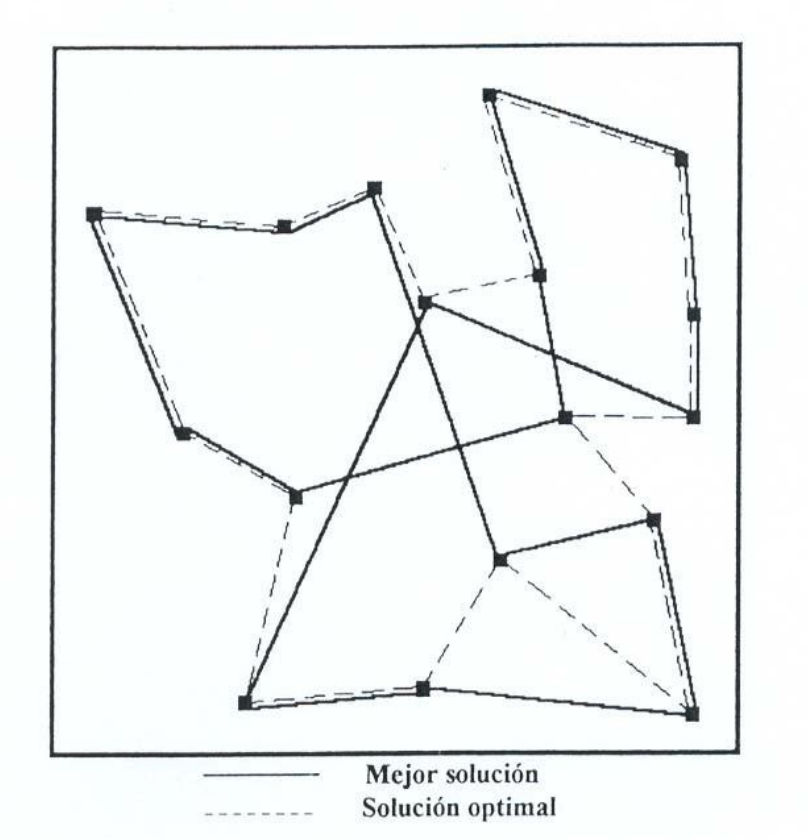
Iteración: 0 Costo: 403.7



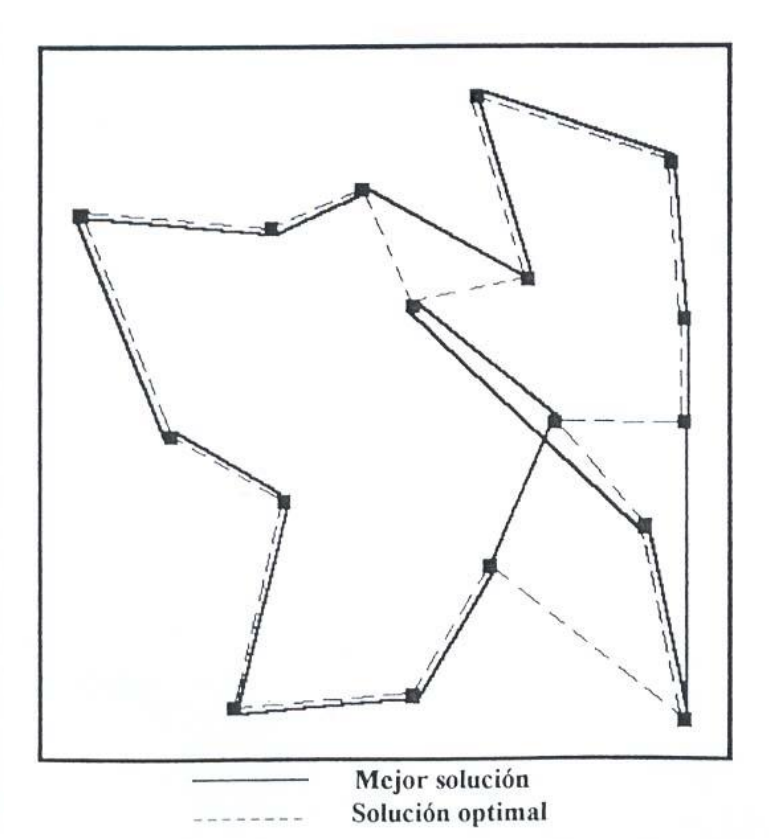
Iteración: 25 Costo: 303.86

Solución óptima: 226.64

Viajante de Comercio



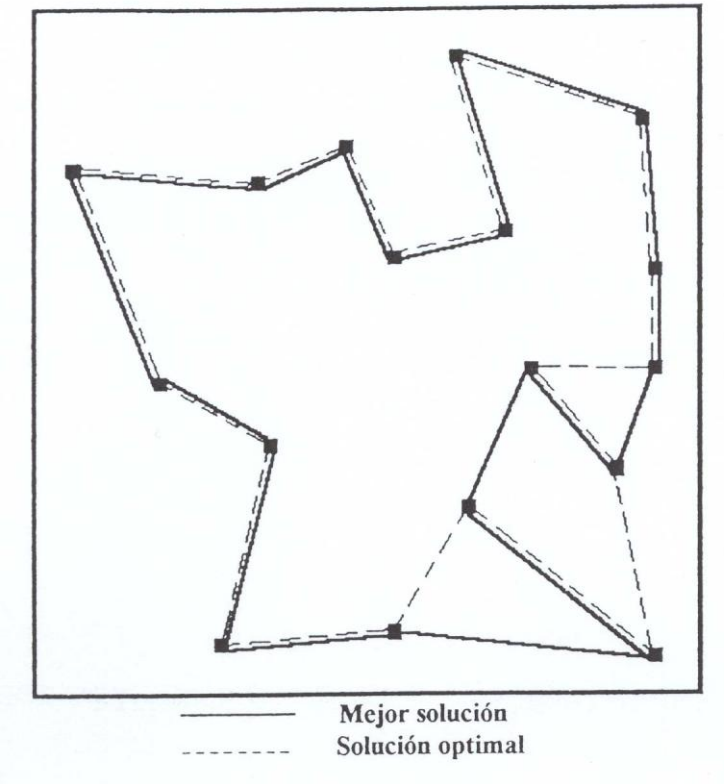
Iteración: 50 Costo: 293.6



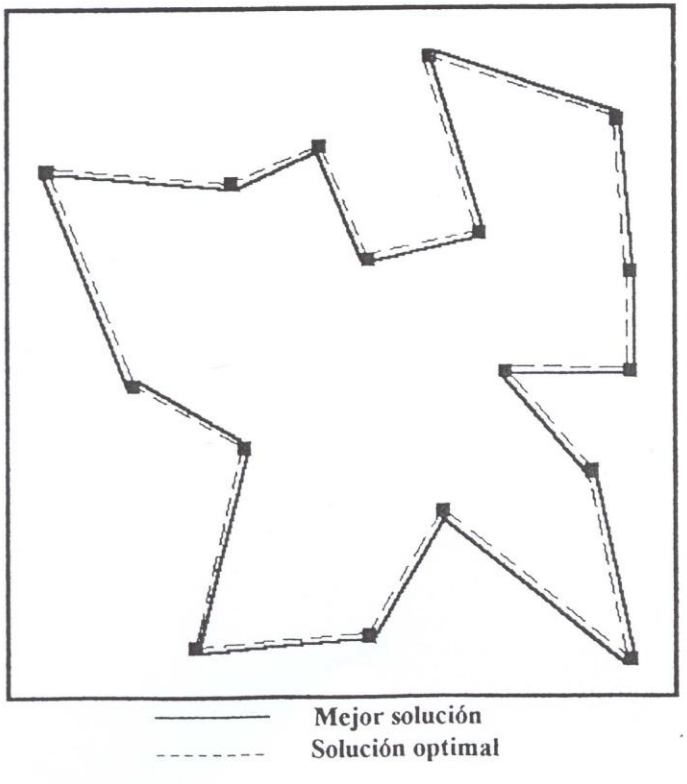
Iteración: 100 Costo: 256.55

Solución óptima: 226.64

Viajante de Comercio

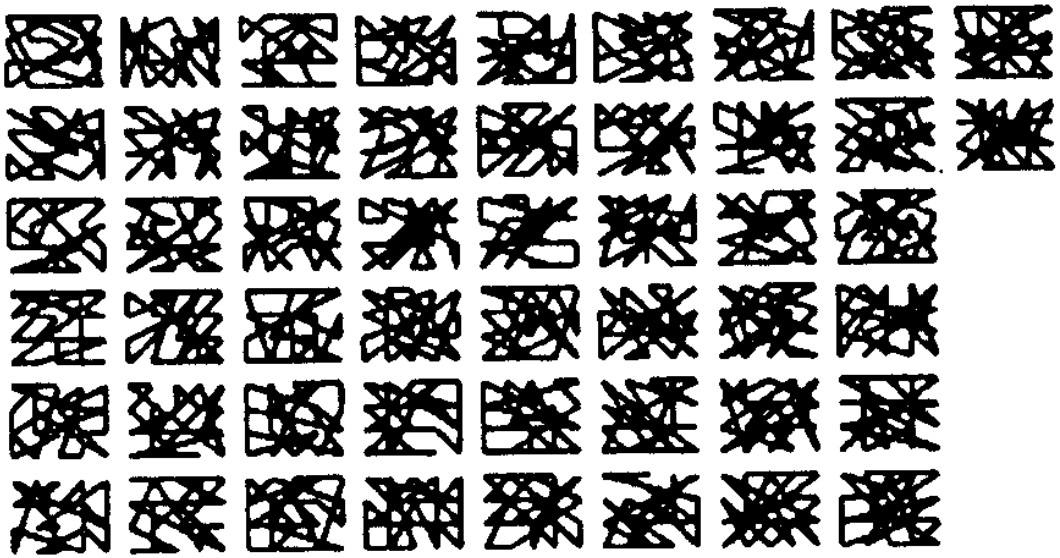


Iteración: 200 Costo: 231.4



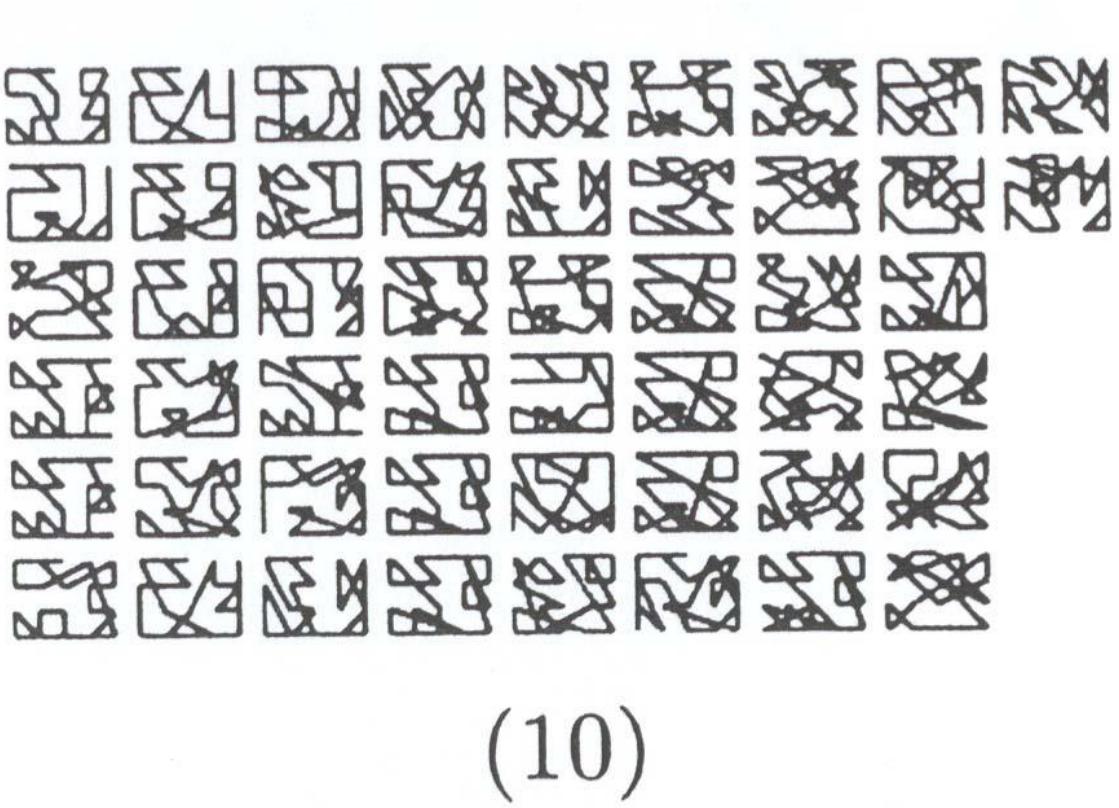
Iteración: 250 Solución óptima: 226.64

Viajante de Comercio



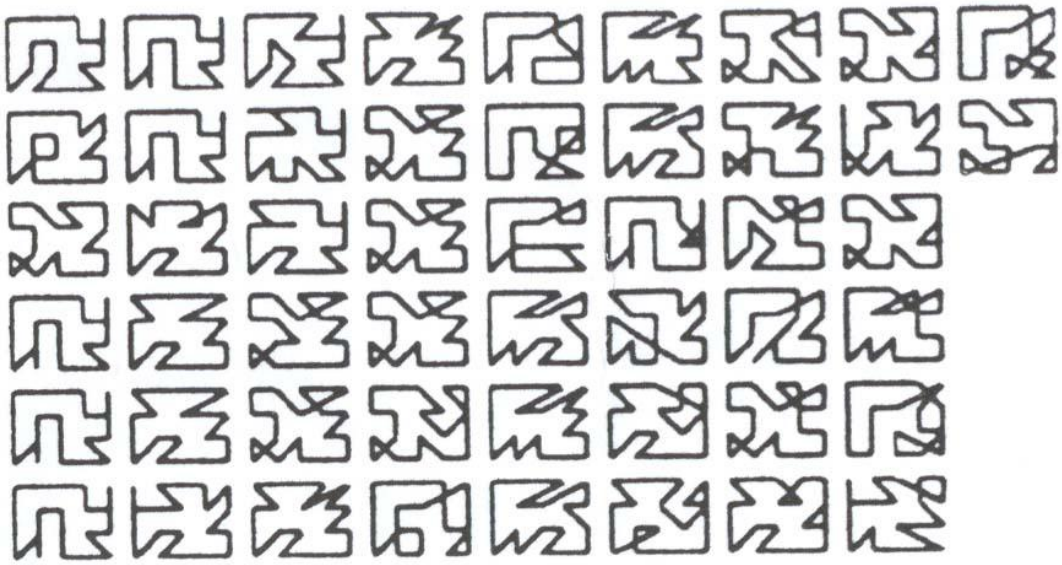
Visualización de la evolución de una población de 50 cromosomas y 70 iteraciones

Viajante de Comercio



Visualización de la evolución de una población de 50 cromosomas y 70 iteraciones

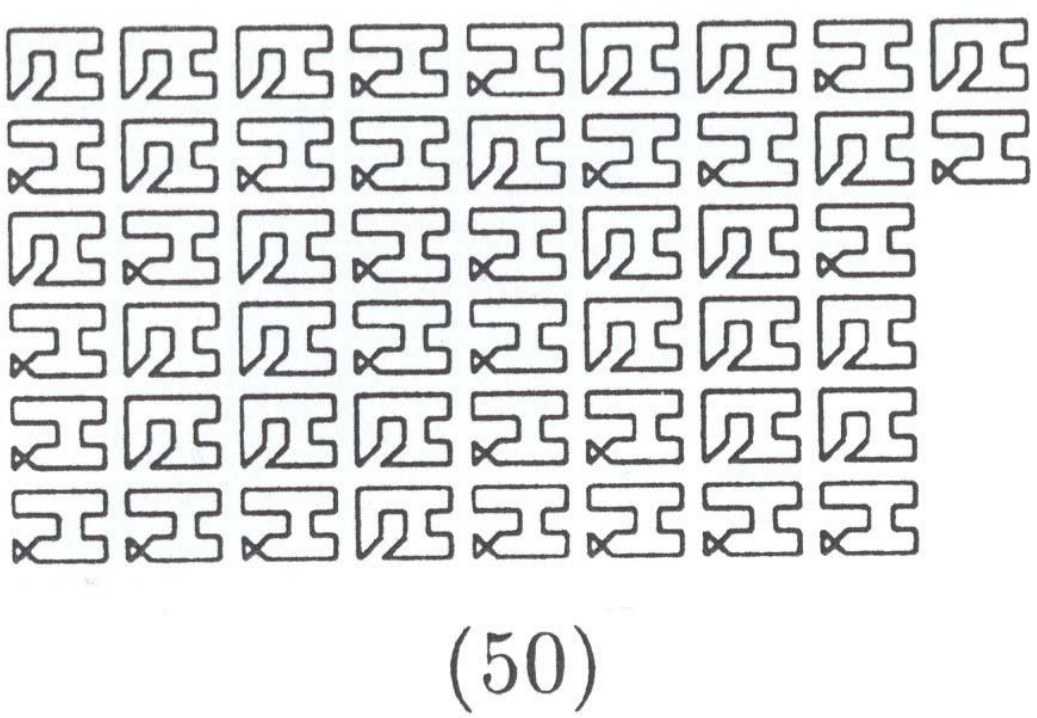
Viajante de Comercio



(30)

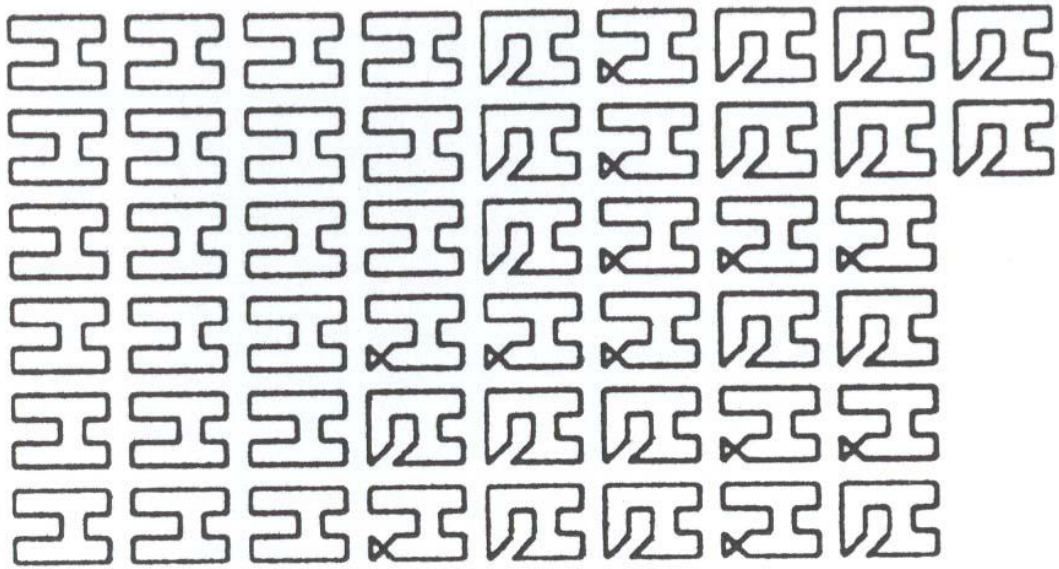
Visualización de la evolución de una población de 50 cromosomas y 70 iteraciones

Viajante de Comercio



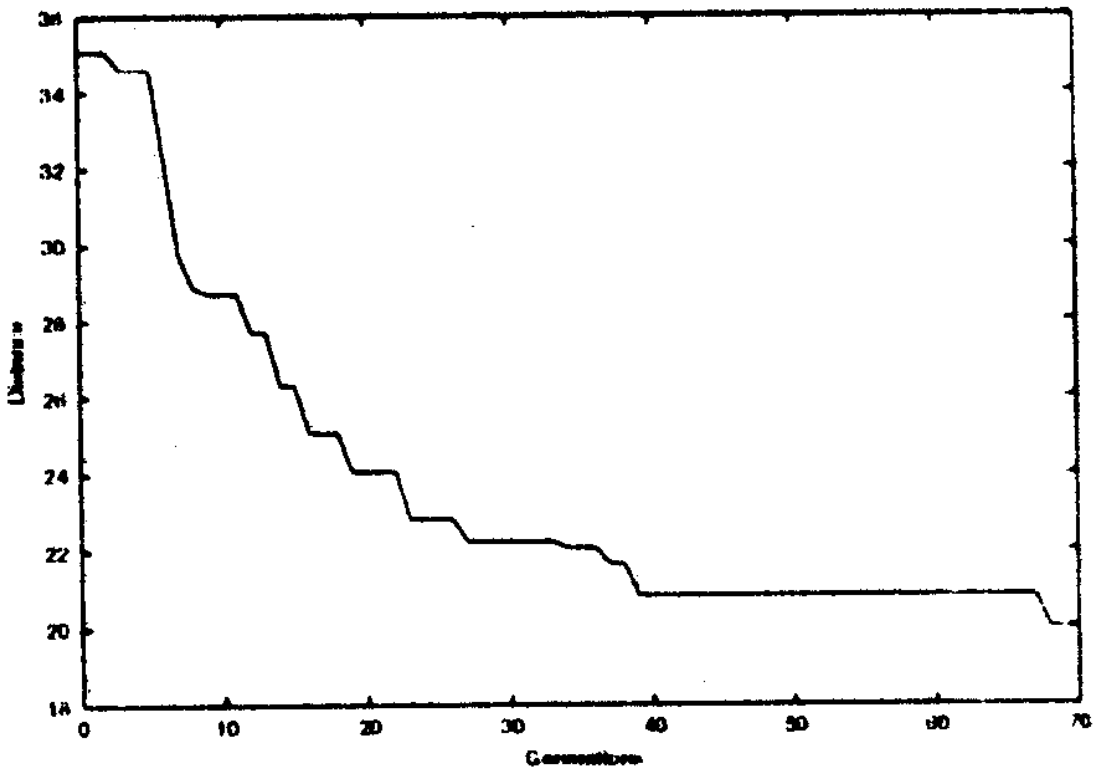
Visualización de la evolución de una población de 50 cromosomas y 70 iteraciones

Viajante de Comercio



Visualización de la evolución de una población de 50 cromosomas y 70 iteraciones

Viajante de Comercio



Visualización de la evolución de una población de 50 cromosomas y 70 iteraciones

COMENTARIOS FINALES

Algoritmos Genéticos

- ***basados en una metáfora biológica: evolución***
- ***gran potencialidad de aplicación***
- ***muy populares en muchos campos***
- ***muy potentes en diversas aplicaciones***
- ***altas prestaciones a bajo costo***

➢ *SON ATRACTIVOS DESDE UN PUNTO DE VISTA COMPUTACIONAL*

COMENTARIOS FINALES

Software de Algoritmos Genéticos

http://eodev.sourceforge.net/

EO is a templates-based, ANSI-C++ compliant evolutionary computation library. It contains classes for almost any kind of evolutionary computation you might come up to - at least for the ones we could think of. It is component-based, so that if you don't find the class you need in it, it is very easy to subclass existing abstract or concrete classes.

EO was started by the Geneura Team at the University of Granada, headed by Juan Julián Merelo.

Java version: GAJIT

COMENTARIOS FINALES

Software de Algoritmos Genéticos

http://jclec.sf.net

JCLEC Libreria en JAVA

JCLEC is a software system for Evolutionary Computation (EC) research, developed in the Java programming language. It provides a high-level software environment to do any kind of Evolutionary Algorithm (EA), with support for genetic algorithms (binary, integer and real encoding), genetic programming (Koza style, strongly typed, and grammar based) and evolutionary programming.

Maintained: Sebastián Ventura, Universad de Córdoba

Algoritmos Genéticos: Extensiones, estudios, modelos, …

Tema 7. Algoritmos Genéticos II. Diversidad y Convergencia

Tema 8. Algoritmos Genéticos III. Múltiples Soluciones en Problemas Multimodales

Tema 12. Algoritmos Evolutivos para Problemas Multiobjetivo

BIOINFORMÁTICA 2013 - 2014

PARTE I. INTRODUCCIÓN

- Tema 1. Computación Basada en Modelos Naturales

PARTE II. MODELOS BASADOS EN ADAPTACIÓN SOCIAL (Swarm Intelligence)

- Tema 2. Introducción a los Modelos Basados en Adaptación Social
- Tema 3. Optimización Basada en Colonias de Hormigas
- Tema 4. Optimización Basada en Nubes de Partículas (Particle Swarm)

PARTE III. COMPUTACÍON EVOLUTIVA

- Tema 5. Introducción a la Computación Evolutiva
- Tema 6. Algoritmos Genéticos I. Conceptos Básicos
- Tema 7. Algoritmos Genéticos II. Diversidad y Convergencia
- Tema 8. Algoritmos Genéticos III. Múltiples Soluciones en Problemas Multimodales
- Tema 9. Estrategias de Evolución y Programación Evolutiva
- Tema 10. Algoritmos Basados en Evolución Diferencial (Diferential Evolution – DE)
- Tema 11. Modelos de Evolución Basados en Estimación de Distribuciones (EDA)
- Tema 12. Algoritmos Evolutivos para Problemas Multiobjetivo
- Tema 13. Programación Genética
- Tema 14. Modelos Evolutivos de Aprendizaje

PARTE IV. OTROS MODELOS DE COMPUTACIÓN BIOINSPIRADOS

- Tema 15. Sistemas Inmunológicos Artificiales
- Tema 16. Otros Modelos de Computación Natural/Bioinspirados