



Universidad de Granada



Algoritmos Genéticos

Manuel Lozano

Email: lozano@decsai.ugr.es

***Técnicas de Soft Computing
para
Aprendizaje y Optimización***



**Departamento de Ciencias de la
Computación e Inteligencia Artificial**

Introducción a los AGs

Evolución natural

- Los organismos vivos poseen una consumada destreza en la resolución de problemas.
- Esta habilidad la obtienen de la *evolución*:
 - *Selección natural*
 - *Reproducción (innovación)*



Darwin, C. (1859). On the Origin of Species by Means of Natural Selection or the Preservations of Favored Races in the Struggle for Life. London: John Murray.

Introducción a los AGs

¿Qué es un Algoritmo Genético?

Los Algoritmos Genéticos (AGs)

Son algoritmos de optimización,
búsqueda y aprendizaje
inspirados en los procesos de

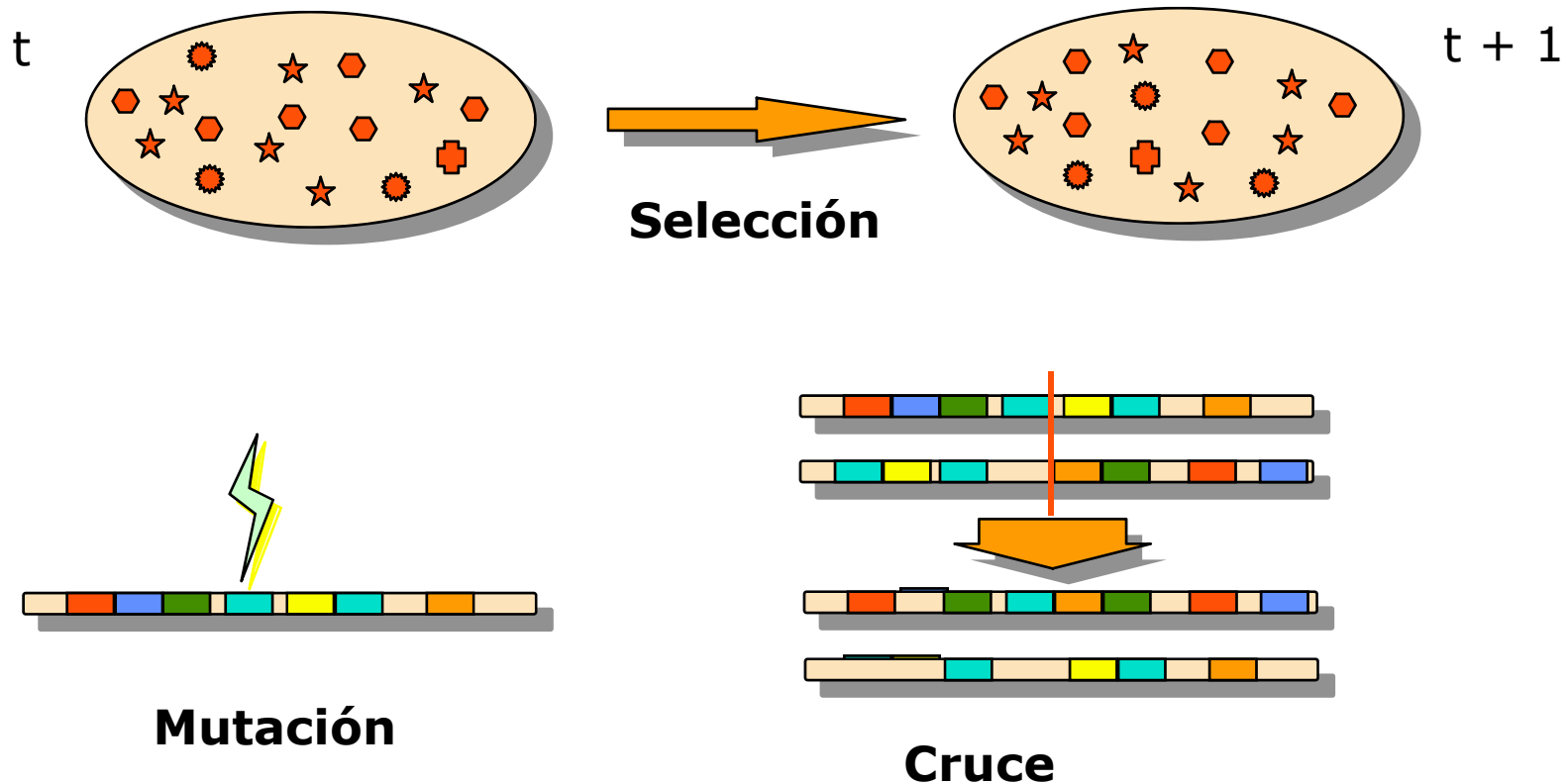
Evolución Natural y Evolución Genética



J. Holland. (1975). Adaptation in natural and artificial systems. The University of Michigan Press.

Introducción a los AGs

Componentes



AG = Competición + Innovación

Introducción a los AGs

Función de evaluación

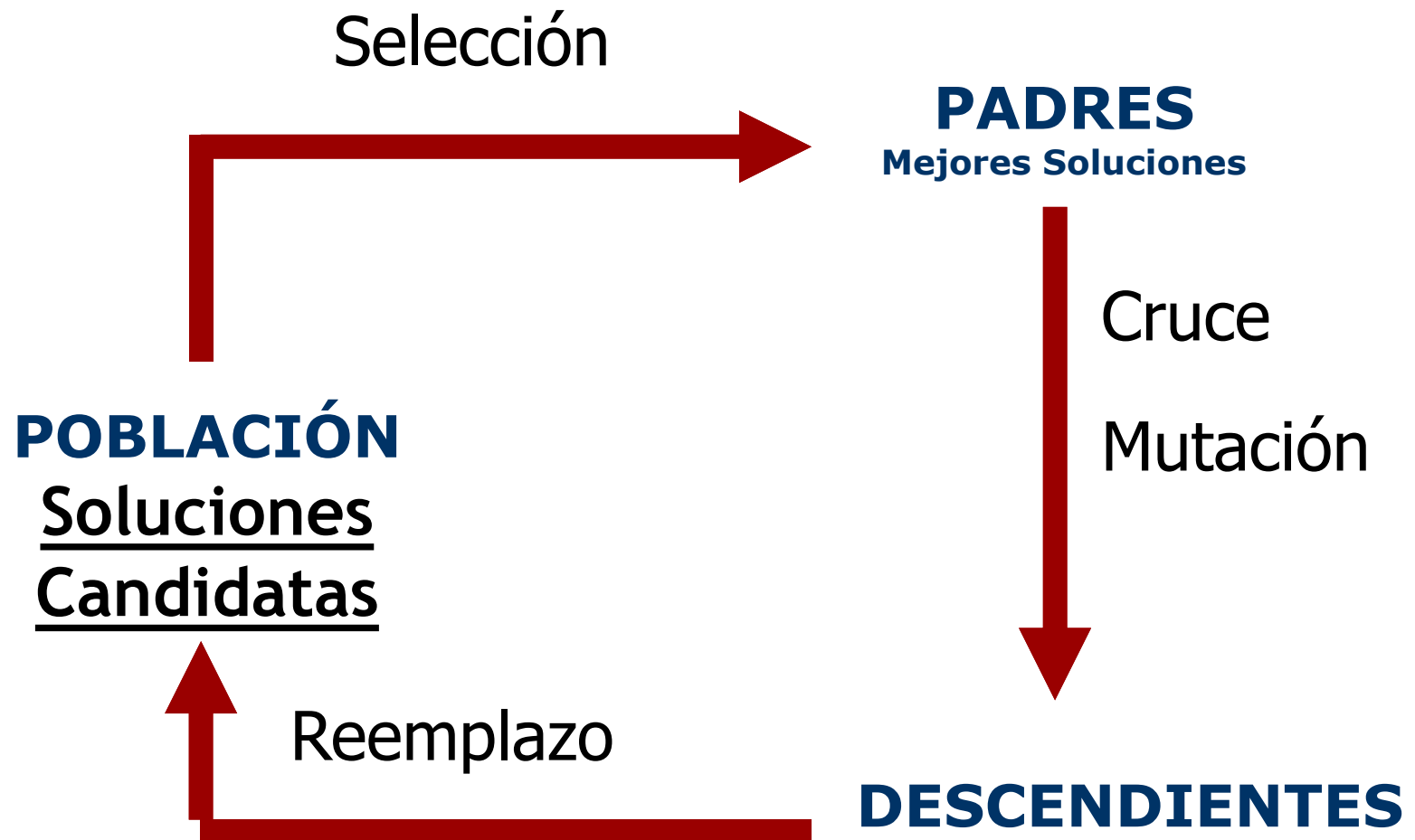
Función de evaluación (fitness, función objetivo, etc.)

Asigna un valor a cada solución candidata, cuantificando su **adecuación** como solución al problema dado

$$f: S \longrightarrow R$$

Introducción a los AGs

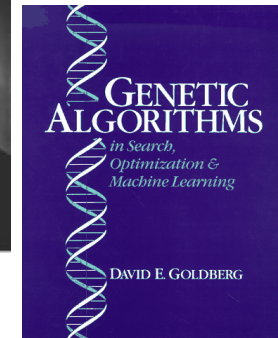
Ciclo de Evolución en los AGs



Introducción a los AGs

Bibliografía básica

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



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

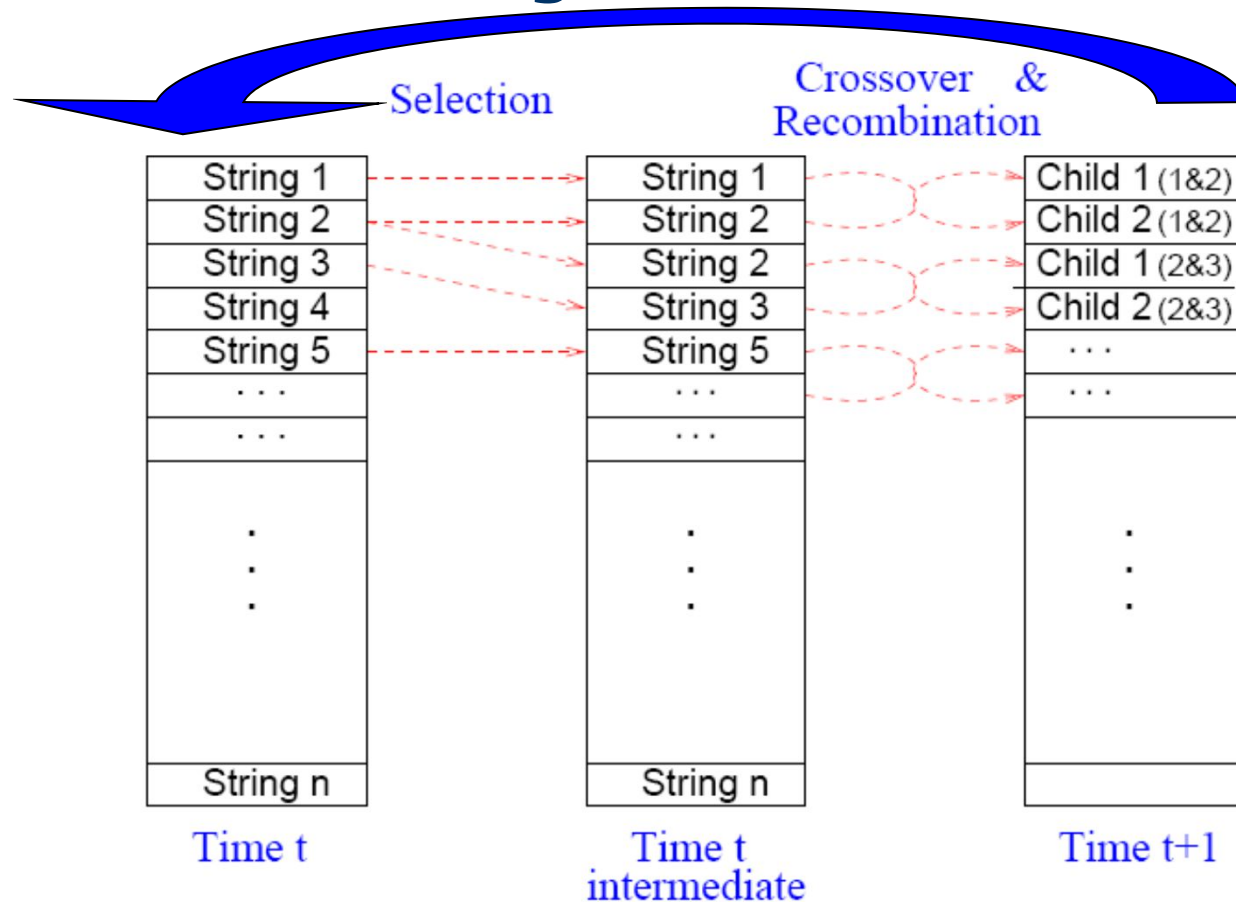
**T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary
Computation, Institute of Physics Publishers, 1997.**

**A.E. Eiben, J.E. Smith. Introduction to Evolutionary Computing.
Springer, 2003. (Segunda edición, 2015).**

Modelos de AGs

AG generacional y AG estacionario

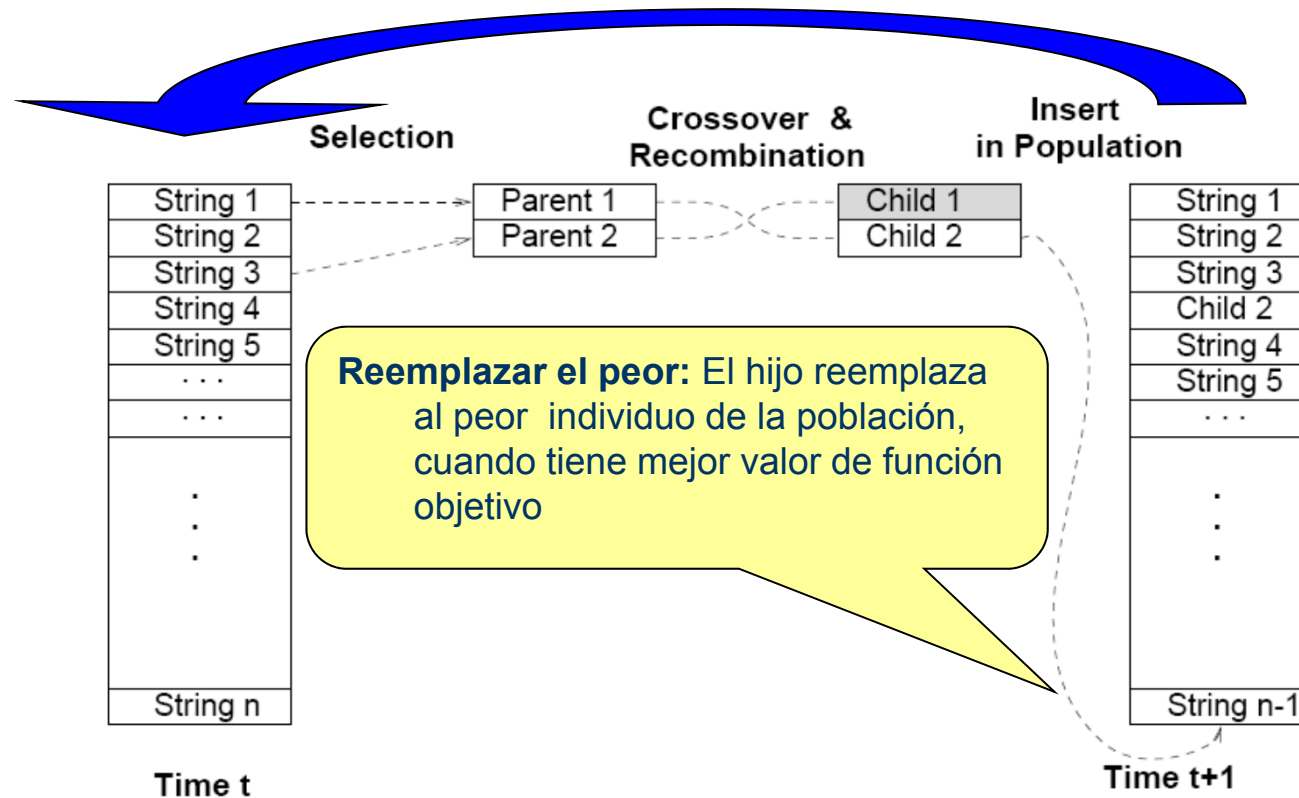
AG generacional



Modelos de AGs

AG generacional y AG estacionario

AG estacionario



Diseño de un AG

Pasos del diseño

Los pasos para diseñar un AG son:

- Escoger una **representación**
- Decidir cómo **inicializar** la población
- 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 AG

Diseño de un AG

Representación

Debemos disponer de un mecanismo para codificar las soluciones como un genotipo (cromosoma).

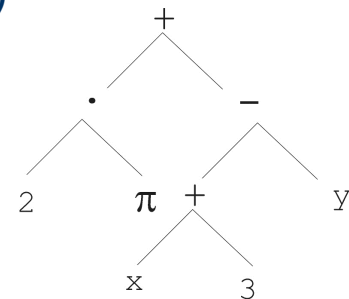
Codificación binaria (Problemas de optimización combinatoria)

1000100010111

Codificación real (Problemas de optimización continua)

2 3.4 5.6 8.7 7

Estructuras de árbol (Evolución de programas)

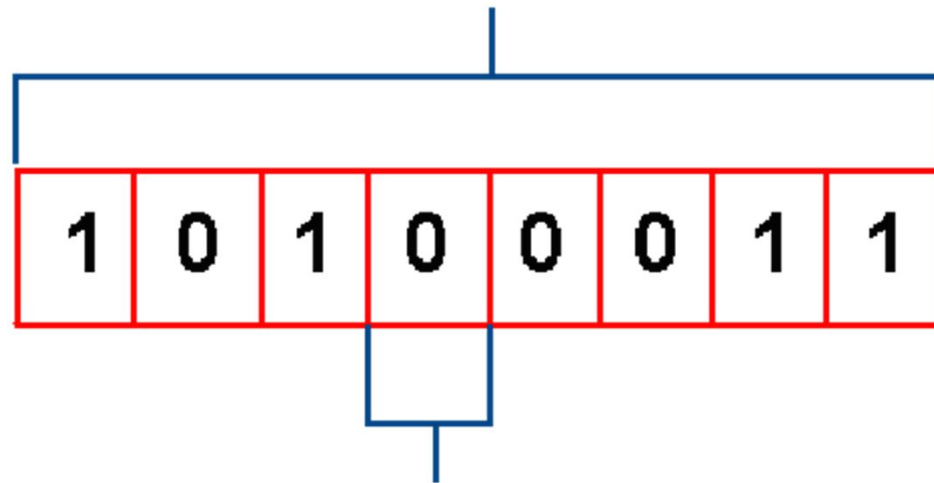


Diseño de un AG

Representación binaria

Para un problema dado, la representación binaria asigna cadena de ceros y unos a las soluciones.

CROMOSOMA

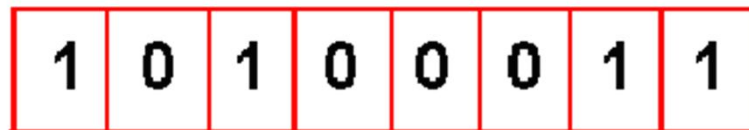


GEN

Diseño de un AG

Representación binaria

Genotipo (8 bits)



Fenotipo

- **Entero**
- **Número real**
- **Secuencia**
- ...
- **Otras ...**

Diseño de un AG

Representación binaria

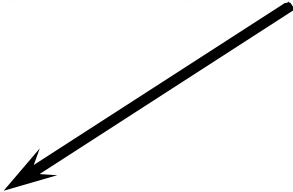
Los cromosomas binarios pueden representar números enteros

Genotipo:

1	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Fenotipo:

= 163


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

$$128 + 32 + 2 + 1 = 163$$

Diseño de un AG

Representación binaria

También pueden representar números reales


Ejemplo: Número entre 2.5 y 20.5 utilizando 8 dígitos binarios

Genotipo:

1	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Fenotipo:

= 13.9609


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

Diseño de un AG

Representación real

- Representación natural para problemas con variables en dominios continuos.
- Las soluciones se codifican utilizando valores reales como genes directamente
- Muchas aplicaciones reales tienen esta forma de codificación

Diseño de un AG

Representación real

- Los individuos se representan como vectores de valores reales:

Cromosoma: $X=(x_1, \dots, x_n)$, x_i pertenece a R

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

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

Diseño de un AG

Representación de orden

- Los individuos se representan como permutaciones.

7	3	6	8	2	4	1	5
---	---	---	---	---	---	---	---

- Se utilizan para problemas de secuenciación.
- Ejemplo: Viajante de Comercio, donde cada ciudad tiene asignado un único número entre 1 y n.
- Es necesario usar operadores genéticos especiales que garanticen que el resultado de aplicarlos siga siendo una permutación.

Diseño de un AG

Inicialización

- Aleatoria sobre el espacio de búsqueda.
 - 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 (algoritmo greedy) previa (partir de buenas soluciones).

¿CÓMO SE CONSTRUYE UN AG?

Estrategia de Selección

- Debemos de garantizar que los mejores individuos tienen una mayor posibilidad de ser padres frente a los individuos menos buenos.
- También hay que dar oportunidad a individuos menos buenos. Éstos pueden incluir material genético útil.
- Presión selectiva: determina en qué grado la reproducción está dirigida por los mejores individuos.

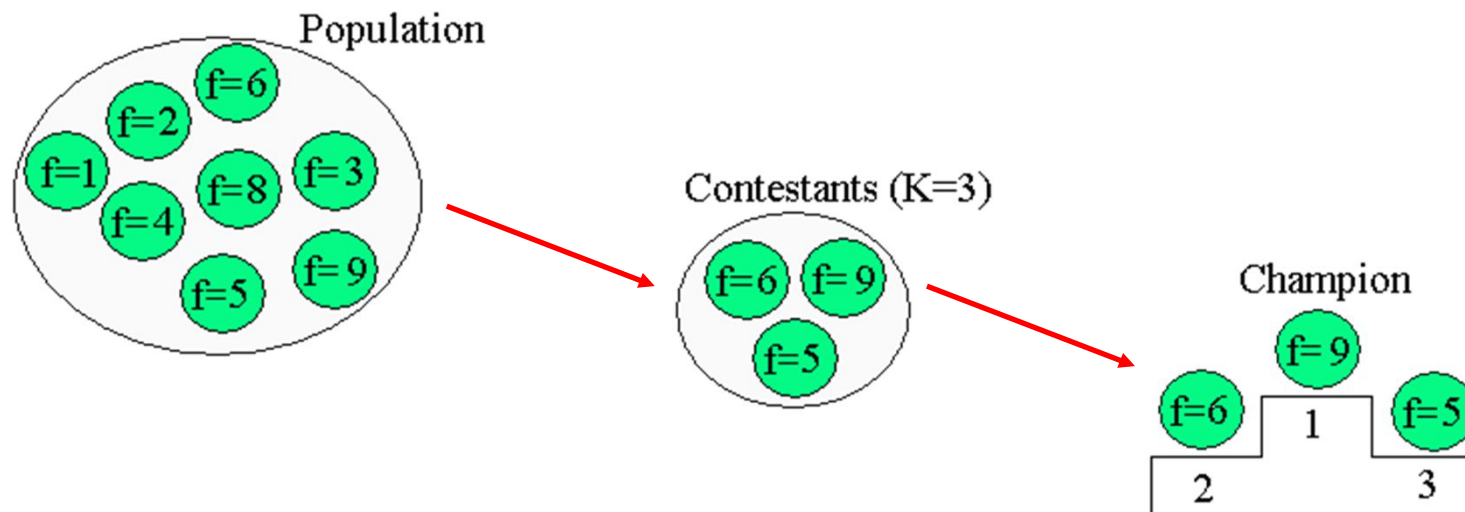
¿CÓMO SE CONSTRUYE UN AG?

Selección por torneo

Para seleccionar un padre:

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

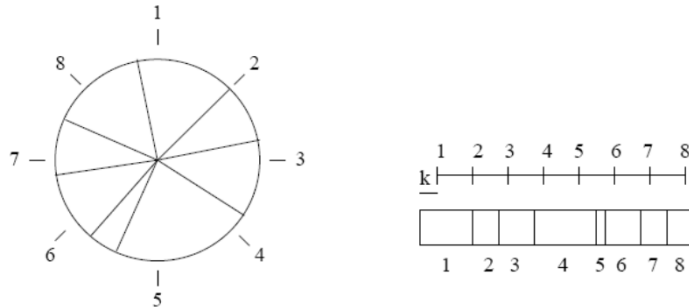
k se denomina *tamaño del torneo*. A mayor k , mayor presión selectiva y viceversa.



¿CÓMO SE CONSTRUYE UN AG?

Otros esquemas de selección

- Selección por ruleta: Se asigna una probabilidad de selección proporcional al valor del fitness de los cromosomas.



- Orden lineal: 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.
- Emparejamiento variado inverso: Se escoge un primer padre, después, para el otro se seleccionan N_{nam} padres y se elige el más diferente (distante) al primero.

¿CÓMO SE CONSTRUYE UN AG?

Operador de cruce

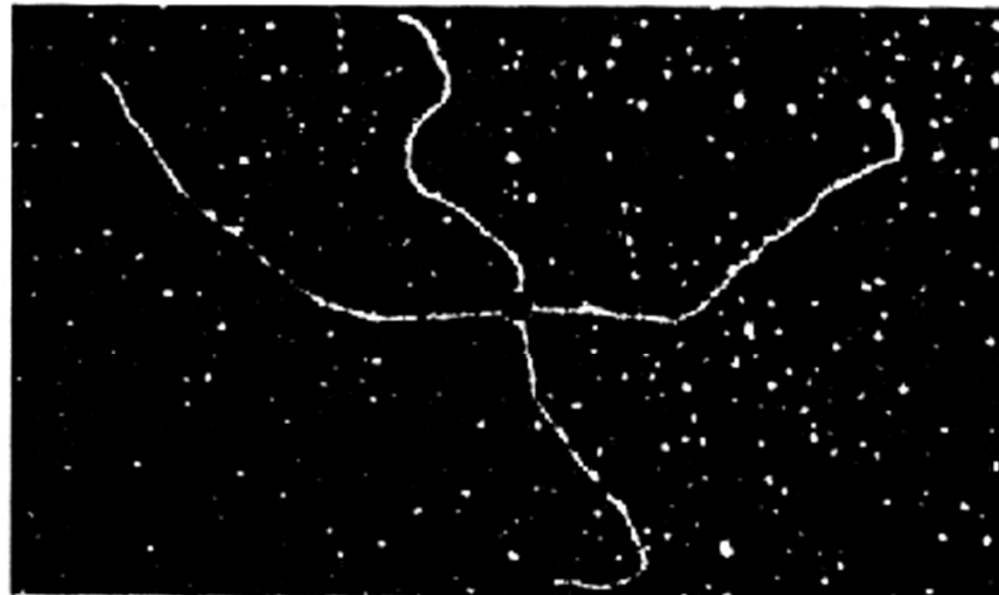
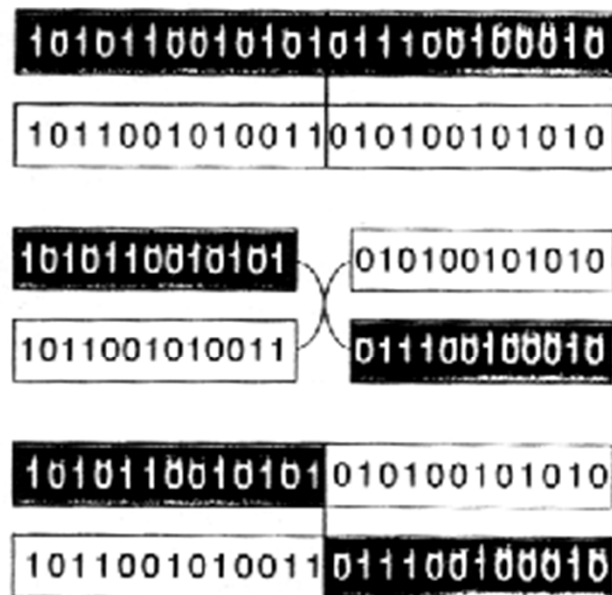
Aspectos a tener en cuenta sobre el cruce:

- Los hijos deben heredar algunas características de cada padre.
- Se debe diseñar de acuerdo a la representación.
- La recombinación debe producir cromosomas válidos.
- Probabilidad de cruce (p_c) entre 0.6 y 0.9.

¿CÓMO SE CONSTRUYE UN AG?

Operador de cruce para cod. binaria

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

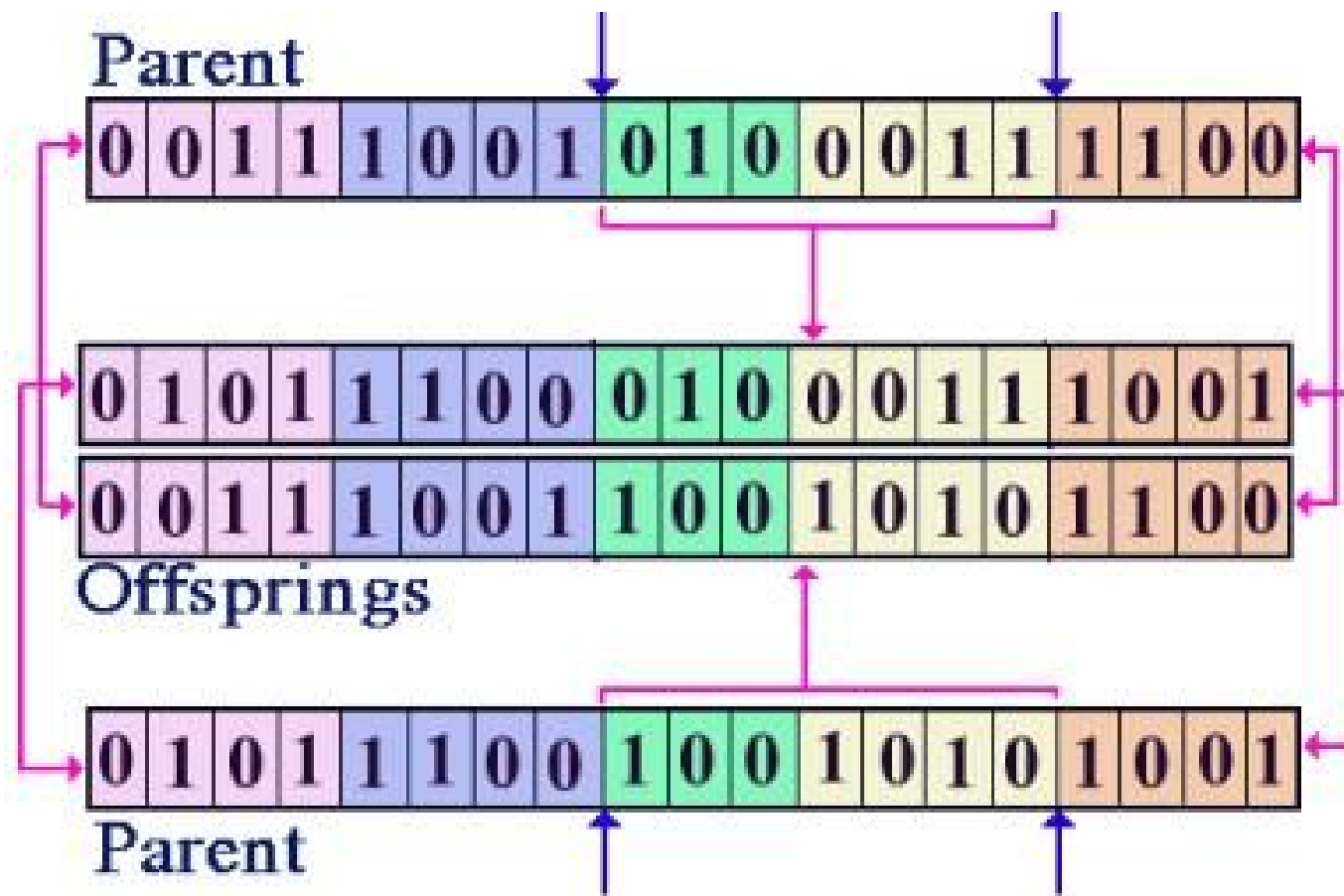


CROSSOVER is the fundamental mechanism of genetic rearrangement for both real organisms and genetic algorithms.

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

¿CÓMO SE CONSTRUYE UN AG?

Operador de cruce en dos puntos



¿CÓMO SE CONSTRUYE UN AG?

Operador de cruce para Cod. Real: BLX- α

- Dados 2 cromosomas

$$X = (x_1, \dots, x_n) \text{ e } Y = (y_1, \dots, y_n)$$

- BLX- α genera un descendiente

$$H = (h_1, \dots, h_i, \dots, h_n)$$

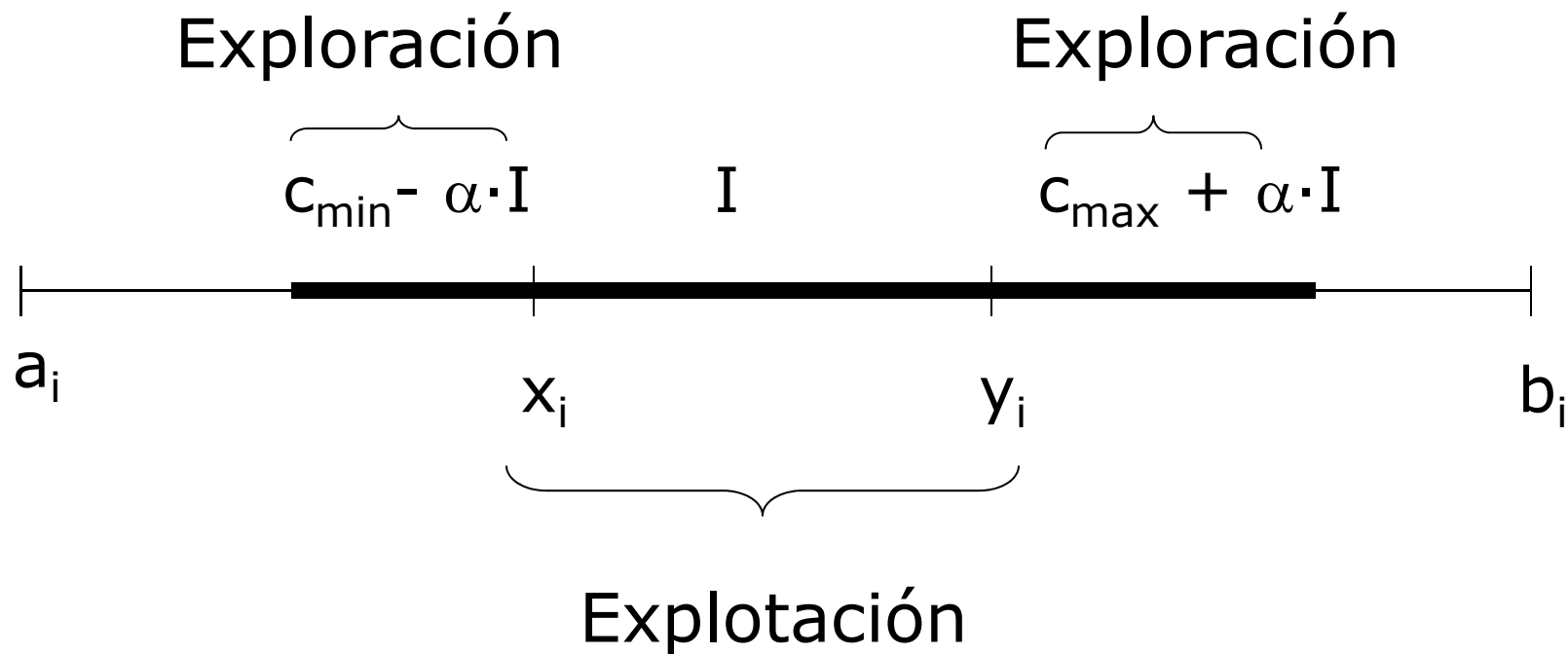
- donde h_i se genera aleatoriamente en el intervalo:

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

- $C_{max} = \max \{x_i, y_i\}$
- $C_{min} = \min \{x_i, y_i\}$
- $I = C_{max} - C_{min}, \alpha \in [0, 1]$

¿CÓMO SE CONSTRUYE UN AG?

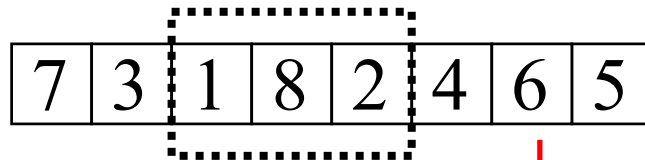
Operador de cruce BLX- α (cod. real)



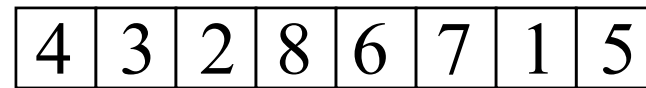
¿CÓMO SE CONSTRUYE UN AG?

Operador de cruce para rep. de orden

Padre 1



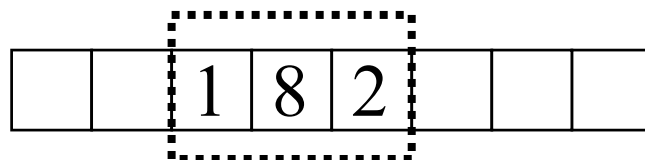
Padre 2



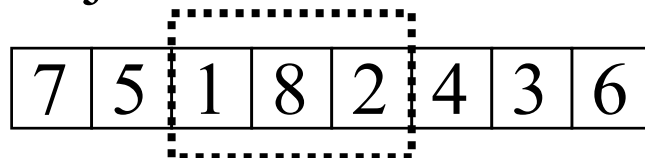
7, 3, 4, 6, 5

Orden

4, 3, 6, 7, 5



Hijo 1



¿CÓMO SE CONSTRUYE UN AG?

Operador de mutación

Aspectos a tener en cuenta sobre la mutación:

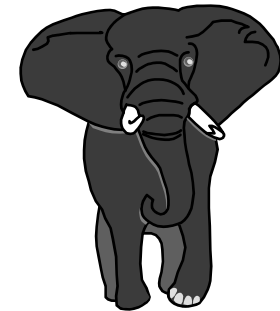
- Debe permitir alcanzar cualquier parte del espacio de búsqueda.
- El tamaño de la mutación debe ser controlado.
- Debe producir cromosomas válidos.
- Probabilidad de mutación (p_m). Determina la frecuencia con la cual se mutan los genes.

Operador de mutación para cod. binaria

1 1 1 1 | 1 1 1

1 1 1 0 | 1 1 1

A cartoon illustration of a grey elephant with white tusks and a trunk, standing and facing forward. The elephant is depicted in a simple, friendly style with large ears and a small tusk on the right side.



Normalmente, se usa p_m muy baja,
para no degradar el material genético!

¿CÓMO SE CONSTRUYE UN AG?

Operador de mutación para cod. real

Perturbación de los valores mediante un valor aleatorio.

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

- 0 es la media
- σ es la desviación típica (step size)

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

para cada parámetro.

¿CÓMO SE CONSTRUYE UN AG?

Operador de mutación para rep. de orden

Selección aleatoria de dos genes e intercambio de ambos.

7	3	1	8	2	4	6	5
---	---	---	---	---	---	---	---



7	3	6	8	2	4	1	5
---	---	---	---	---	---	---	---

¿CÓMO SE CONSTRUYE UN AG?

Estrategias de reemplazo

Cuando se considera un modelo estacionario nos encontramos con diferentes propuestas:

- Reemplazar al peor de la población.
- Reemplazar al más viejo.
- Torneo restringido: Se reemplaza al más parecido de entre w individuos (w es un parámetro del modelo).
- Reemplazar un individuo aleatorio.

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

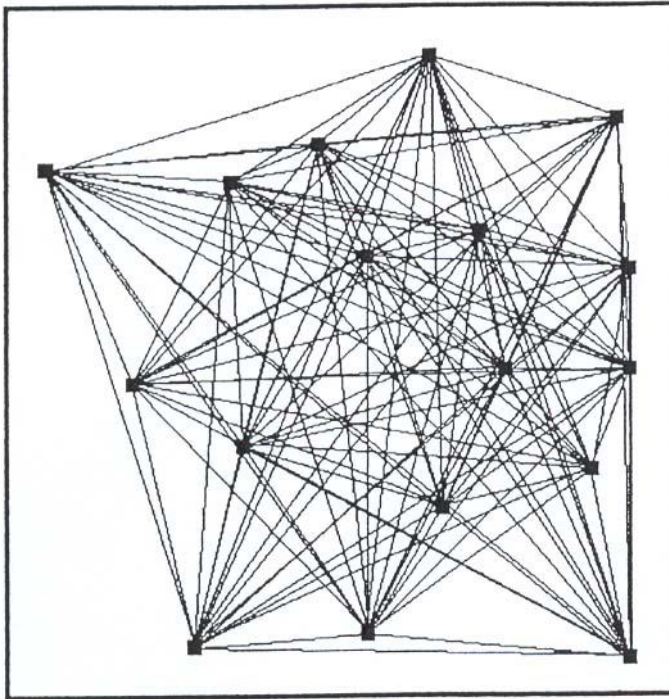
Objetivo: Suma de la distancia entre las ciudades.

Población: 61 cromosomas - Elitismo

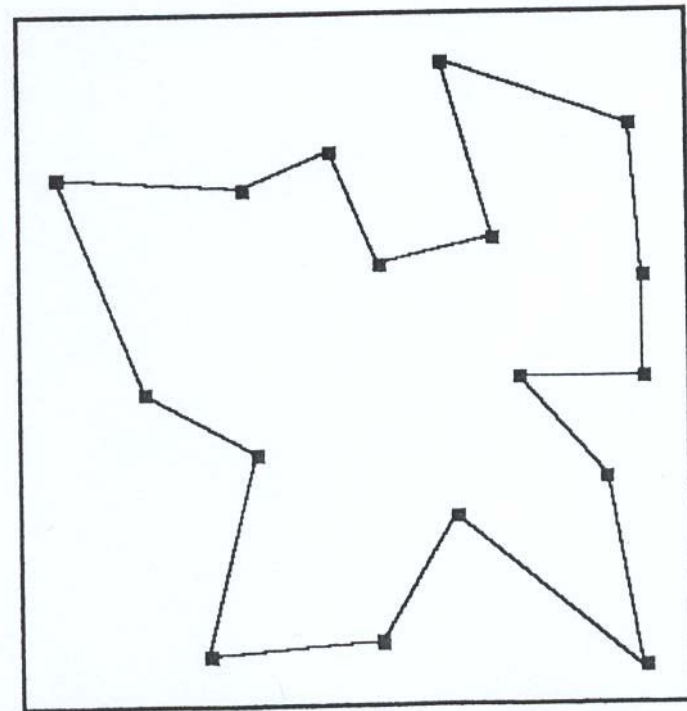
Cruce: OX ($P_c = 0.6$)

Mutación: Inversión de una lista ($P_m = 0.01$ - cromosoma)

EJEMPLO: VIAJANTE DE COMERCIO

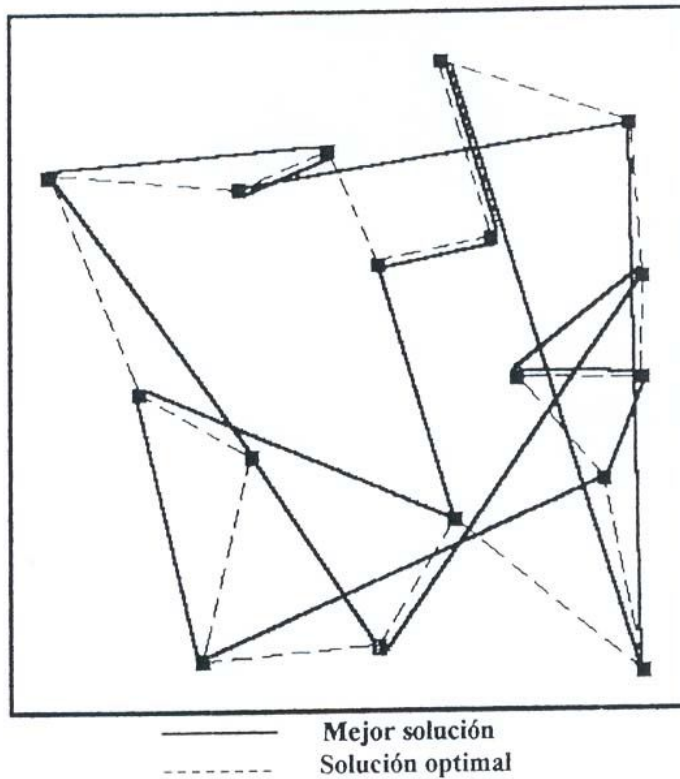


$17! = 3.5568743 \text{ e}14$ recorridos posibles

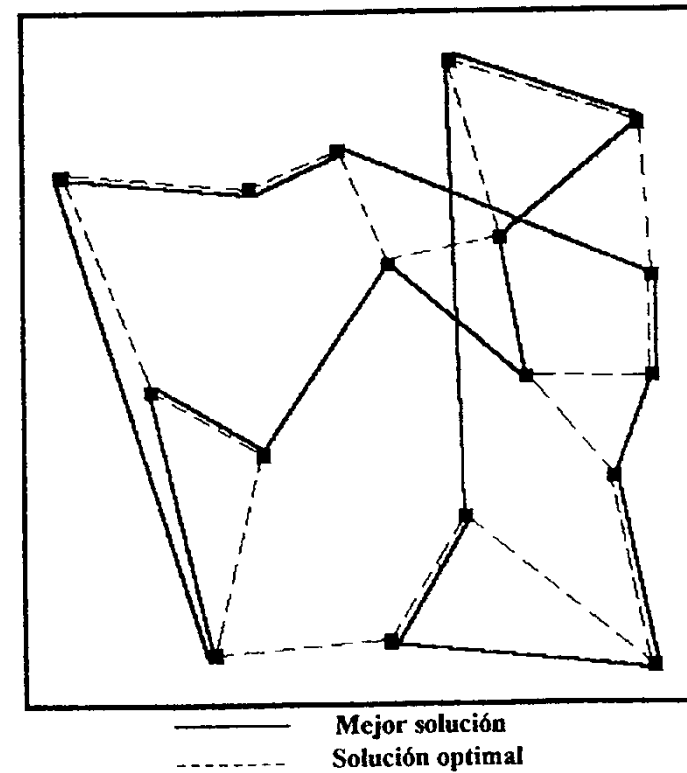


Solución óptima: 226.64

EJEMPLO: VIAJANTE DE COMERCIO



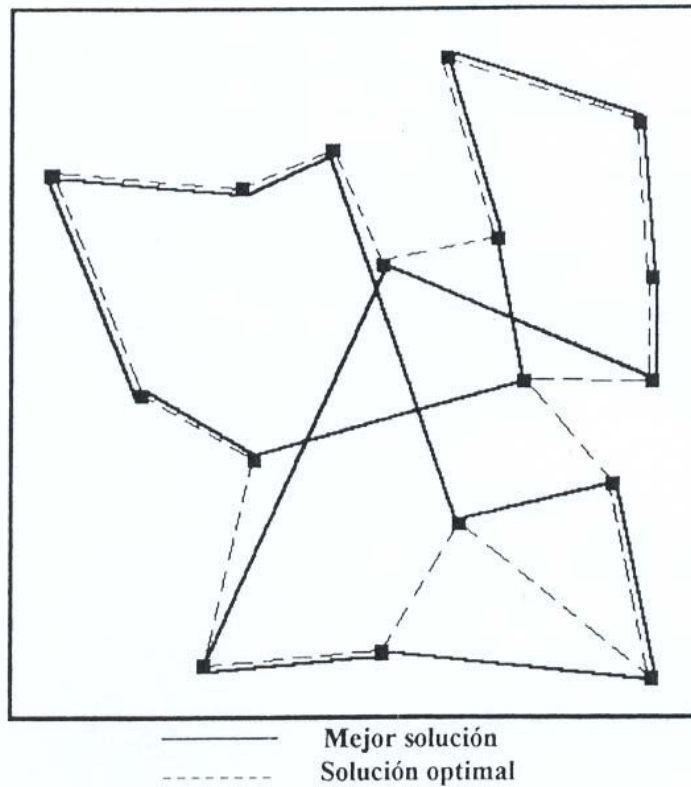
Iteración: 0 Costo: 403.7



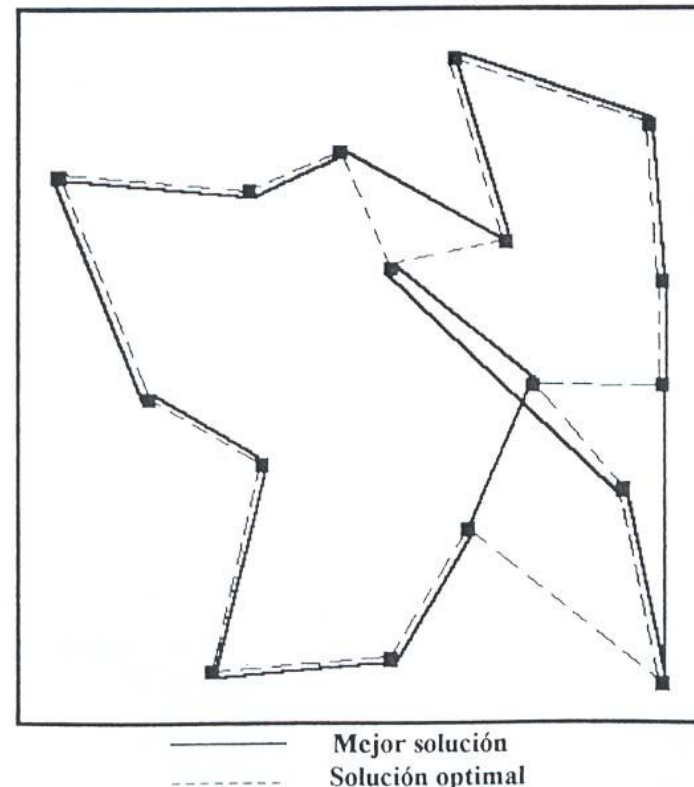
Iteración: 25 Costo: 303.86

Solución óptima: 226.64

EJEMPLO: VIAJANTE DE COMERCIO



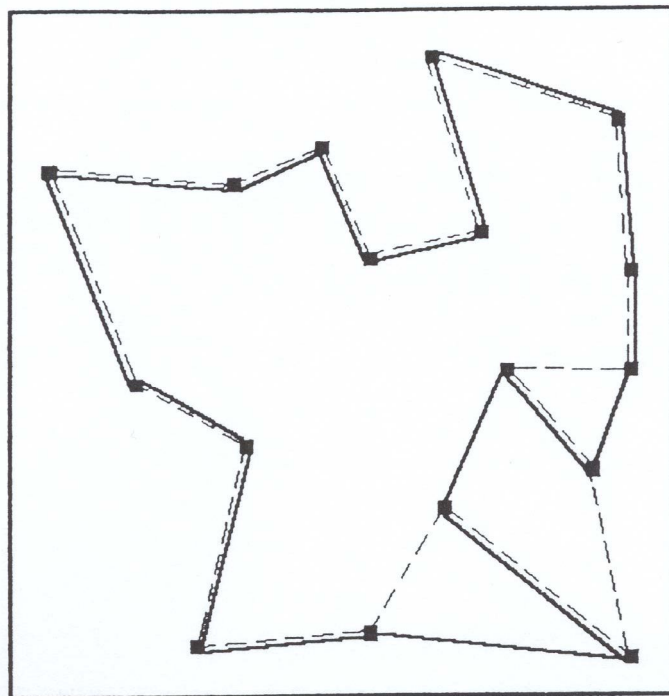
Iteración: 50 Costo: 293.6



Iteración: 100 Costo: 256.55

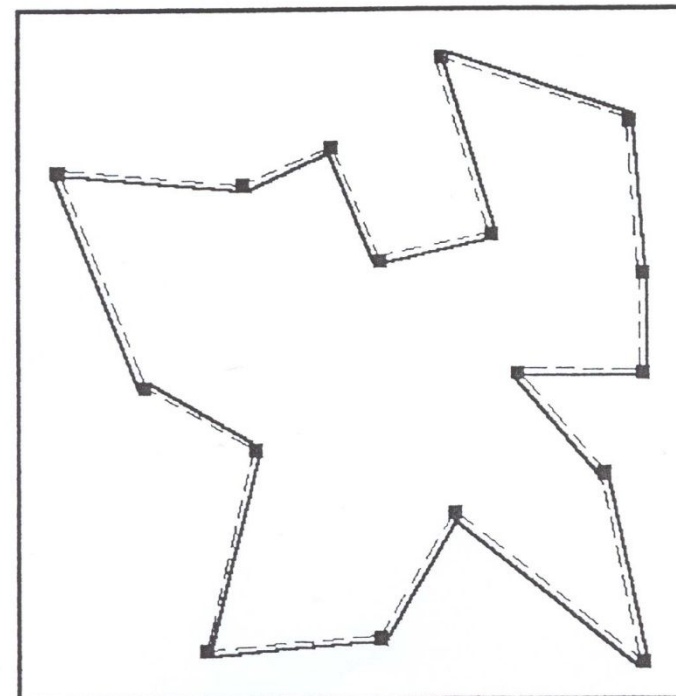
Solución óptima: 226.64

EJEMPLO: VIAJANTE DE COMERCIO



——— Mejor solución
----- Solución optimal

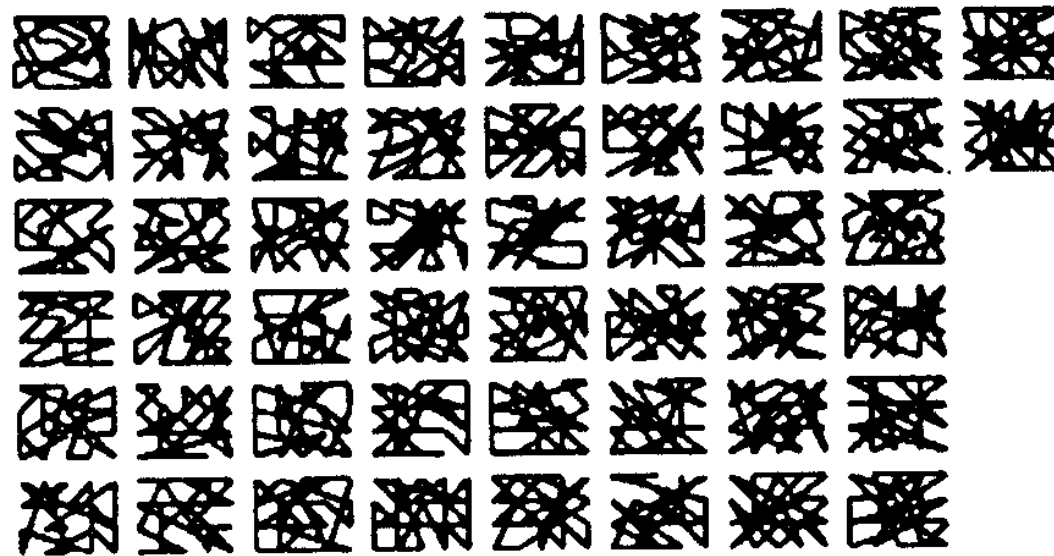
Iteración: 200 Costo: 231.4



——— Mejor solución
----- Solución optimal

Iteración: 250 Solución
óptima: 226.64

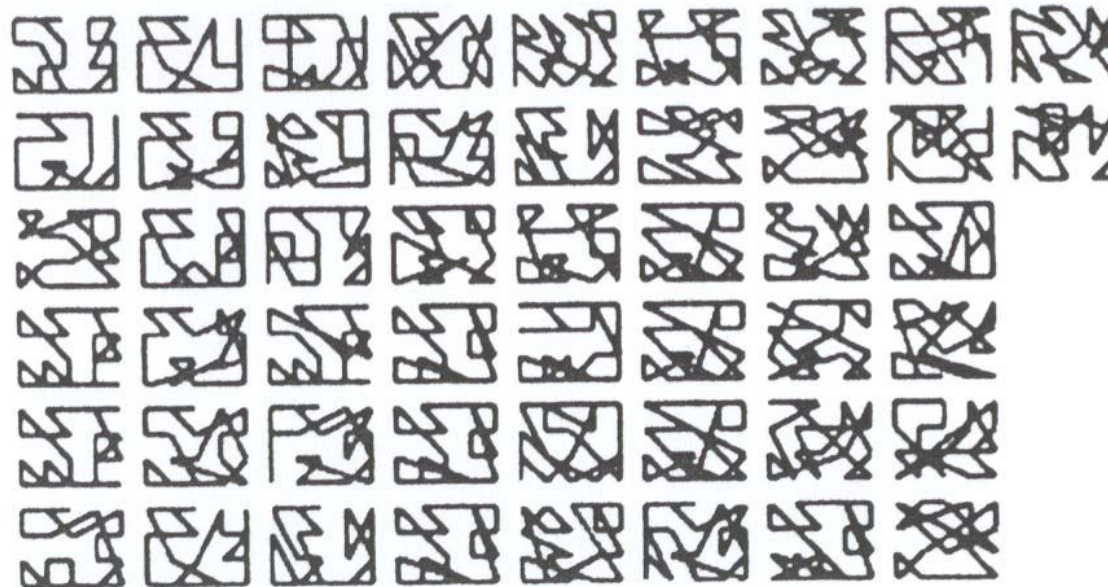
EJEMPLO: VIAJANTE DE COMERCIO



(0)

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

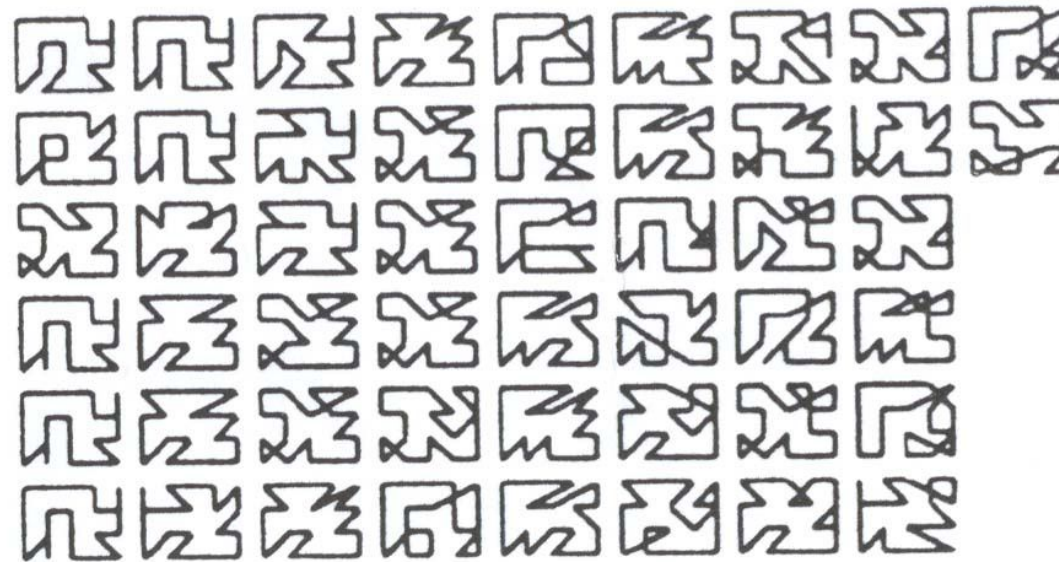
EJEMPLO: VIAJANTE DE COMERCIO



(10)

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

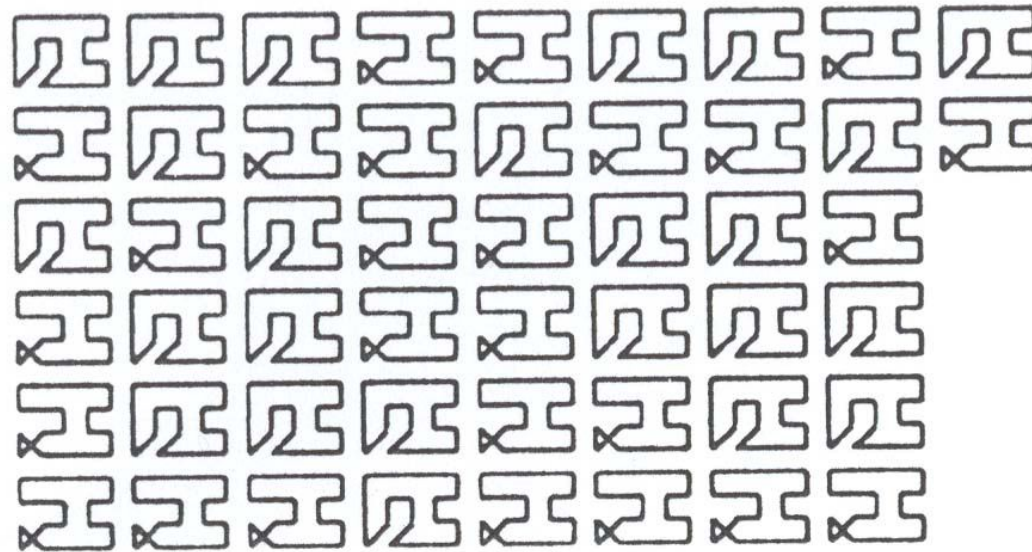
EJEMPLO: VIAJANTE DE COMERCIO



(30)

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

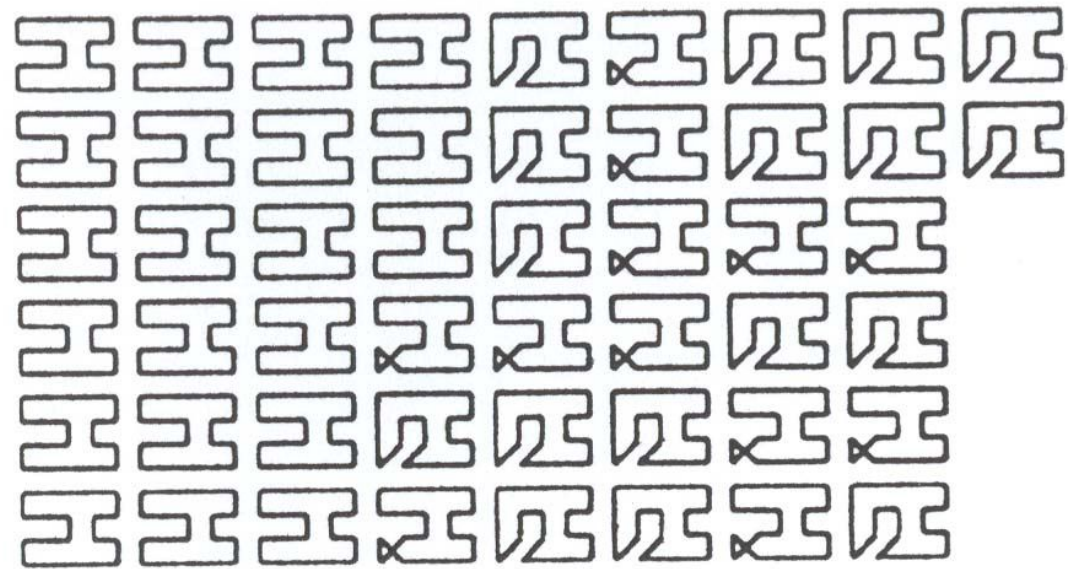
EJEMPLO: VIAJANTE DE COMERCIO



(50)

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

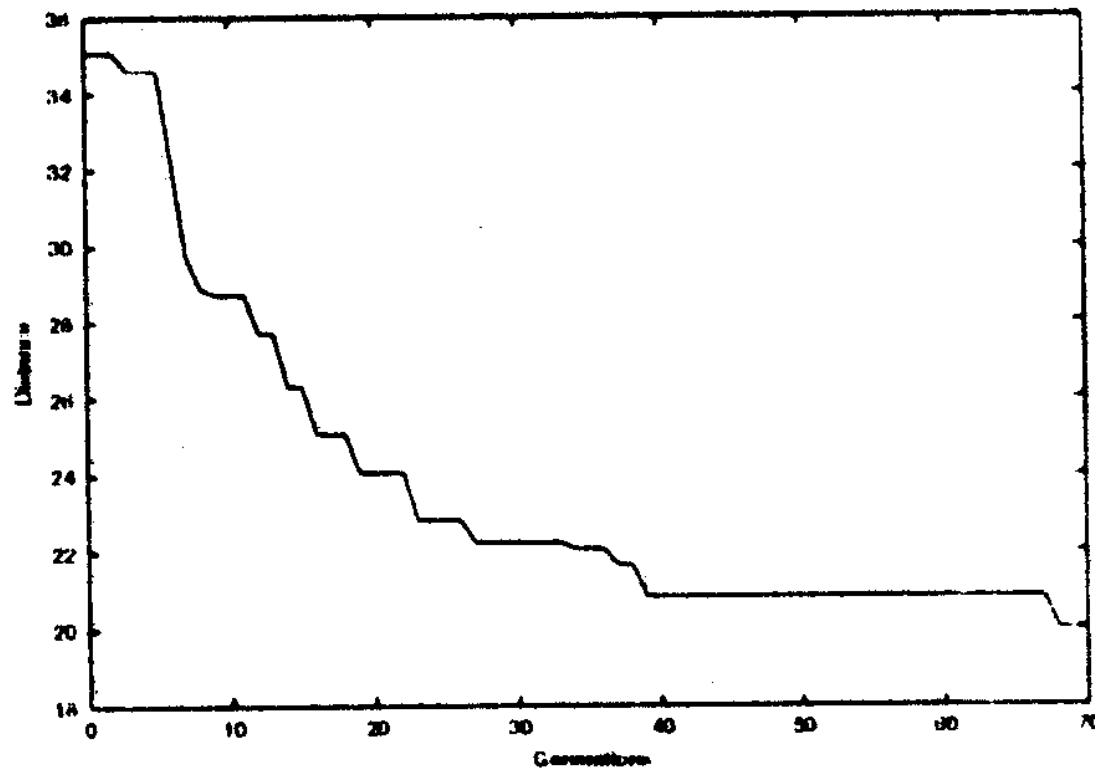
EJEMPLO: VIAJANTE DE COMERCIO



(70)

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

EJEMPLO: VIAJANTE DE COMERCIO



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

Investigación sobre AGs

Líneas más importantes

Las principales líneas de investigación se han centrado en:

- Aplicación de los AGs
 - *Problemas de optimización específicos*
 - *Problemas multimodales*
 - *Problemas con restricciones*
 - *Problemas multiobjetivo*
- Mejorar el comportamiento de los AGs

Investigación sobre AGs

AGs para problemas de opt. continuos



Special Issue of Soft Computing:

**Scalability of Evolutionary Algorithms and
other Metaheuristics for Large Scale
Continuous Optimization Problems**

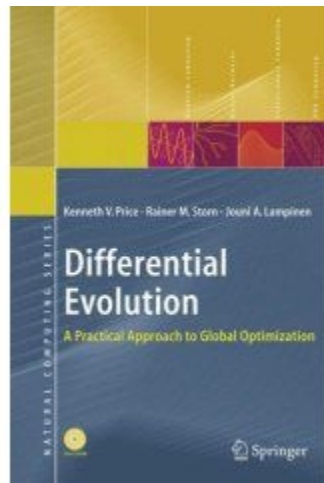
Volume 15, Number 11, 2011



**Special Session & Competition on
Large Scale Global Optimization at
CEC 2010, 2012, 2013 ...**

Investigación sobre AGs

Differential Evolution



The DE approach (Storn and Price (1997)) starts from an initial population of solutions that are mutated and crossed over to eventually obtain better solutions for the optimization problem at hand.



R. Storn and K. V. Price, “Differential evolution-A simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” *Journal of Global Optimization*, 11:341-359,1997.

Investigación sobre AGs

Estrategias de evolución: CMA-ES



- Hansen, N. and A. Ostermeier (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2), pp. 159-195;
- Hansen, N., S.D. Müller and P. Koumoutsakos (2003). Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1), pp. 1-18;

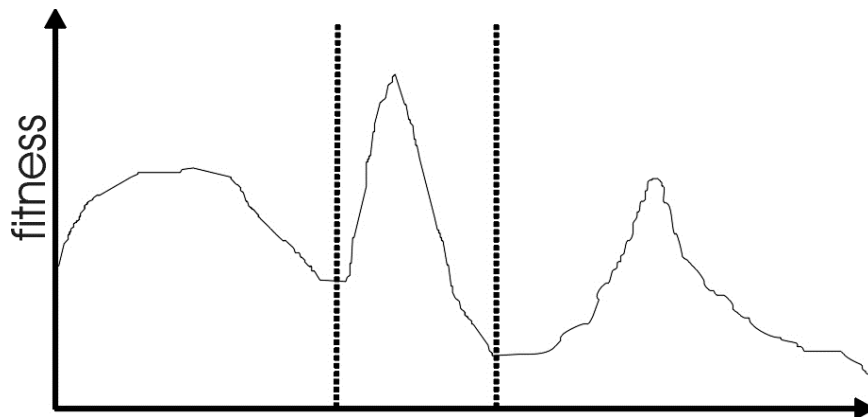
Nikolaus Hansen

www.lri.fr/~hansen/

<http://cma.gforge.inria.fr/>

Investigación sobre AGs

AGs para problemas multimodales

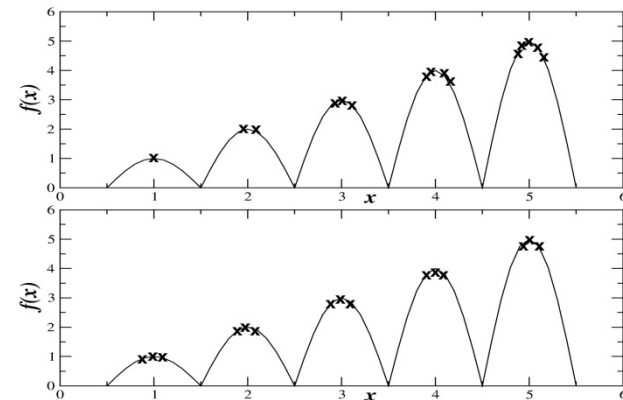


El AG trata de obtener los óptimos de una función multimodal

- Métodos de sharing
- Métodos de crowding

B. Sareni, L. Krähenbühl. **Fitness sharing and niching methods revisited**. IEEE Transactions on Evolutionary Computation 2(3): 97–106, 1998.

J.-P. Li, M. E.B., G. T.P., C. P.J. **A species conserving genetic algorithm for multimodal function optimization**. Evolutionary Computation 10(3): 207–234, 2002.



Investigación sobre AGs

AGs para problemas con restricciones

Esp. Búsq. = Esp. más general + Conj.
restricciones

$$\min f(x_1, \dots, x_n)$$

$$L_i \leq x_i \leq U_i \quad \text{Para } i = 1, \dots, n$$

$$g_i(x) \leq 0 \quad \text{Para } i = 1, \dots, q$$

$$h_i(x) = 0 \quad \text{Para } i = q+1, \dots, m$$

Z. Michalewicz, M. Schoenauer. **Evolutionary algorithms for constrained parameter optimization problems.** Evolutionary Computation 4(1): 1-32, 1996.

Investigación sobre AGs

AGs para problemas multiobjetivo

Optimizar $f(x) = (f_1(x), \dots, f_n(x))$

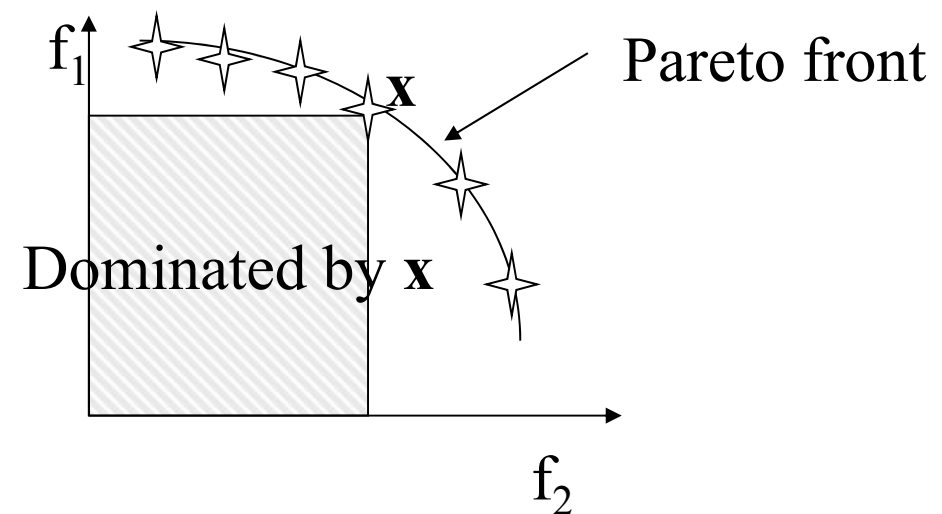
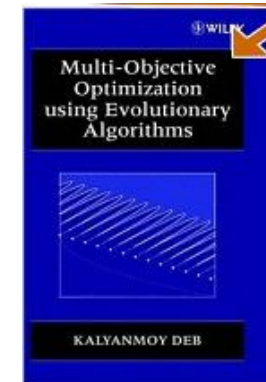
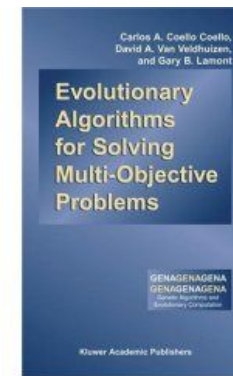
$f_i(\cdot)$ son objetivos posiblemente conflictivos

Soluciones con AGs:

- Agregar objetivos

$$f'(x) = \sum_{i=1}^n w_i f_i(x)$$

- Métodos basados en conjunto óptimo Pareto



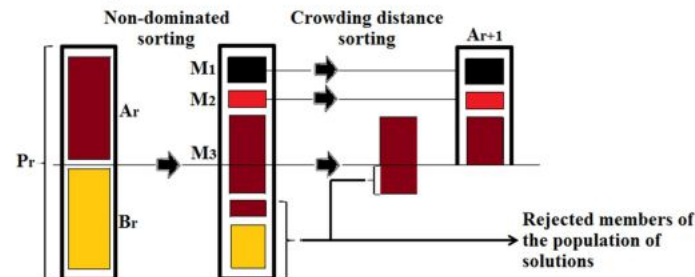
Investigación sobre AGs

AGs para problemas multiobjetivo

- Modelos recomendados:

- Multi-objetivo: 2-3 objetivos

MOEA/D
NSGA-II



- Many-Objective: más de 3 objetivos.

NSGA-III

Investigación sobre AGs

AGs para problemas multiobjetivo

■ Software disponible:

- **jMetal** (Java) ó **jMetalPy** (Python)
 - Algoritmos Multi-objetivo.
 - Fácil de usar, problemas propios.
 - Se programa, permite gráficas.



Investigación sobre AGs

Mejora del comportamiento de los AGs

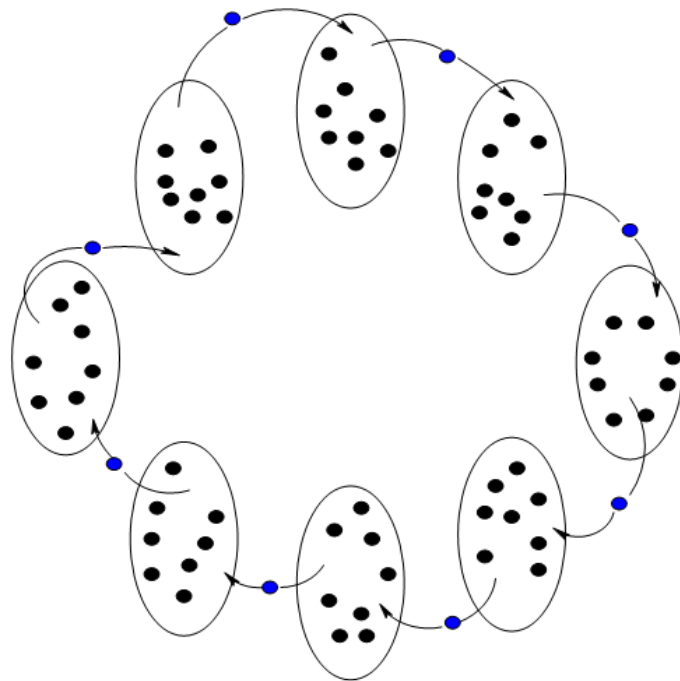
PROBLEMA: *Convergencia prematura hacia zonas que no contienen el óptimo global*

SOLUCIONES: *Equilibrio adecuado entre exploración y explotación mediante:*

- AGs paralelos (AGs distribuidos y celulares)
- Control de parámetros (N , p_c , p_m , etc.)
- Operadores más efectivos (codificación real, de orden, etc.).
- Equilibrio diversidad/convergencia.
- Hibridación de los AGs con otras técnicas de búsqueda (algoritmos meméticos) y otras tecnologías inteligentes.

Investigación sobre AGs

AGs distribuidos



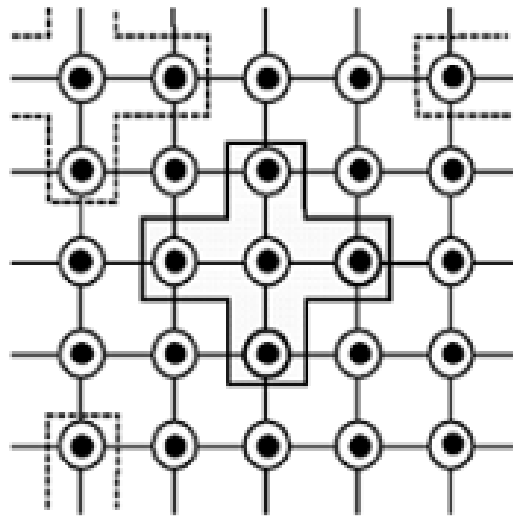
*Se aplican AGs independientes
sobre distintas subpoblaciones*

*Un operador de migración
intercambia cromosomas entre
las subpoblaciones*

F. Herrera, M. Lozano (2000). Gradual distributed real-coded genetic algorithms. IEEE Transactions on Evolutionary Computation 4(1): 43-63.

Investigación sobre AGs

AGs celulares



E. Alba, B. Dorronsoro (2005). The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Transactions on Evolutionary Computation* 9(2): 126-142.

Investigación sobre AGs

Control de los parámetros del AG

GAs have many **strategy parameters**, e.g.

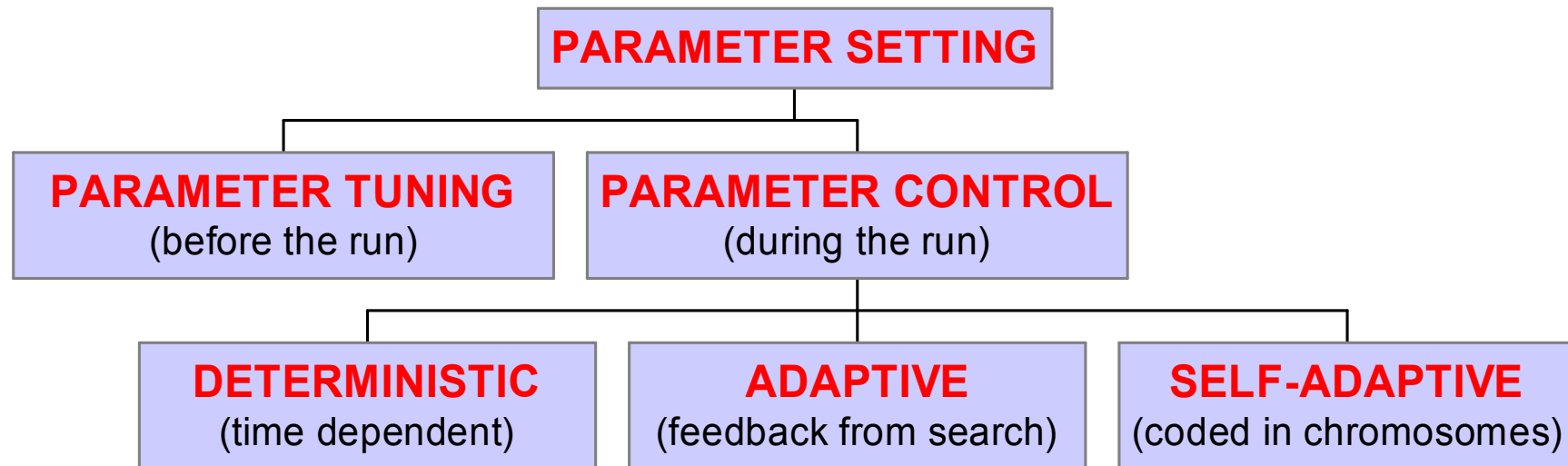
- mutation operator and mutation rate
- crossover operator and crossover rate
- selection mechanism and selective pressure
- population size

However, finding **robust control parameter** settings is not a trivial task:

- Their interaction with GA performance is a **complex** relationship
- The optimal ones are **problem-dependent**
- **Different** control parameter values **may be necessary** during the GA run

Investigación sobre AGs

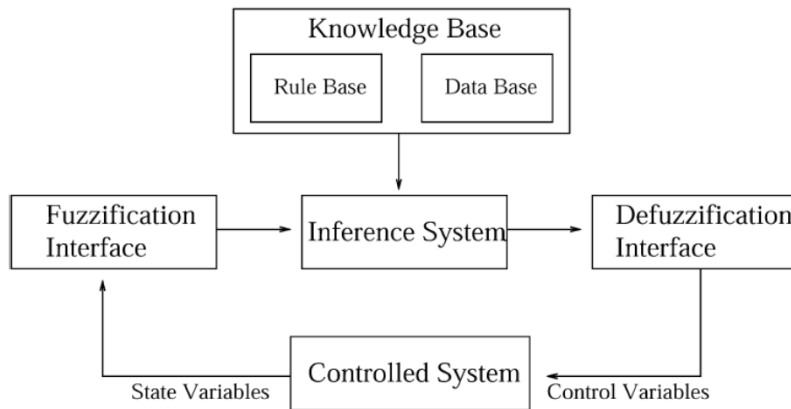
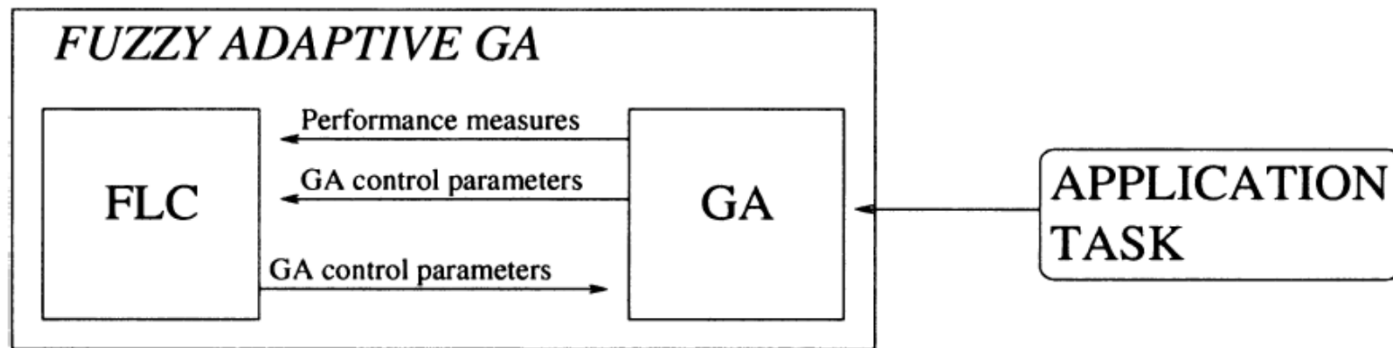
Control de los parámetros del AG



- Angeline PJ (1995) **Adaptive and self-adaptive evolutionary computations.** In: Computational Intelligence: A Dynamic Systems Perspective (pp.152-163). Piscataway, NJ: IEEE Press.
- Eiben AE, Hinterding R, Michalewicz Z (1999) **Parameter control in evolutionary algorithms.** IEEE Trans. Evolutionary Computation, 3(2), 124-141.
- Tuson AL, Ross P (1998) **Adapting operator settings in genetic algorithms.** Evolutionary Computation, 6(2), 161-184.

Investigación sobre AGs

Adaptive GA control based on fuzzy logic controllers



F. Herrera, M. Lozano. **Fuzzy Adaptive Genetic Algorithms: Design, Taxonomy and Future Directions.**

Soft Computing 7:8 (2003) 545-562.

Herrera F, Lozano M (2001) **Adaptive genetic operators based on coevolution with fuzzy behaviours.** IEEE Trans. on Evolutionary Computation, 5(2), 1-18.

Algoritmos Evolutivos

Computación Evolutiva

Existen cuatro paradigmas básicos:

Algoritmos Genéticos que utilizan operadores genéticos sobre cromosomas. **1975, Michigan University**



John Holland

Professor of CS and Psychology at the U. of Michigan.

Estrategias de Evolución que enfatizan los cambios de comportamiento al nivel de los individuos. **1964, Technische Universität Berlin**



Hans-Paul Schwefel
Universität Dortmund



Ing. Ingo Rechenberg
Bionics & Evolutionstechnik
Technical University Berlin
<http://www.bionik.tu-berlin.de/>

Programación Evolutiva que enfatizan los cambios de comportamiento al nivel de las especies. **1960-1966, Florida**



Lawrence J. Fogel,
Natural Selection, Inc.

Programación Genética que evoluciona expresiones representadas como árboles.
1989, Stanford University



John Koza
Stanford University.