

Tema 2: Metaheurísticas Poblacionales (Evolutivas)

Algoritmos Genéticos

Algoritmos meméticos

Búsqueda dispersa (*Scatter Search*)

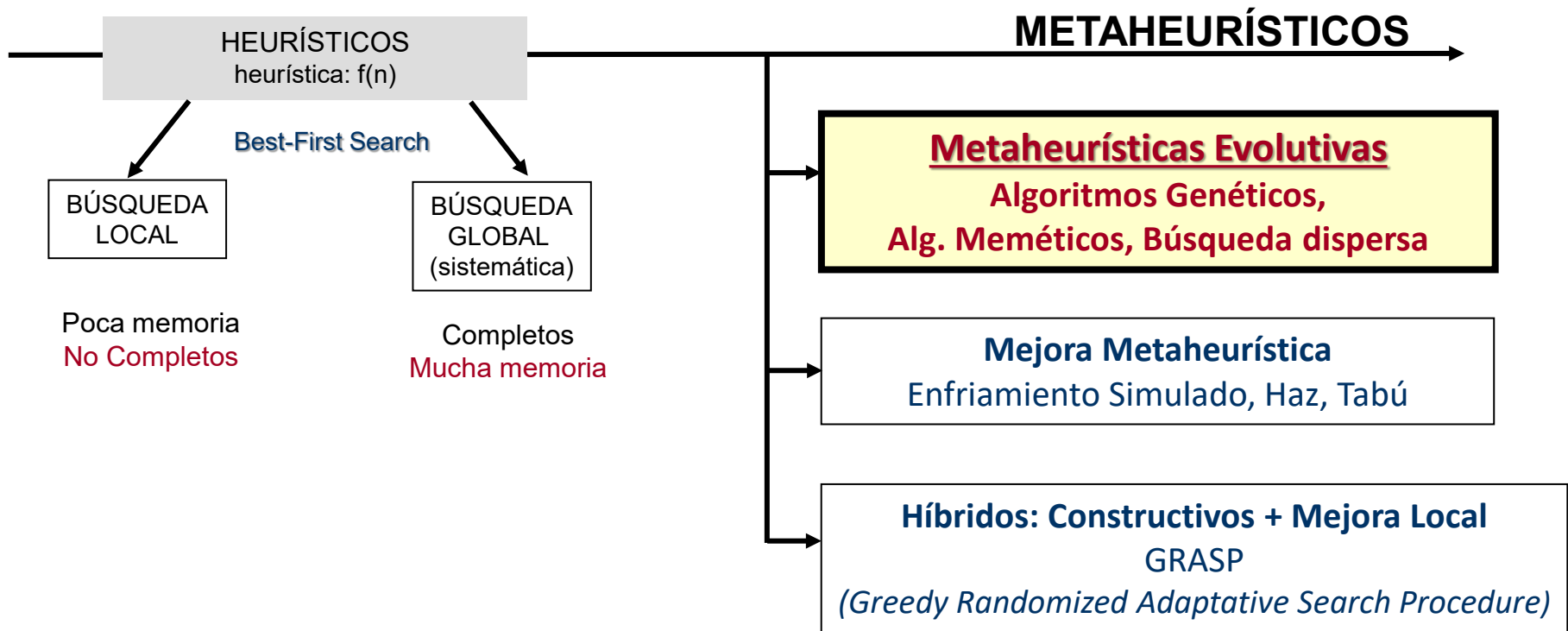
Prácticas

Aplicación metaheurísticas – **Algoritmos Genéticos** (2 semanas)

Bibliografía

(Artículos en Poliformat)

- **An Introduction to Metaheuristics for Optimization.** B. Chopard, M. Tomassini. Springer (2018). Cap. 8 & 9.
- **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, J. Marcos) (2003). <http://journal.iberamia.org/index.php/ia/article/view/360/article%20%281%29.pdf>
- **Genetic Algorithms: Advances in Research & Applications**, J. Carson. Nova Sc. Pub. 2017
- **Handbook of Evolutionay Computation**, T. Bäck, D.B. Fogel, Z. Michalewicz. Oxford University Press, 2007
- **Algoritmos genéticos con Python: un enfoque práctico para resolver problemas de ingeniería**, D. Gutiérrez, A. Tapia, A. Rodríguez. Marcombo, 2020
- <https://geneticprogramming.com/>



Principales características

- Parten de un **conjunto de individuos (población)** que se van reconstruyendo y/o mejorando
- Mejoran la búsqueda local, sin la complejidad (espacial) de una búsqueda global
- Existe una **colaboración** y **competición** entre individuos (soluciones) mediante las operaciones de **cruce** y **selección**. Tiene una componente **estocástica**
- Puede haber un **control centralizado** y los individuos no son autónomos (Alg. Genéticos), o descentralizado y con autonomía (Alg. Meméticos)
- Característica **any-time**: mejora sucesiva de las soluciones
- Implementación sencilla. Dificultad: **Diseño Individuo** y **Ajuste**

Algoritmos Genéticos

1. Conceptos Generales

2. Estructura de un Algoritmo Genético

3. Etapas de un Algoritmo Genético:

- Codificación en un AG
- Población Inicial
- Aptitud / fitness
- Estrategia de Selección
- Cruce, Mutación y Reemplazo de la Población
- Condición de Parada
- Otras variantes



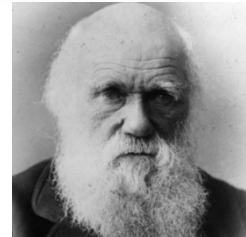
4. Algoritmos Genéticos Multiobjetivo



Ideados por John Holland (1975) inspirándose en la evolución natural de los seres vivos. *"Adaptation in Natural and Artificial Systems". University of Michigan Press, 1975*

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



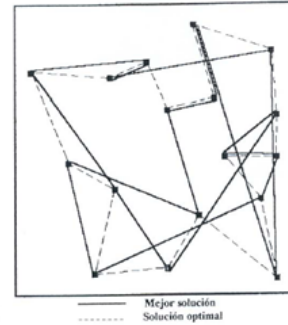
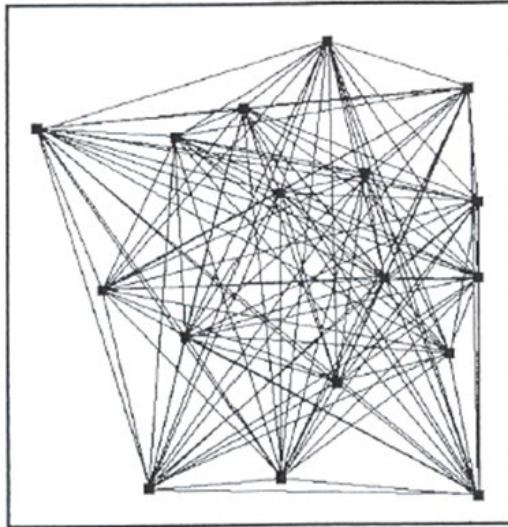
Los AG se inspiran en los procesos de Evolución Natural y Genética (métodos bioinspirados):

- 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*) basadas 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
- AG: mejora colectiva de individuos con control centralizado
- Son el método metaheurístico más aplicado y con más relevantes resultados prácticos

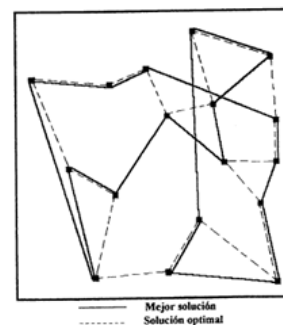
Aplicaciones: Optimización combinatoria en general

Ejemplo 1: Viajante de Comercio (Ciclo Hamiltoniano)

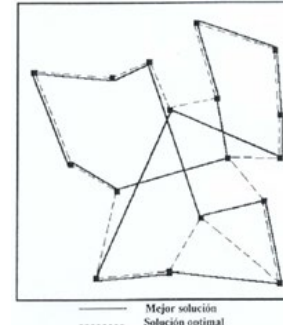
Generaciones de un AG →



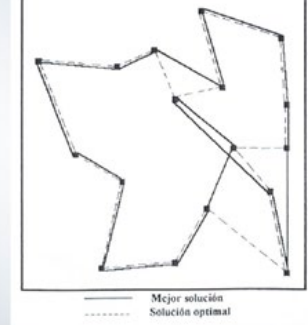
Iteración: 0 Costo: 403.7



Iteración: 25 Costo: 303.86



Iteración: 50 Costo: 293,6



Iteración: 100 Costo: 256,55

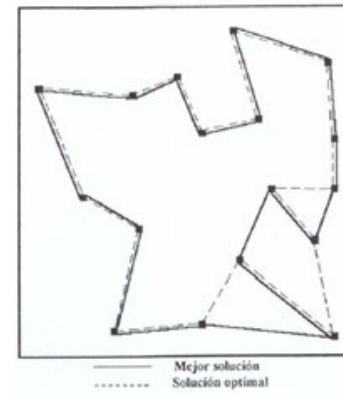
Con 17 ciudades

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

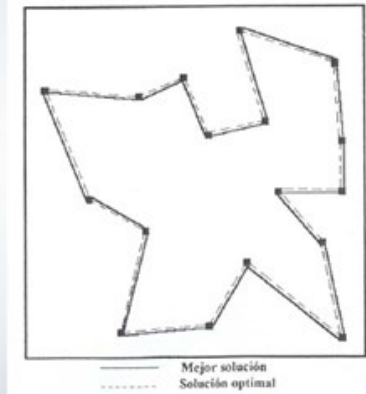
Con 1.000.000.000 recorridos/s, 4 días de cómputo

20 ciudades requerirían 77 años,

100 ciudades... 10^{135} millones de años

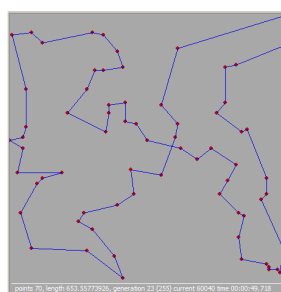
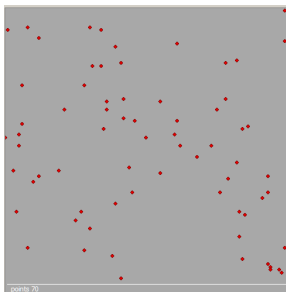


Iteración: 200 Costo: 231,4



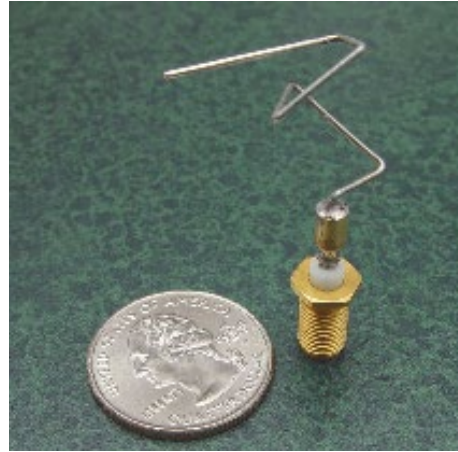
Iteración: 250 Solución
óptima: 226,64

Solución óptima: 226.64



Aplicación TSP: logística, configuración canales en sistemas de comunicación, etc.

Ejemplo 2: Diseño de antenas

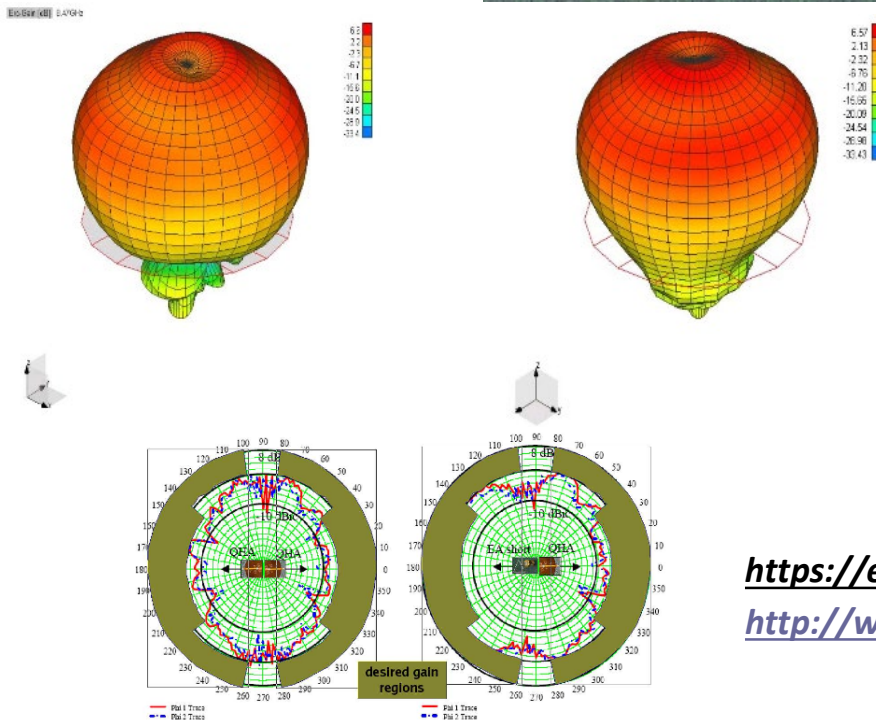


Diseño antenas Banda-X para su aplicación en satélites artificiales (NASA)

Evolved *X-band antenna*:

NASA ST5 spacecraft antenna.

This complicated shape was found by an evolutionary computer design program to create the best radiation pattern



Computer-automated evolution of an X-band antenna for NASA's Space Technology 5 mission

Hornby GS, Lohn JD, Linden DS.

Evol Computation 19(1):1-23

https://en.wikipedia.org/wiki/Evolved_antenna

http://www.nasa.gov/mission_pages/st-5/main/index.html

Ejemplo 3: Aprendizaje neuronal mediante AG

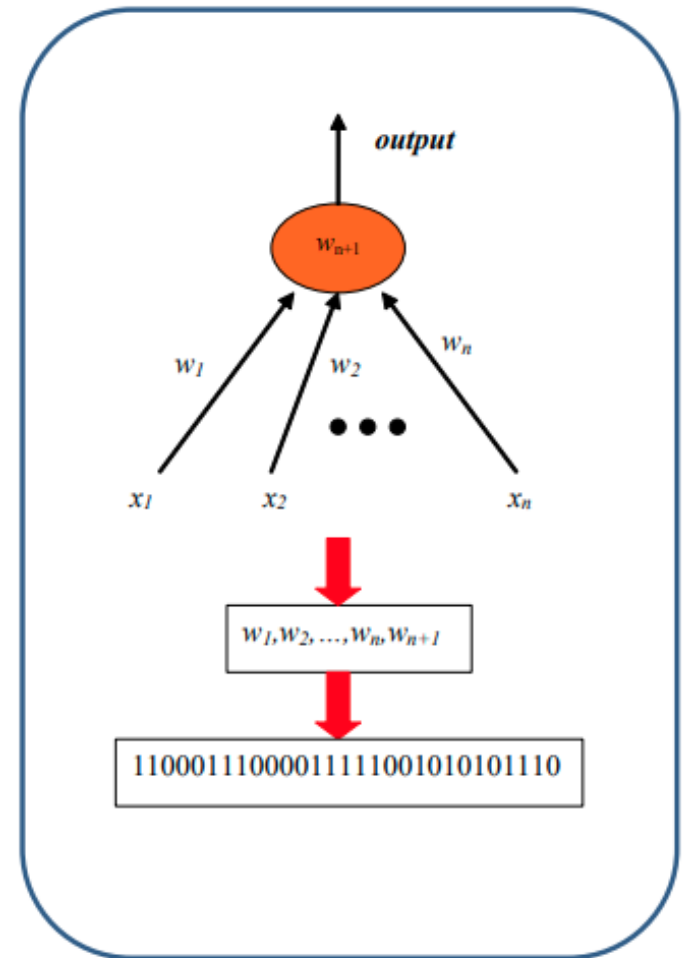
- A neuron is represented by its weights $\{w_1..w_n\}$ and the threshold $\{w_{n+1}\}$. The threshold is considered a special weight
- The genotype of a neuron can be defined as the weight list $\{w_1, \dots, w_{n+1}\}$. Each weight can be represented in a binary number
- Reconstruction of the phenotype from the genotype:

B : number of bits per weight

b_{ik} : k -th bit for the i -th weight

$$y_i = \sum_{k=1}^B b_{ik} 2^{-k}, \quad w_i = a[y_i] + b$$

where a and b are the scaling and shifting factors.



GA for deep learning (used for finding the “best structure” of a deep convolutional neural network)

- ***“An introduction to optimizacion: Genetic Algorithm”.*** Qiangfu Zhao
- ***“Fast and Unsupervised Neural Architecture Evolution for Visual Representation Learning,”*** Xue, Chen, Xie, Zhang, Gong and Doermann, in IEEE Comp. Intelligence Magazine, 16, 3, pp. 22-32, Aug. 2021.

Y más...

Genetic algorithms applied to the scheduling of the hubble space telescope *

Jeffrey L. Sponsler (a software systems engineer on the staff)

📄 Show more

[https://doi.org/10.1016/S0736-5853\(89\)80015-2](https://doi.org/10.1016/S0736-5853(89)80015-2)

Get rights and content

[Comput Math Methods Med.](#) 2018; 2018: 6154025.


Published online 2018 Jan 29. doi: [10.1155/2018/6154025](https://doi.org/10.1155/2018/6154025)

Prediction of Pathological Subjects Using Genetic Algorithms

[Murat Sari](#)  and [Can Tuna](#)

Applications of genetic algorithms from 2022

An Improved Genetic Algorithm for Path-Planning of Unmanned Surface Vehicle

Junfeng Xin ¹, Jiabao Zhong ¹, Fengru Yang ¹, Ying Cui ¹  and Jinlu Sheng ^{2,*}

¹ College of Electromechanical Engineering, Qingdao University of Science and Technology, Qingdao 266061, China; jfxin@163.com (J.X.); zhongjbchn@163.com (J.Z.); seawolf_yfr@163.com (F.Y.); cuiying@qust.edu.cn (Y.C.)

² Transport College, Chongqing Jiaotong University, Chongqing 400074, China


* Correspondence: seawolf_algorithm@163.com

Received: 9 April 2019; Accepted: 5 June 2019; Published: 11 June 2019



Genetic algorithms for computational materials discovery accelerated by machine learning

Paul C. Jennings, Steen Lysgaard, Jens Strabo Hummelshøj, Tejs Vegge  & Thomas Bligaard


npj Computational Materials **5**, Article number: 46 (2019) | [Download Citation](#) 

Genetic algorithms for lens design: a review

December 2018

DOI: 10.1007/s12596-018-0497-3

License · [CC BY 4.0](#)

 Kaspar Höschel ·  Vasudevan Lakshminarayanan

Mathematical Problems in Engineering
Volume 2018, Article ID 9270802, 32 pages
<https://doi.org/10.1155/2018/9270802>

Review Article

Recent Research Trends in Genetic Algorithm Based Flexible Job Shop Scheduling Problems

¿Qué necesitamos para diseñar un AG?

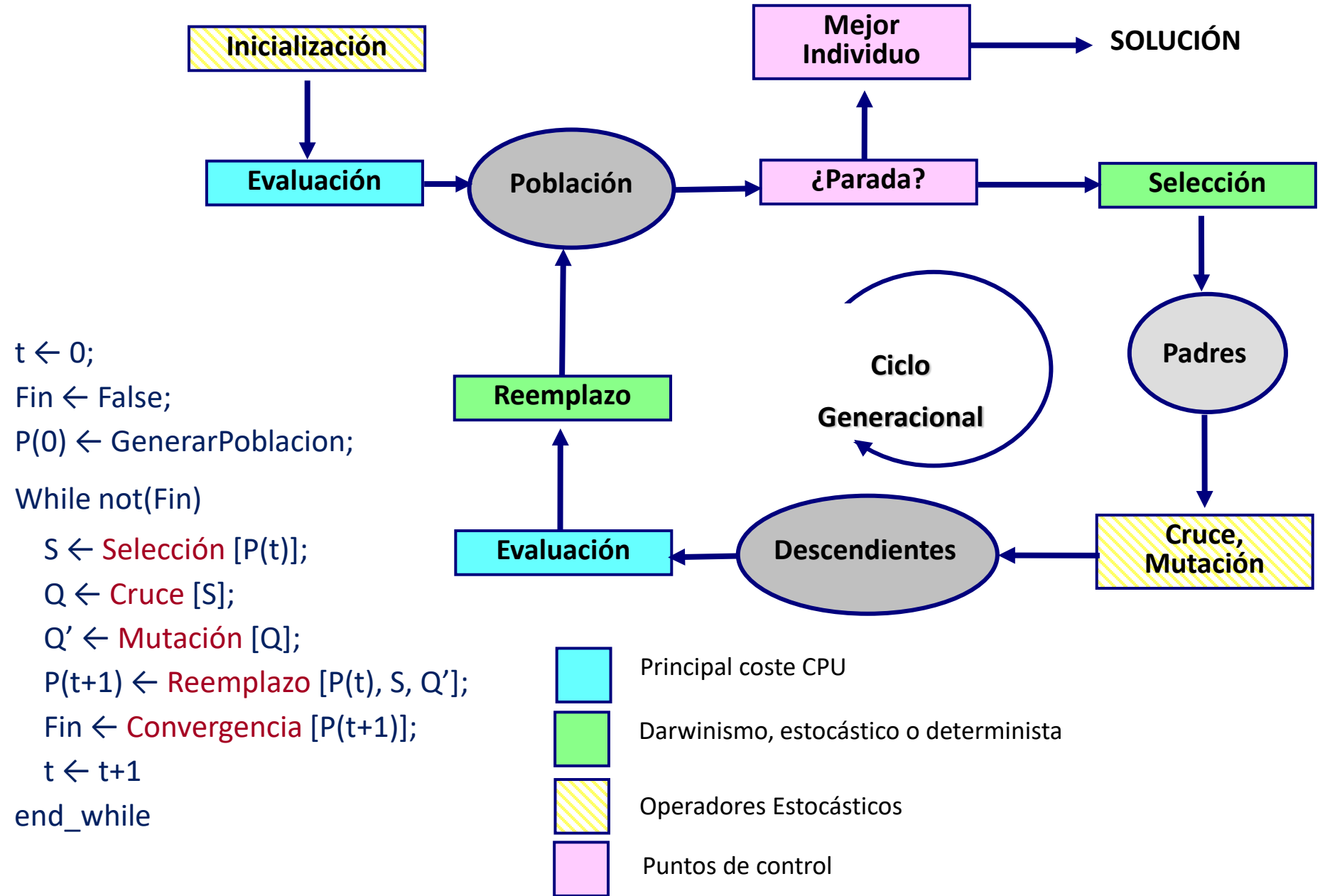
Población de Individuos \Leftrightarrow Conjunto de Soluciones

Evolución \Leftrightarrow Búsqueda

Mejor individuo \Leftrightarrow Solución Final

1. Una **representación** adecuada de las soluciones del problema (individuos). Típicamente binarias (pero no necesariamente). **Cromosomas, genes, genotipo, fenotipo**
2. Una forma de crear una **población de soluciones inicial** (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 'entorno' 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 de entrada**: tamaño de la población, número de iteraciones (generaciones), probabilidades de selección, condición de parada, etc.

Estructura de un AG



```
t ← 0;  
Fin ← False;  
P(0) ← GenerarPoblacion;  
While not(Fin)  
  S ← Selección [P(t)];  
  Q ← Cruce [S];  
  Q' ← Mutación [Q];  
  P(t+1) ← Reemplazo [P(t), S, Q'];  
  Fin ← Convergencia [P(t+1)];  
  t ← t+1  
end_while
```

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).

2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.

3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

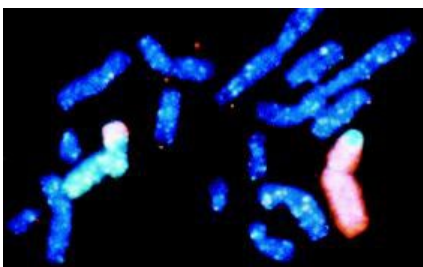
[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud $f(x)$** .

[Ciclo] Ir al paso 2

Codificación (representación) del individuo: una solución del problema

Evolución Genética

Individuo \equiv 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

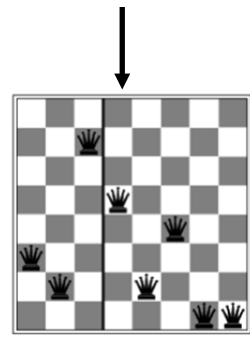
Individuo \equiv Cromosoma: estructura de datos que contiene una **secuencia de parámetros** que permite definir (o formar) una **solución** al problema.

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

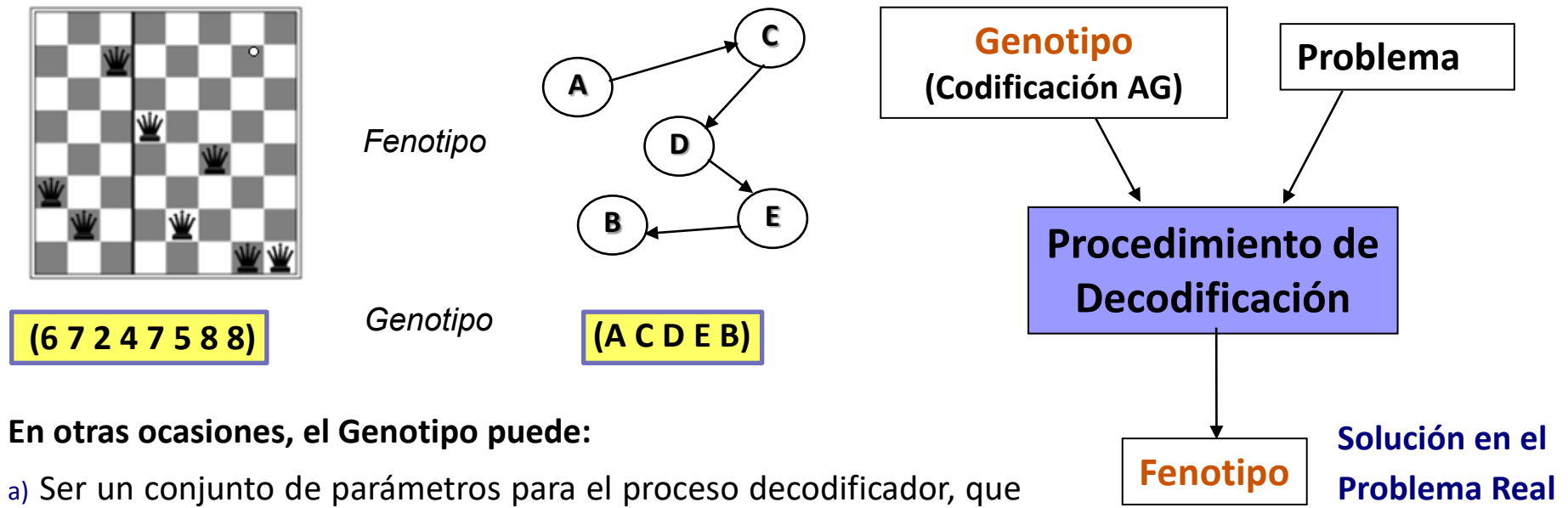
¡Es muy importante el diseño de los individuos (cada solución)!

Codificación de Soluciones en un AG: Solución \Leftrightarrow 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.

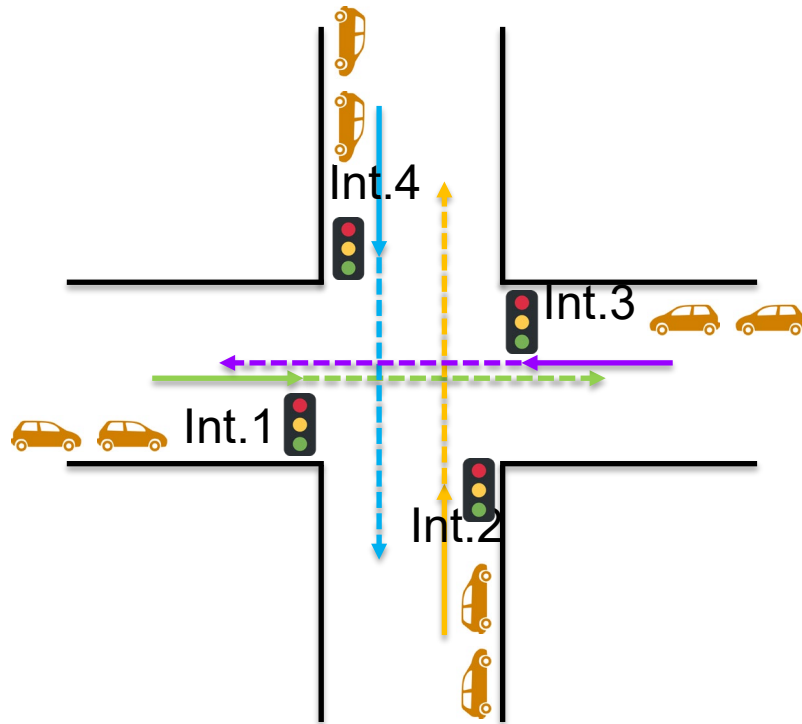


En otras ocasiones, el Genotipo puede:

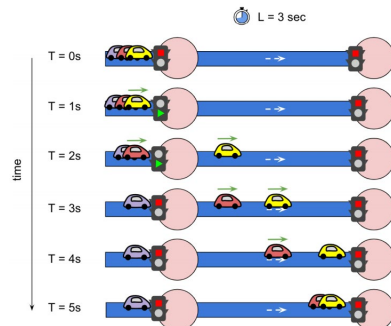
- a) Ser un conjunto de parámetros para el proceso decodificador, que sobre los datos de un problema, obtendrá el fenotipo correspondiente.
- b) Tener una estructura no linearizada (árbol, <atributo-valor>, etc.)

Ejercicio. Planificar 4 semáforos en una intersección de 4 calles.

Codificar/representar un individuo



- Solo nos interesa modelar el color **rojo/verde** de cada semáforo
- Los vehículos **siempre** se mueven en línea recta (no hay giros)
- Intersecciones **compatibles en verde**: {Int.1, Int.3}, {Int.2, Int.4}
- Intersecciones **incompatibles en verde**: {Int.1, Int.2}, {Int.1, Int.4}, {Int.2, Int.3}, {Int.3, Int.4}, porque los vehículos se chocarían
- Hay **100 fases/ticks/pasos de tiempo** (cada fase representa un período temporal, ej. 15 s, 30 s, 60 s, etc.)
- Si en una **fase** un semáforo está en **verde**, solo pueden pasar **10 vehículos** por él como máximo



[Google](#)
[hashcode 2021](#)

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).

2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.

3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud** $f(x)$.

[Ciclo] Ir al paso 2

Método inicialización:

- 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
- b) Elegir la población inicial a partir de los resultados de una **heurística** previa
 - Posible pérdida de la diversidad genética
 - Posible preferencia no recuperable



Tamaño de la población. Típicamente dependiente del problema

- a) Hay propuestas teóricas que proponen que el tamaño óptimo para individuos de longitud L , con codificación binaria, debe crecer exponencialmente con L
- b) Otros resultados empíricos sugieren un tamaño de población entre L y $2 \cdot L$
- c) **¿Es mejor un mayor tamaño en la población o una mayor diversidad?**

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).

2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.

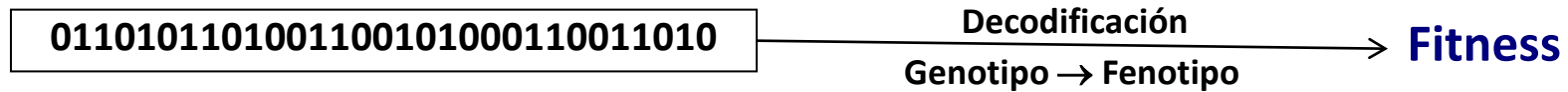
3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud $f(x)$** .

[Ciclo] Ir al paso 2



- **Evalúa la calidad** del individuo (solución) o cromosoma
- Este es el **paso más costoso** en las aplicaciones reales
 - Requiere obtener el fenotipo (solución) del genotipo
 - 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? Como en la naturaleza misma...
 - Si es un problema de **factibilidad**: típicamente se penaliza el fitness de la solución (ejemplo: 8-reinas) o se desecha el individuo
 - Si es un problema de **optimalidad**: típicamente, se repara la solución (ejemplo: viajante de comercio). Si no es posible, el individuo se desecha

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).

2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.

3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud $f(x)$** .

[Ciclo] Ir al paso 2

Estrategia de selección de individuos a reproducir en cada generación

- Debemos garantizar que los mejores individuos tengan una mayor posibilidad de ser padres (reproducirse) frente a los individuos menos buenos
Cuanto 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. Estos 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** (ver Reemplazo: *generacional, estacionario, gap generacional*)

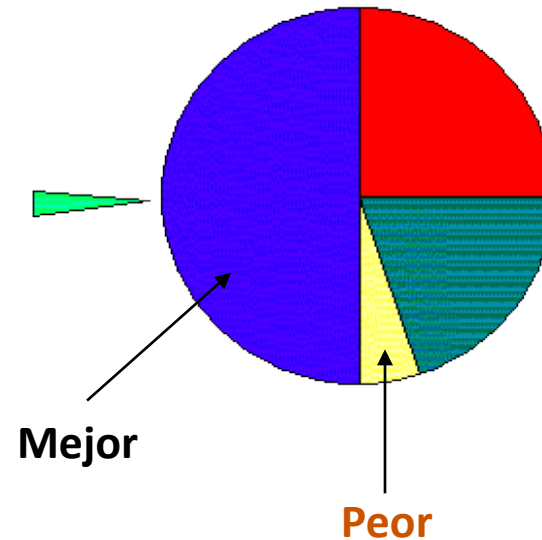
- **Selección elitista:** elección directa de los miembros más aptos de cada generación
- **Selección proporcional:** los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza. Los peores individuos también pueden ser elegidos
 - Selección por rueda de ruleta:** basado en el fitness del individuo y la media de grupo
- **Selección jerárquica:** los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias (hay muchos individuos), mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente
 - Selección por torneo:** grupos de individuos compiten entre ellos, sucesivamente
- **Selección por rango:** más que por el valor del fitness, la selección se basa en el rango (posición) de cada individuo. A cada individuo se le asigna un ranking numérico 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 multiobjetivo** (basado en frentes de Pareto, diversidad de soluciones, etc.): NSGA-II: (Non-dominated Sorting Genetic Algorithm II, Dec'2002)

Selección proporcional. Selección por rueda de ruleta

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

A cada individuo de la población se le asigna una parte proporcional a su fitness f_i 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

Pero todos los individuos pueden ser potencialmente elegidos

Ejemplo Selección por Rueda de Ruleta

SUPERINDIVIDUO

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

Individuo

Adaptación

Prob. De Selección

Nº

f_i

$p_i = f_i / \sum f_i$

1

200

0.797

2

30

0.119

3

10

0.040

4

8

0.032

5

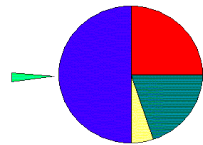
2

0.008

6

1

0.004

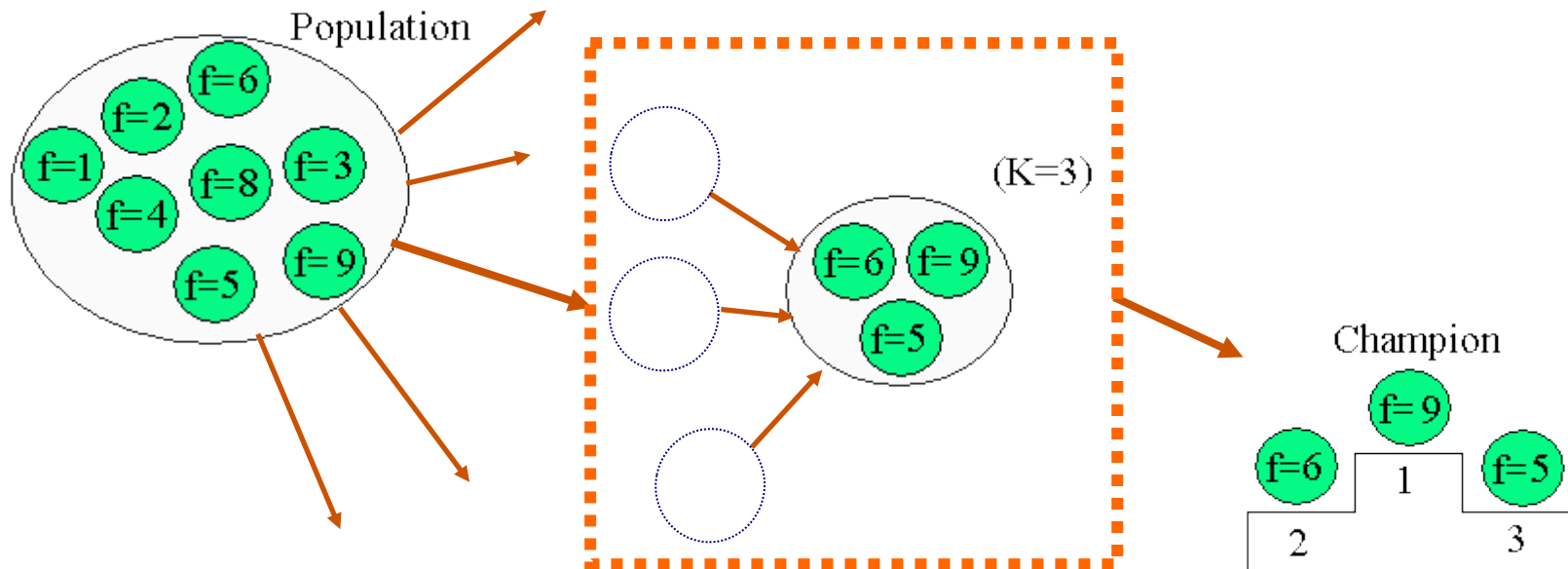


Inconvenientes:

- 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**
- **Modificación** (*windowing*). Aplicar $f'(i) = f(i) - \beta(i)$, donde $\beta(i)$ =peor fitness en cada generación

Selección jerárquica. 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 ($K \uparrow$), la presión de selección es elevada y los peores individuos apenas tienen oportunidades de reproducción
- Cuando el tamaño del torneo es reducido ($K \downarrow$), 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:

$\max + \min = 2$

$\max \geq \min$: $\max \in [1, 2]$, $\min \in [0, 1]$,

(\max recomendado = 1.1)

$\uparrow \max \Rightarrow$ Mayor Separación $P(x)$

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

Ejemplo ($\max = 1.1 \rightarrow \min = 0.9$):

<i>Fitness(x)</i>	<i>Ranking(x)</i>	<i>h(x)</i>	<i>P(x)</i>		<i>P(x)</i> <i>max=1</i>	<i>P(x)</i> <i>max=2</i>	<i>Propor</i> <i>cional</i>
5	2	1	0.33	} Aplicar Ruleta u otra técnica Proporcional	0.33	0.33	0.19
2	3	0.9	0.30		0.33	0.00	0.08
19	1	1.1	0.37		0.33	0.67	0.73
Σ		3	1				

Otros esquemas:

Ranking Lineal, Ranking exponencial, etc.

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).

2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.

3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud $f(x)$** .

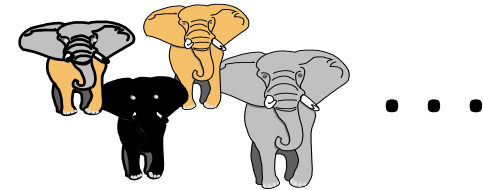
[Ciclo] Ir al paso 2

Tipos de cruce

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

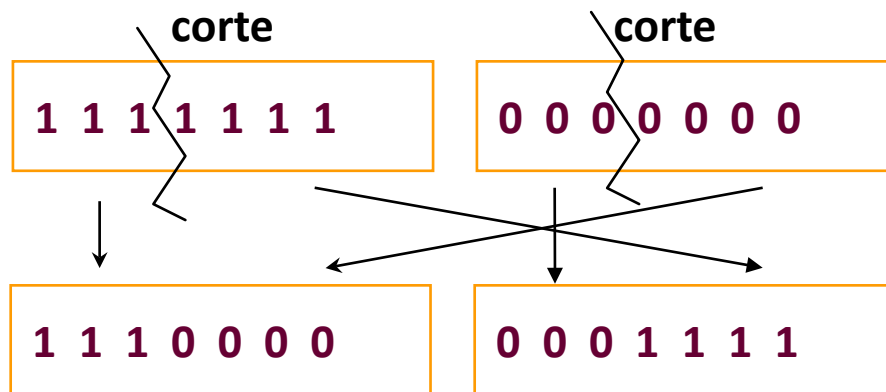
En la computación evolutiva: se simula la recombinación intercambiando segmentos de cadenas lineales de longitud fija (los cromosomas)

Población:

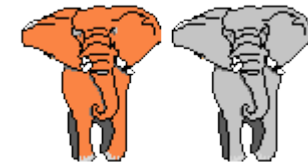


Cruce de n-puntos

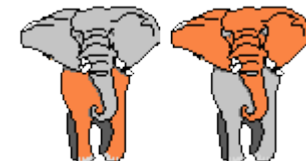
Cada cromosoma es cortado en **n puntos** y las porciones de cada cromosoma se recombinan (Ejemplo para **$n=1$**)



padres



hijos



Cruce de 2 puntos:

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

Padre1 = 10|1010|1010

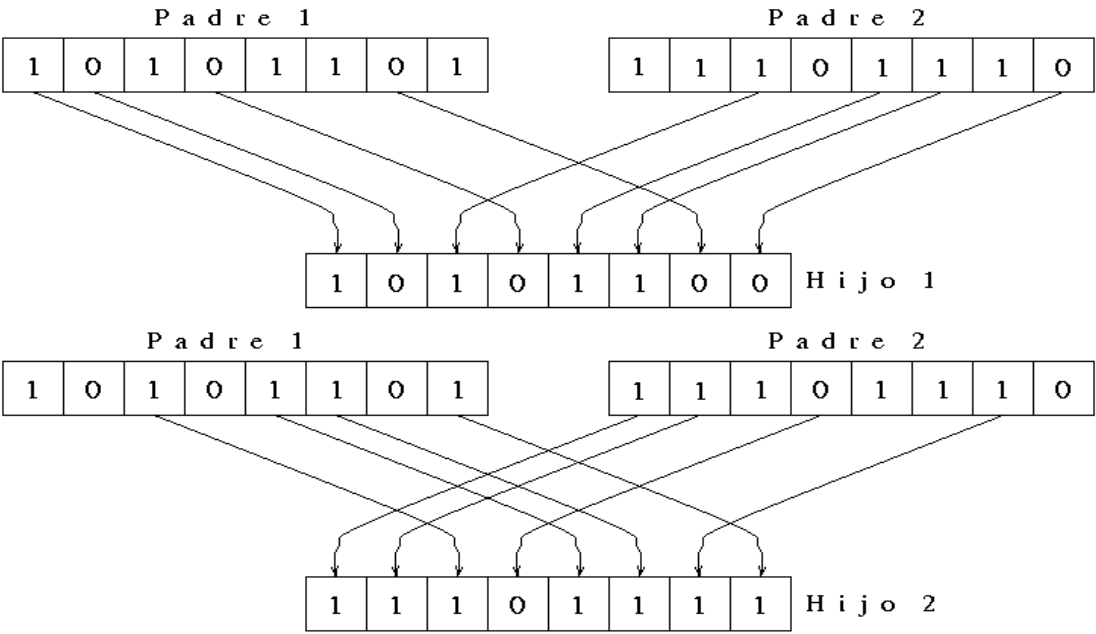
Padre2 = 11|1000|1110

Hijo1 = 11|1010|1110

Hijo2 = 10|1000|1010

Cruce Uniforme:

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



Recomendaciones en la literatura sobre puntos de cruce

- El cruce de n -puntos es el método más típico, si no existen restricciones adicionales
- El cruce de un punto ($n=1$) proporciona poca flexibilidad
- No existe consenso sobre uso de valores para $n \geq 3$
- Incrementar el valor de n se asocia con un mayor efecto destructivo del cruce
- En general, la **recombinación de 2 puntos** y la **recombinación uniforme** son generalmente preferibles a la recombinación de 1 punto
- La elección más acertada dependerá del problema a resolver

Existen otros diversos métodos de cruce...

Ejemplo: Cruce de Permutación (alelos acotados):

- Se selecciona un conjunto de genes consecutivos del Padre-1
- Se trasladan al Hijo-1 y se marcan los genes en el Padre-2
- Comenzando en el lado derecho del cromosoma, trasladar los genes del padre-2 (que no estén ya) al hijo-1
- Para el Hijo-2, repetir el proceso, invirtiendo previamente los padres

Padre-1: 8 4 7 3 6 2 5 1 9 0

Padre-2: 0 ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~ 7 8 9

Hijo-1: 0 4 7 3 6 2 5 1 8 9

Padre-1: 0 9 1 5 2 6 3 7 4 8

Padre-2: 9 8 ~~7~~ ~~6~~ ~~5~~ 4 ~~3~~ ~~2~~ 1 0

Hijo-1: 9 8 4 5 2 6 3 7 1 0

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).

2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.

3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud** $f(x)$.

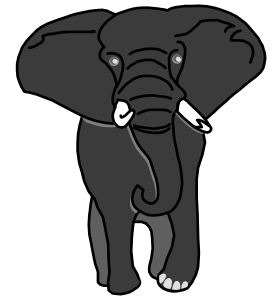
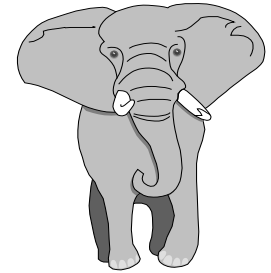
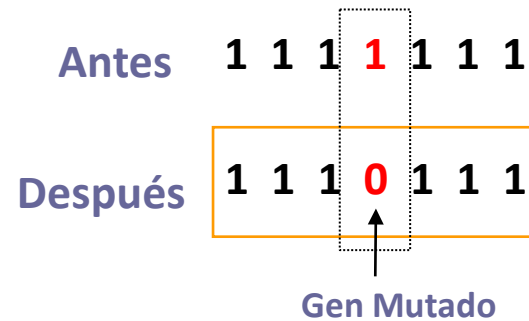
[Ciclo] Ir al paso 2

Mutación como alternativa para introducir diversidad en la búsqueda

Mutación bit-a-bit

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

La probabilidad de mutar (P_{mut}) en general es baja, aprox. 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.

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

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

Ejemplo:

elegimos la posición 7 y movemos ese valor a la posición 2

Mutación por Intercambio Recíproco:

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

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

$\wedge \longleftrightarrow \wedge$

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

Otras alternativas de mutación:

- Inversión (de un grupo de genes)
- Scramble (reordenación aleatoria de un subconjunto) , etc.

Ejemplo de AG. Problema del viajante de comercio

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

Genotipo: *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 3 1 4)

Cruce (2-puntos):

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

Cromosoma-j (**5 8** 7 6 8 3 **1 4**)

Hijo k (**5 8 7 2 1 6** **3** **4**)

reparación

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 = $-\infty$

Mutación (intercambio):

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

Después: (5 8 **6** 2 1 **7** 3 4)

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).
2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo

4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud $f(x)$** .

[Ciclo] Ir al paso 2

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**

- En cada generación se generan pocos hijos (1, 2, 3,...) y solo pocos individuos (los peores) de la población son reemplazados

□ **Gap Generacional (*Generacional Elitista*)**

- 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) o Proporción = $1/|población|$ para SSAG

□ **Reemplazo catastrofista:** Cada cierto tiempo se eliminan 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

0. Diseño. Codificación y Decodificación de los individuos (soluciones)

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 (p_c).
2. **[Cruce]** Combinar los genes de los dos padres para obtener hijos.
3. **[Mutación]** Con una probabilidad de mutación (p_{mut}), mutar cada gen de los cromosomas hijo
4. **[Reemplazo]** Añadir nuevos hijos y determinar nueva población.

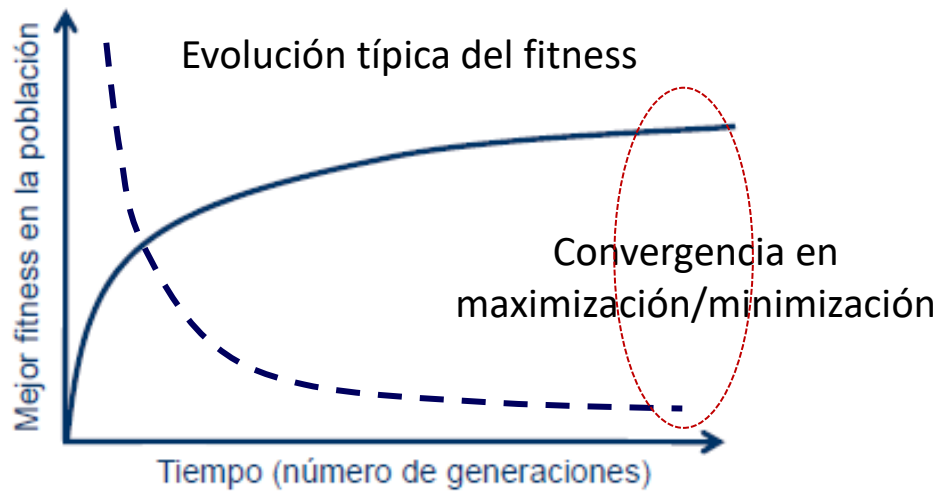
4. **[Verificar]** Condición de parada:

[Parada], proporcionar como solución el **cromosoma con mejor valor de aptitud $f(x)$** .

[Ciclo] Ir al paso 2

Condición de parada. Múltiples alternativas

- 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**



Los algoritmos genéticos pueden implementarse en paralelo de manera natural

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 más 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

Más
variantes...

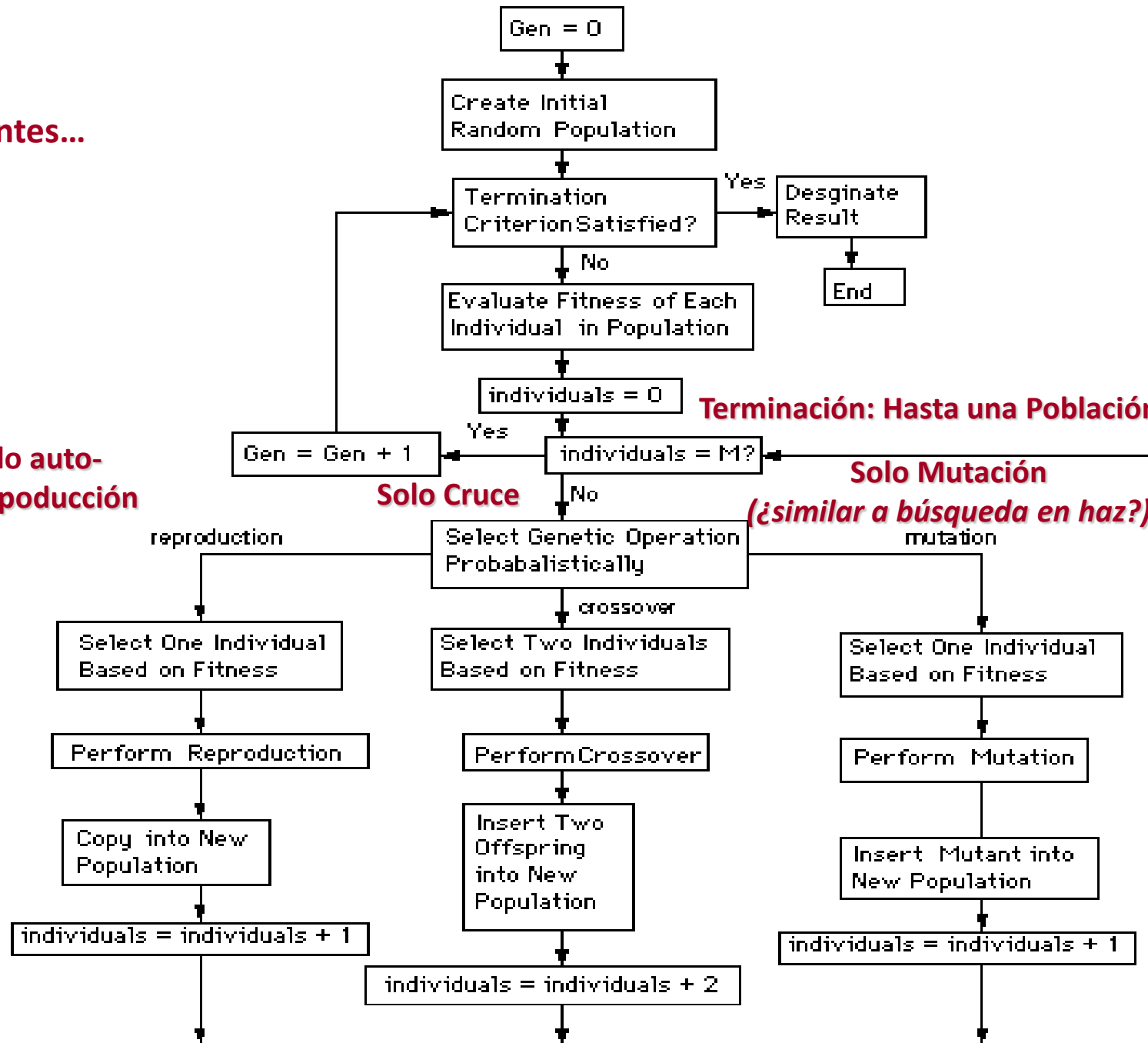
Solo auto-
Reproducción

Solo Cruce

Terminación: Hasta una Población M

Solo Mutación

(¿similar a búsqueda en haz?)



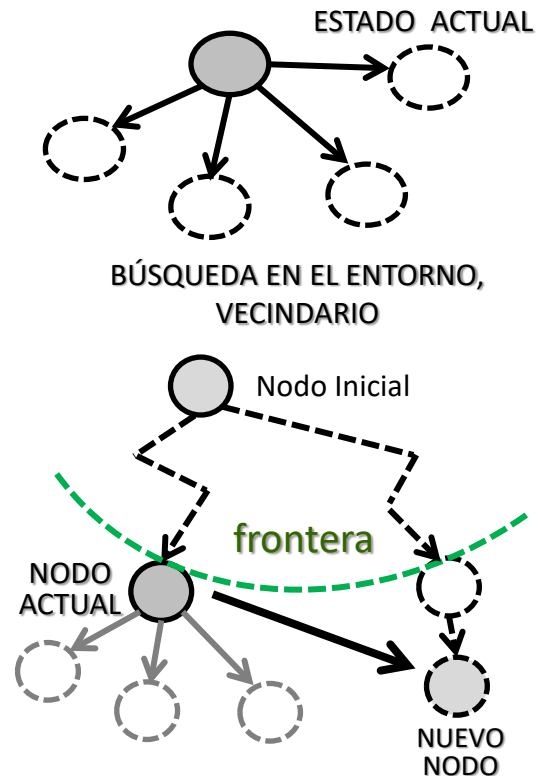
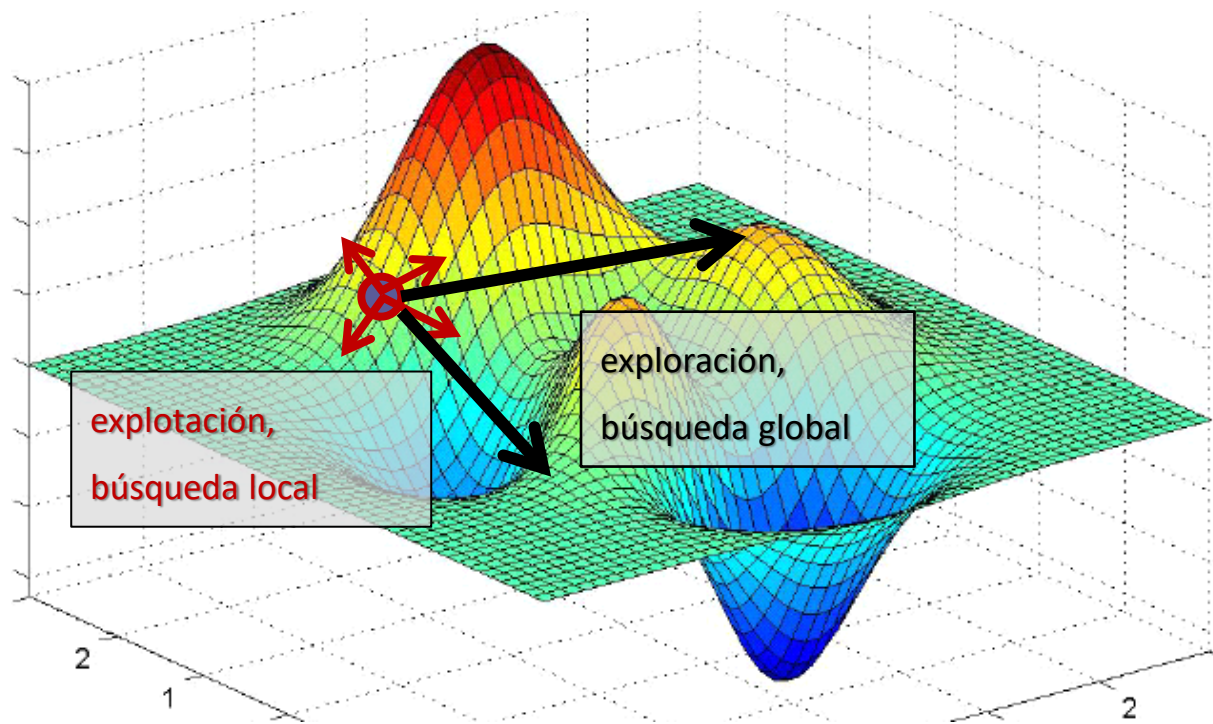
- **Utilidad relativa de las herramientas genéricas para desarrollar AG concretos:**
 - La codificación/decodificación/fitness suelen ser muy dependientes del problema (suponer un individuo como una secuencia de bits sin estructura no suele bastar)
 - Las operaciones (cruce, mutación, etc.) son dependientes de la codificación de los individuos, etc.
 - El proceso 'genérico' del AG es muy simple
- **Entornos libres (o demos)**
 - Opt4J (<https://sdarg.github.io/opt4j/>). Formulación sencilla utilizando librerías implementadas en Java.
 - HeuristicLab <http://dev.heuristiclab.com/> Entorno generalista; original método de programación.
 - PARADISEO <https://nojhan.github.io/paradiseo/> API de programación en C++
 - <http://www.rubicite.com/Tutorials/GeneticAlgorithms.aspx> ,
 - http://people.sc.fsu.edu/~jburkardt/cpp_src/simple_ga/simple_ga.html
 - <http://www.aridolan.com/ga/gaa/gaa.html>
 - <https://geneticprogramming.com/software>
- **Entornos comerciales:** MATLAB (*Toolbox libres sobre AG*). MATLAB & Global Optimization Toolbox
- **Librerías que codifican código generalista en los AG:**
 - En entornos generales: Matlab, Mathematica, GoldenBerry-GA, etc.
 - En **java** (JGAP, JCLEC, ECJ, Opt4J, OSGiLiath), **C++** (Open BEAGLE, MALLBA, Evolving Objects-EO, GALIB), **Python** (DEAP) etc.

Estas herramientas pueden ayudar, pero la verdadera complejidad es el diseño de los individuos y semántica de las operaciones (cruce, mutación) en cada problema concreto

Exploración vs. Explotación

En el proceso de búsqueda de buenas soluciones, existe un equilibrio entre los conceptos de **Exploración vs. Explotación**:

- Buscar mejores soluciones en zonas lejanas desconocidas (exploración, búsqueda global)
- Buscar mejores soluciones alrededor de la solución actual (explotación, búsqueda local)



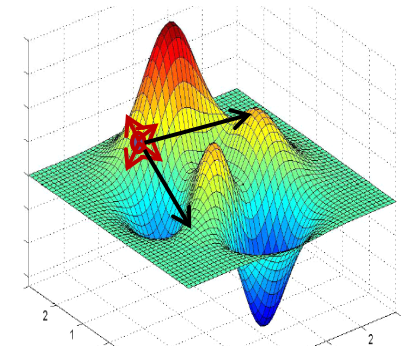
Teoría de la Selección r/K , donde las fuerzas evolutivas tienen dos estrategias diferentes:

- **Estrategia r** : muchos descendientes, con baja probabilidad de supervivencia
- **Estrategia K** : se invierten grandes recursos en pocos descendientes, con alta probabilidad de supervivencia

Exploración vs. Explotación en AG

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 se incrementa 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 y diversificación**
 - 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, memoria, intensificación, explotación**
 - Relacionado con la exploración/explotación:
mayor nº puntos de cruce \Rightarrow mayor exploración. Menor nº puntos \Rightarrow mayor explotación
- Selección (*fitness*): **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)

Cruce vs. Mutación

- Depende del problema pero, en general, es mejor utilizar ambos operadores (se podría utilizar solo mutación, pero no solo cruce)
- El cruce permite explorar áreas lejanas de los padres (combinando lo mejor de ellos)
- La mutación explora áreas cercanas al individuo que muta

Selección

- Es necesaria la selección por la limitación de los recursos (*similar a la naturaleza*)
- Se debe garantizar la probabilidad de combinación de todos los individuos. Pero siempre debe haber un cierto elitismo

Reemplazo

- Típicamente, la selección se aplica para seleccionar padres. ¿Y para el reemplazo (*no siempre los peores*)? \Rightarrow Reemplazo Proporcional
- Reemplazo basado en Fitness vs. basado en la Antigüedad

Un problema de optimización **mono-objetivo** se puede formular como obtener una solución al problema, valor de las variables $X = \{x_1, x_2, \dots, x_n\}$, tal que:

$$\min/\max [f_{\text{objetivo}}(X)],$$

y tal que se cumpla el conjunto de restricciones del problema.

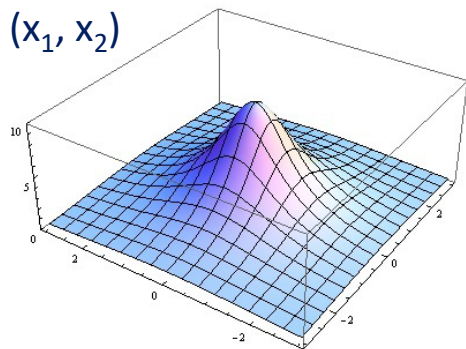
Ejemplos: *Problema del Viajante, donde (x_1, x_2, \dots, x_n) representaría la secuencia de ciudades, Problema de la Mochila, donde (x_1, x_2, \dots, x_n) representaría la inclusión (o no) del objeto Planificación clásica de acciones, etc.*

Un problema de optimización **multi-objetivo** se puede formular como la maximización/minimización de una función vectorial:

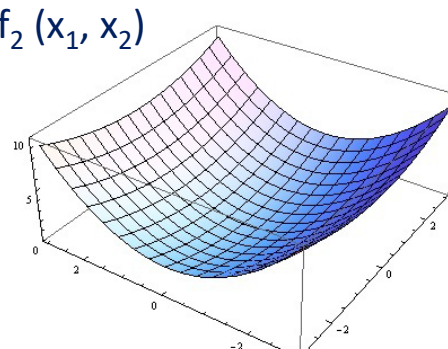
$$\min/\max [f_1(X), f_2(X), \dots, f_m(X)],$$

tal que se quiere maximizar/minimizar *simultáneamente* un conjunto de funciones

$f_1(x_1, x_2)$



$f_2(x_1, x_2)$



a) Combinación ponderada de objetivos (pesos w_i): se convierte en problema mono-objetivo

$$\min \sum_{i=1,m} w_i f_i(X)$$

Ejemplo (viajante multiobjetivo):

$$\text{Min } (w_1 \text{ Coste } (x_1, x_2, \dots, x_n) + w_2 \text{ Distancia } (x_1, x_2, \dots, x_n) + w_3 \text{ Tiempo } (x_1, x_2, \dots, x_n))$$

Ventajas: Sencillez, *Proyección* a problema mono-objetivo

Inconvenientes: Diferencia en valor/unidades de los objetivos (requiere normalización)

No realiza una verdadera optimización multiobjetivo

Solo ofrece una solución

Devuelve: Una única solución con el valor objetivo *proyectado*

b) Métodos basados en la Eficiencia de Pareto: $\min/\max [f_1(X), f_2(X), \dots, f_m(X)]$

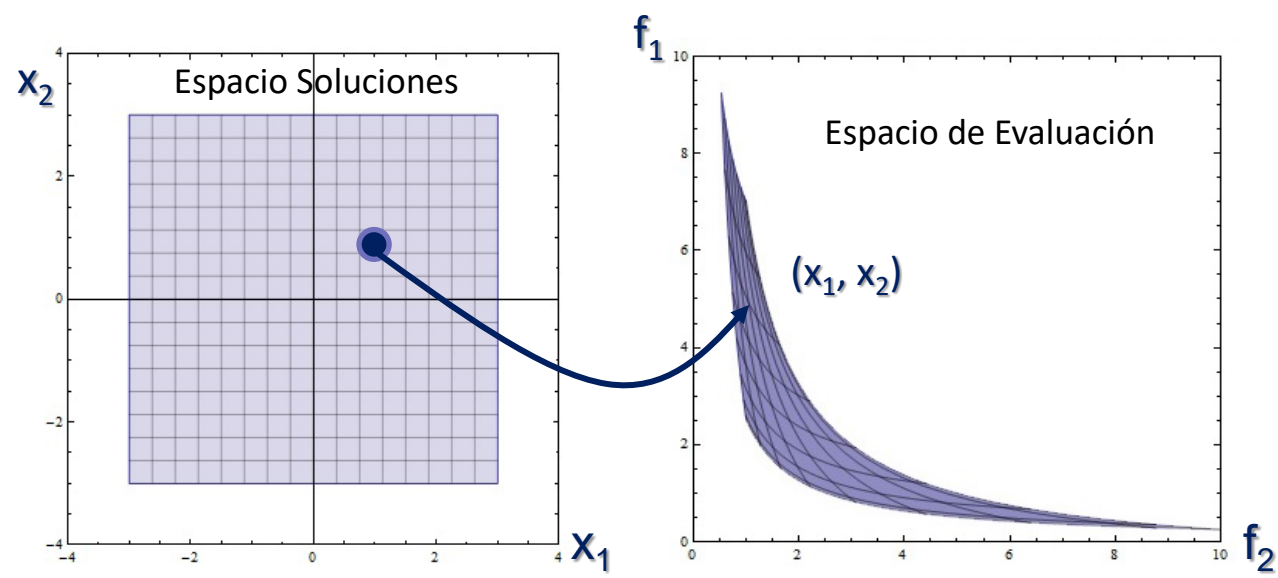
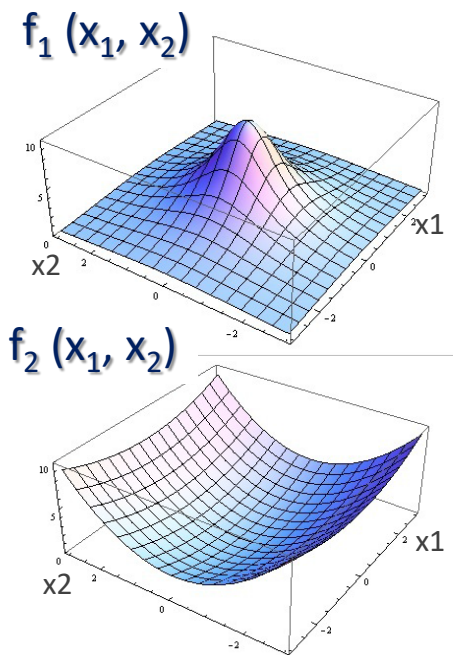
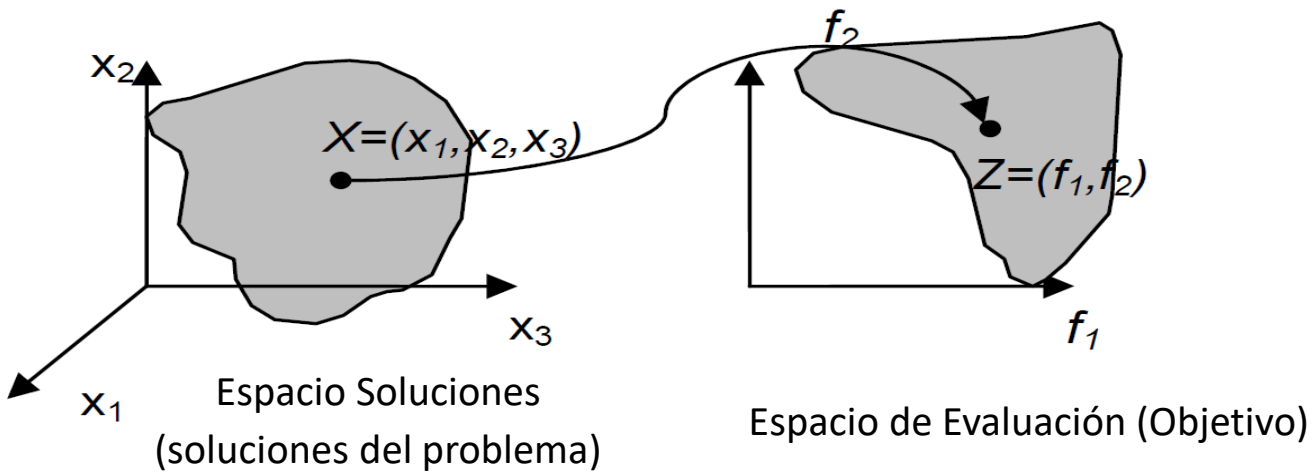
Ventajas: Trata realmente un problema multi-objetivo

Inconvenientes: Mayor complejidad

Devuelve: Un conjunto de soluciones (frontera de Pareto o puntos cercanos a ella)

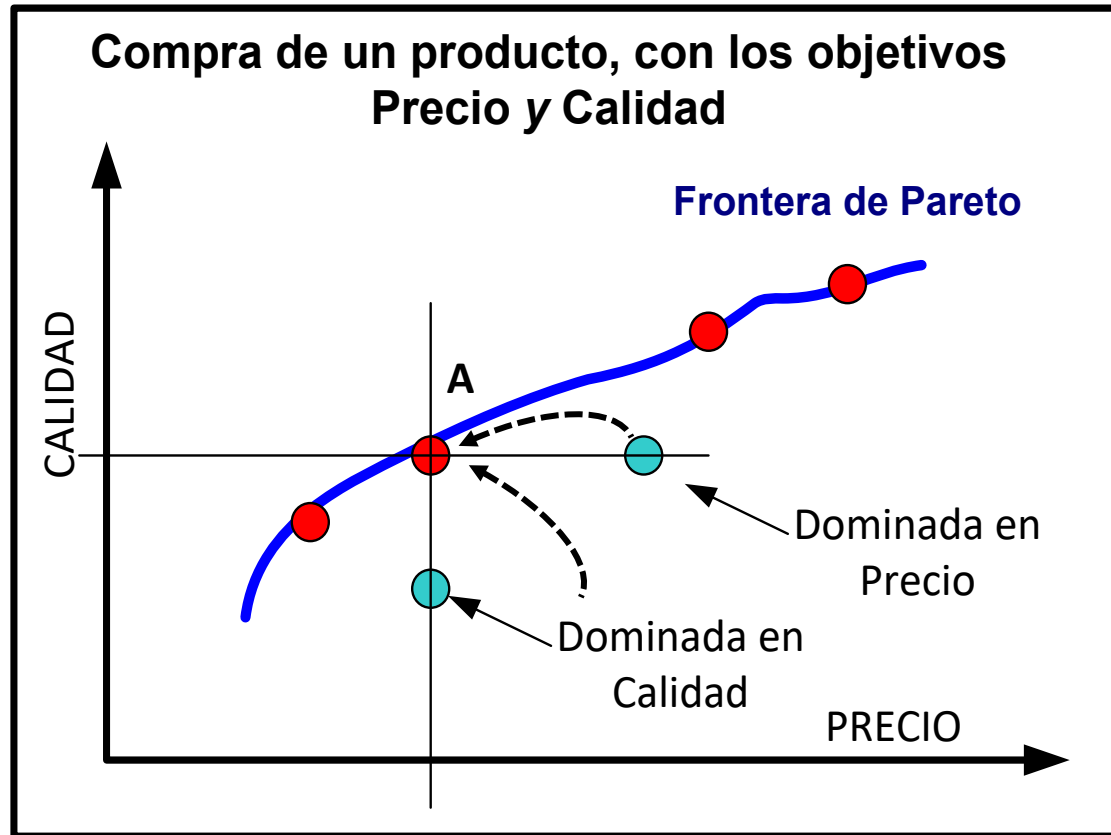
Espacio de Soluciones y Espacio de Evaluación

- El Espacio de Soluciones representa todas las combinaciones posibles en el valor de las variables
- El Espacio de Evaluación (o Espacio Objetivo) representa el valor de los objetivos del problema



Frente (o Frontera) de Pareto (soluciones de la optimización multiobjetivo)

En problemas multi-objetivo, la mejora de un objetivo habitualmente empeora otro



Dominancia de Soluciones: Una solución X_1 domina a otra solución X_2 , si:

- X_1 es mejor que X_2 en al menos un objetivo, y
 - X_1 no es peor que X_2 en todos los objetivos
- \Rightarrow Es preferible X_1 a X_2

Aplicándolo iterativamente se obtiene el Conjunto de Soluciones Dominantes:

Conjunto de soluciones dominantes (*soluciones no dominadas*) \Rightarrow Frontera de Pareto

Conjunto de soluciones de la Frontera de Pareto:

- Soluciones que no pueden mejorar uno de los objetivos sin empeorar algún otro.
- Estas soluciones determinan las mejores alternativas para la solución del problema.

Formalizando:

Una solución $X_i = (v_{i1}, v_{i2}, \dots, v_{in})$ **domina** a otra solución $X_j = (v_{j1}, v_{j2}, \dots, v_{jn})$, si y solo si:

$$\exists f_p \in \{f_1, f_2, \dots, f_m\}, f_p(X_i) < f_p(X_j) \quad \text{y} \quad \forall f_k \in \{f_1, f_2, \dots, f_m\}, f_k(X_i) \leq f_k(X_j)$$

Una solución X^* es **Pareto-óptima** si y solo si no existe otra solución X' que la domine: conjunto de soluciones dominantes

En caso de maximización, el cambio es trivial

Dominancia de Soluciones: Una solución X_1 domina a otra solución X_2 , si:

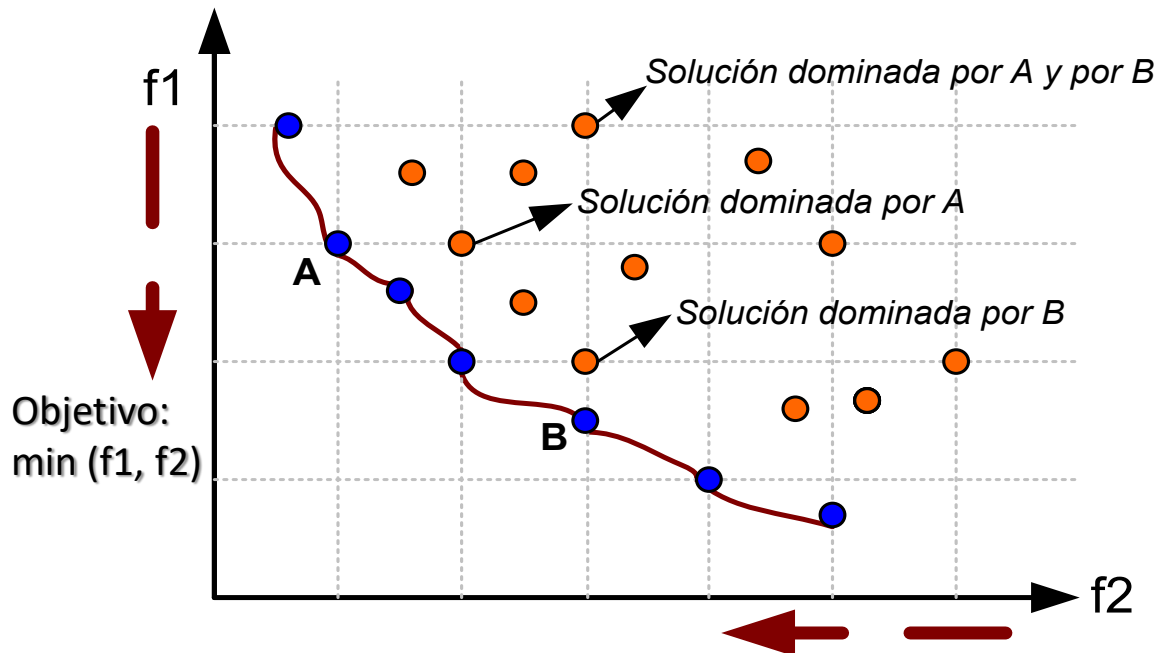
- X_1 es mejor que X_2 en al menos un objetivo, y
 - X_1 no es peor que X_2 en todos los objetivos
- \Rightarrow Es preferible X_1 a X_2

Aplicándolo iterativamente se obtiene el Conjunto de Soluciones Dominantes:

Conjunto de soluciones dominantes (*soluciones no dominadas*) \Rightarrow Frontera de Pareto

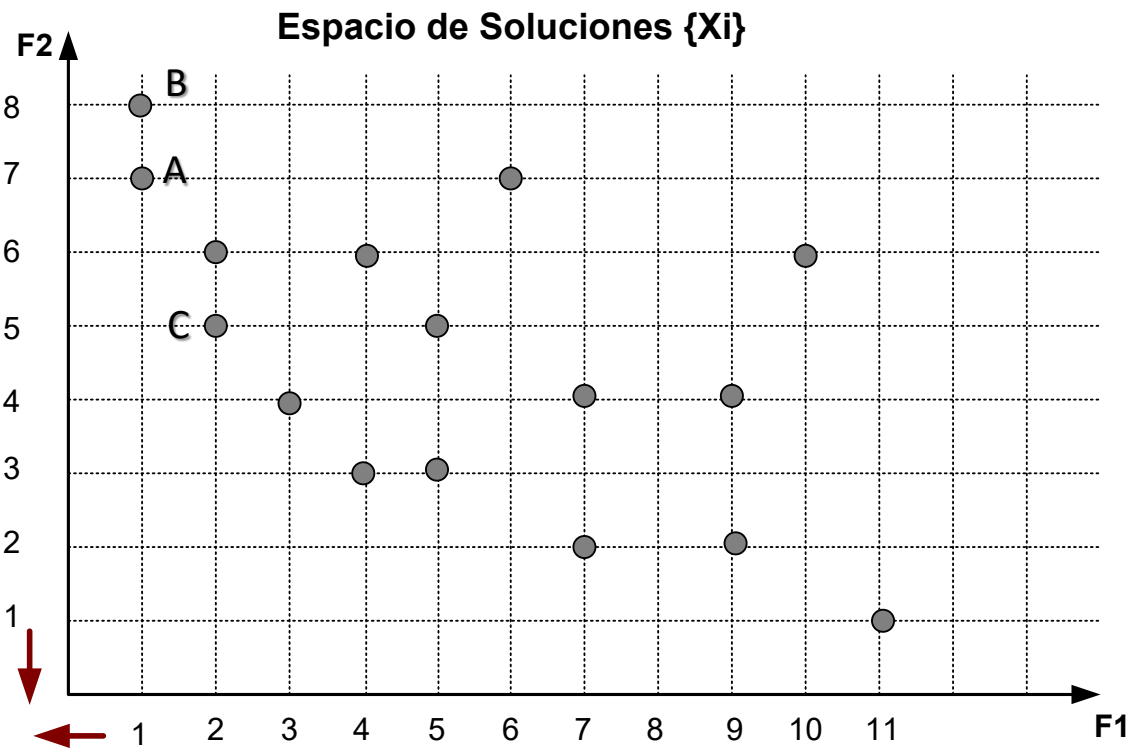
Conjunto de soluciones de la Frontera de Pareto:

- Soluciones que no pueden mejorar uno de los objetivos sin empeorar algún otro
- Estas soluciones determinan las mejores alternativas para la solución del problema



- Un resolutor multiobjetivo debe ofrecer el Conjunto de Soluciones de la Frontera de Pareto (o soluciones más cercanas al frente)
- La frontera puede tener una, muchas o incluso infinitas soluciones
- Se prefiere que las soluciones estén bien distribuidas

Cálculo de la frontera de Pareto (se desea minimizar F1 y F2)



Se ordenan las soluciones según las funciones fitness objetivo (F1, F2,)

	F1	F2
A	1	7
B	1	8
C	2	5
	2	6
	3	4
	4	3
	4	6
	5	3
	5	5
	6	7
	7	2
	7	4
	9	2
	9	4
	10	6
	11	1

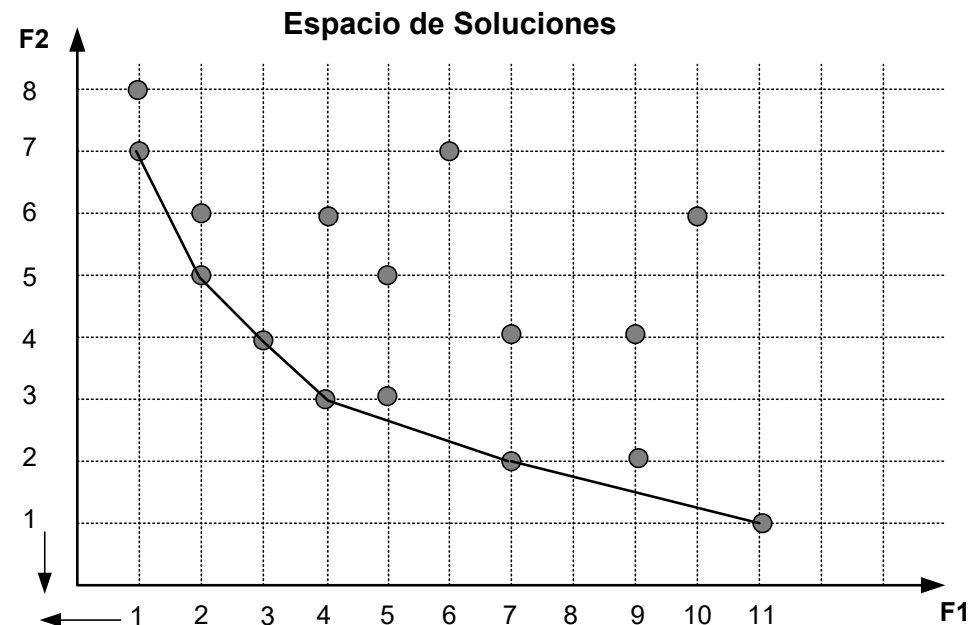
Se puede seguir cualquier orden (F1,F2 / F2,F1, ...)

- 1) Para cada valor de F1, se determina el valor que minora F2
- 2) Siguiendo el orden creciente de F1, se **seleccionan los valores** que vayan minorando F2

F1	F2	
1	7	Primer valor de F1
1	8	No minora F2 (obvio, repite valor F1)
2	5	Minora valor de F2 (baja a 5)
2	6	No minora F2 (obvio, repite valor F1)
3	4	Minora valor de F2 (baja a 4)
4	3	Minora valor de F2 (baja a 3)
4	6	No minora F2 (obvio, repite valor F1)
5	3	No minora F2
5	5	No minora F2 (obvio, repite valor F1)
6	7	No minora F2
7	2	Minora valor de F2 (baja a 2)
7	4	No minora F2 (obvio, repite valor F1)
9	2	No minora F2
9	4	No minora F2 (obvio, repite valor F1)
10	6	No minora F2
11	1	Minora valor de F2 (baja a 1)

La frontera de Pareto, con el conjunto de soluciones Pareto-óptimas, es:

F1	F2
1	7
2	5
3	4
4	3
7	2
11	1

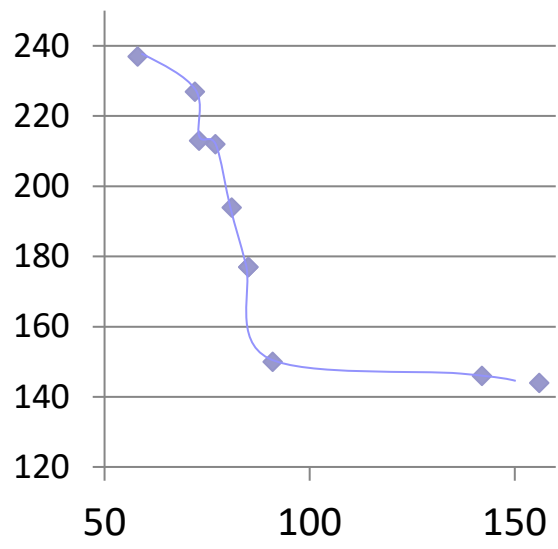


Ejemplo Opt4J (viajante de comercio multi-objetivo)

Cada arista entre un par de nodos i,j tiene dos componentes: $(Coste_{ij}, Distancia_{ij})$

Problema multiobjetivo: $\min [Coste(X), Distancia(X)]$ recorrida

Population Monitor				
#	Individual	State	Coste total (M...	Distancia reco...
1	7 6 5 0 4 9 3 1 ...	evaluated	156.0	144.0
2	7 3 0 9 4 1 2 8 ...	evaluated	91.0	150.0
3	7 6 5 9 4 0 3 1 ...	evaluated	142.0	146.0
4	5 7 3 1 2 8 9 4 ...	evaluated	81.0	194.0
5	7 3 0 9 4 2 8 6...	evaluated	77.0	212.0
6	9 4 0 1 5 6 7 3 ...	evaluated	72.0	227.0
7	2 8 9 4 0 6 1 5 ...	evaluated	58.0	237.0
8	9 4 0 3 1 5 6 7 ...	evaluated	85.0	177.0
9	9 4 0 6 3 1 5 7...	evaluated	73.0	213.0
10	9 4 0 3 1 5 6 7...	evaluated	85.0	177.0
11	5 7 3 1 2 8 9 4...	evaluated	81.0	194.0
12	7 3 0 9 4 2 8 6 ...	evaluated	77.0	212.0
13	7 3 0 9 4 1 2 8...	evaluated	91.0	150.0
14	1 2 8 7 6 5 9 4 ...	evaluated	142.0	146.0
15	1 2 8 7 6 5 0 4 ...	evaluated	156.0	144.0
16	1 2 8 7 6 5 9 4 ...	evaluated	142.0	146.0
17	1 2 8 7 6 5 0 4 ...	evaluated	156.0	144.0
18	1 2 8 7 6 5 0 4 ...	evaluated	156.0	144.0
19	1 2 8 7 6 5 9 4 ...	evaluated	142.0	146.0
20	3 1 2 8 7 6 5 0 ...	evaluated	156.0	144.0
21	1 2 8 7 6 5 0 4 ...	evaluated	156.0	144.0
22	1 2 8 7 6 5 9 4 ...	evaluated	142.0	146.0
23	1 2 8 7 6 5 0 4 ...	evaluated	156.0	144.0
24	1 2 8 7 6 5 0 4 ...	evaluated	156.0	144.0



Se debe decidir una solución de la frontera de Pareto – típicamente decisión del usuario

(MOEA Multi-Objective Evolutionary Algorithms)

El objetivo es obtener tantas soluciones Pareto-óptimas como sea posible y bien distribuidas

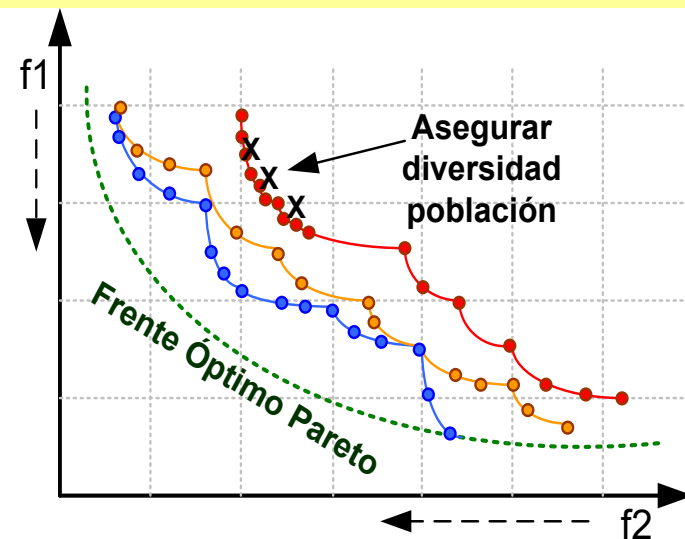
Fundamentos del método elitista de selección:

- Se intenta **priorizar las soluciones que sean dominantes** en cada sucesiva generación del AG (no tienen por qué ser 'todavía' soluciones del frente óptimo de Pareto)
- ¡También debe mantenerse la diversidad de la población!

Proceso de selección:

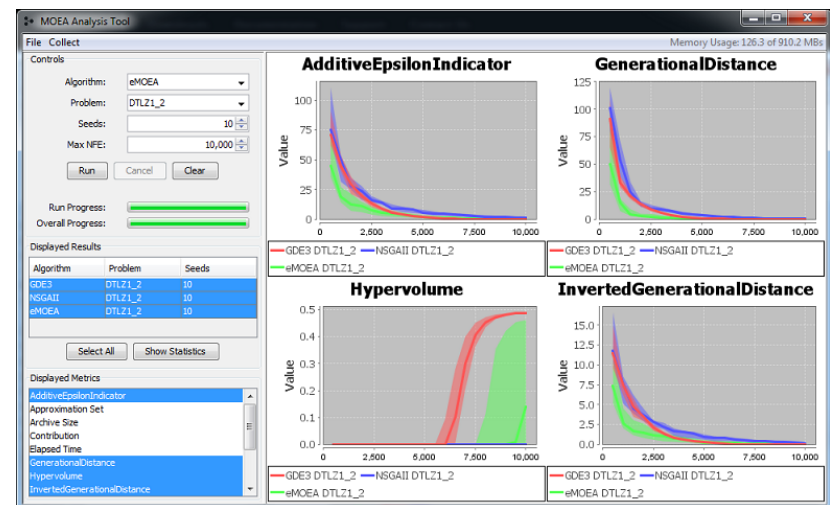
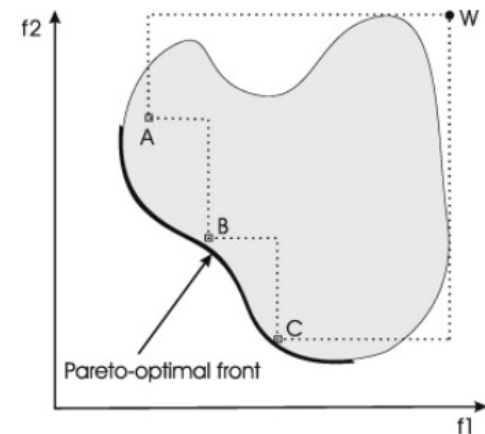
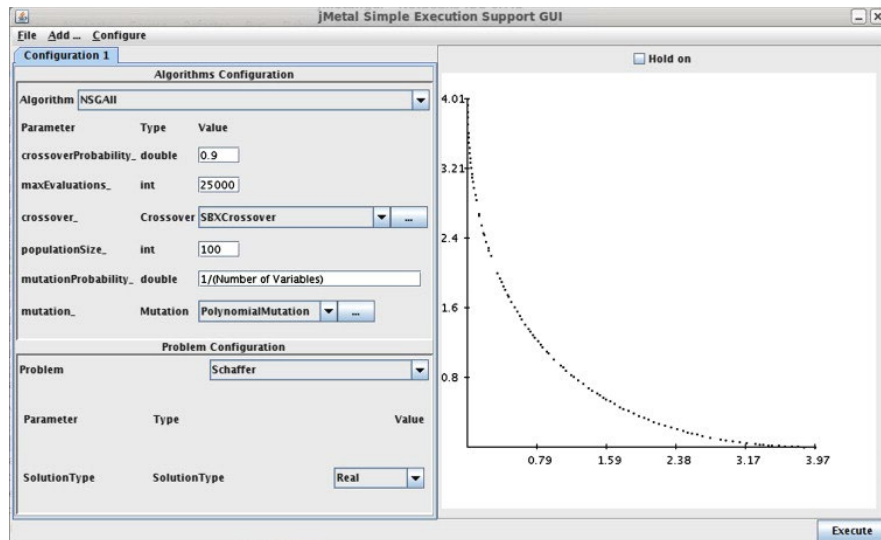
- Las soluciones de la población se agrupan en diferentes *frentes*, típicamente según:
 - nº de soluciones que las dominan (*ranking* de dominancia), o
 - nº de soluciones que dominan (*count* de dominancia)
- Mediante un método elitista, se seleccionan como padres las soluciones de los frentes (de mejor a peor), generando la nueva población hasta alcanzar el tamaño máximo
- Para mantener la diversidad, se priorizan las soluciones más dispersas (que tengan mayor distancia respecto a otras soluciones del frente: *crowding distance sorting*)
- Diversas variantes de ordenación: Rank (NSGA, NSGA-II), Count (MOGA, NPGA), Mixto (SPEA, SPEA2), etc.
- El más conocido es NSGA (*Elitist Non-Dominated Sorting Genetic Algorithm*), con sus variantes

Los procesos de cruce y mutación son similares a un AG normal



Librerías y entornos para AG multi-objetivo

- Project-Platypus/Platypus (Python) <https://github.com/Project-Platypus/Platypus>
- Opt4J (Java): <https://sdarg.github.io/opt4j/>
- jMetal (Java): <https://jmetal.sourceforge.net/>
- MOEAFramework (Java): <http://moeaframework.org/>



- ❑ **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, número prefijado de generaciones o tiempo disponible
- ❑ **Lenguaje de Representación:**
Mejor utilizar alfabetos con ***baja cardinalidad*** (es decir, con pocas letras) como el binario
- ❑ **Función de Aptitud:**
Debe hacer una presión adecuada y reflejar la calidad de la solución en el problema
- ❑ **No fiarse de la autoridad central (operadores que ‘controlan’ a toda la población):**
La Naturaleza actúa de forma distribuida y segmentada. Se debe de minimizar la necesidad de operadores que "vean" a toda la población

Ej: 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

Algoritmos Meméticos

Un Algoritmo Memético trabaja con una población de *agentes* (como extensión de individuo, al ser **más inteligente** que un simple individuo) que alternan períodos de:

- auto-mejora (vía local search),
- con períodos de cooperación (vía recombinación)
- y competición (vía selección)



"On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms". P. Moscato. (en Poliformat)

Caltech Concurrent Computation Program Report 826, Caltech, Pasadena, California (1989)

Aparecen como evolución de los AG Híbridos (incorporan un proceso de mejora local de la solución)

También conocidos como Algoritmos Lamarckianos, Algoritmos evolutivos híbridos, Algoritmos de búsqueda local genética, etc.

Son algoritmos poblacionales, con los principios de selección, cruce y mutación, pero **introduciendo información local del problema en los operadores**, procesos de mejora local y transmisión del conocimiento no exclusivamente genética (colaboración y aprendizaje en los procesos de mejora local, mutación, etc.)

En el **ejemplo de adivinar frases**,

¿qué información podemos añadir al realizar una mutación, o para mejorar un individuo tras realizar un cruce?

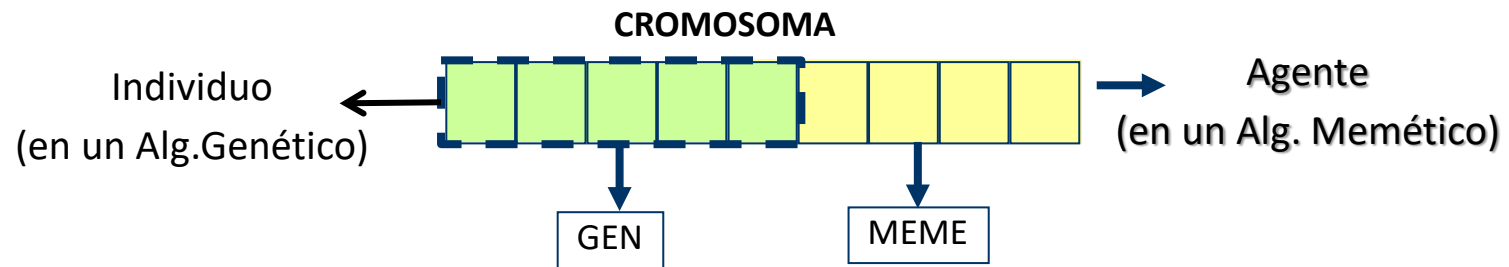
En español, una vocal va habitualmente seguida de...
y una consonante va habitualmente seguida de...

Podemos añadir información local sobre el problema

Similitud con la naturaleza

Los individuos, además de la información genética de sus padres, también aprenden sobre cómo mejorar en el dominio del problema concreto a resolver – además del conocimiento recibido de sus padres, un individuo (agente) puede aprender nuevas cosas que sus padres no saben

- Gen: unidad básica de información genética \Rightarrow Genotipo: características de la solución
- **Meme**: unidad básica de transmisión cultural (imitación, aprendizaje) en el dominio del problema \Rightarrow Aprendizaje sobre proceso mejora local



MEME: Concepto análogo al gen (pero en la evolución cultural / aprendizaje)

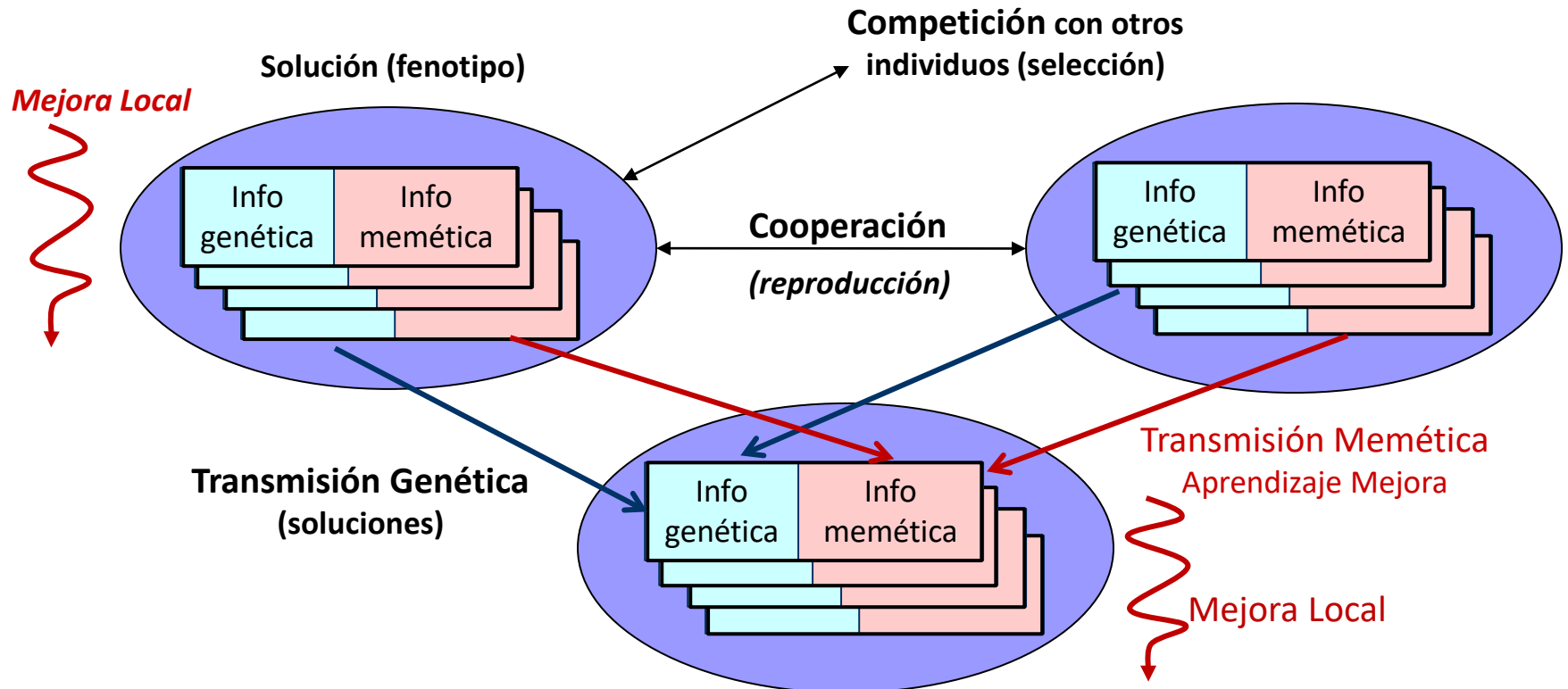
Tratado por Lamarck, 1850: “las experiencias de un organismo afectan directamente el marcaje genético de su prole, por lo que lo aprendido por los padres se traslada genéticamente a sus hijos”

- Los “memes” se propagan de individuo a individuo en un proceso de ‘imitación’ (aprendizaje y mejora)
- Representan información del comportamiento (heurísticas a emplear en un proceso de mejora local)

Los agentes (extensión de individuos) compiten, cooperan y... **mejoran y aprenden a mejorar**

- AG \Rightarrow Los individuos interactúan entre sí en un marco de competición (selección) y cooperación (cruce) de manera análoga al comportamiento de los individuos en la naturaleza
- AG-HÍBRIDO \Rightarrow Los agentes tienen un proceso interno de mejora local de sus soluciones (guiado por heurísticas / metaheurísticas)
- AM \Rightarrow Cada agente es capaz de aprender de sus padres y de **transmitir su conocimiento** (sobre su proceso de mejora local) a sus descendientes

Cada agente puede contener una o más soluciones (aunque no es lo habitual)

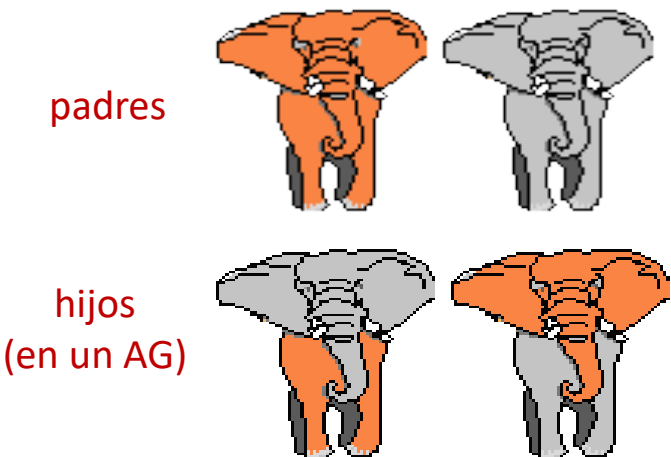


- Los **procesos de mejora local** de los agentes se pueden realizar:
 - a) **Tras cada cruce:** como un proceso de mutación y como un proceso de mejora local, priorizando ciertas características de individuo
 - b) **Al final de cada ciclo** inferencial (generacional)

Cada agente tiene su propio proceso de mejora local (y heurística). En función de su resultado, las características del proceso de mejora local son transmitidas de padres a hijos (meme)

Para que un algoritmo híbrido sea considerado AM, la búsqueda local debe aplicarse integrada en el proceso evolutivo

- Se requieren **Meta-operadores** (extensión de los existentes o **creación de otros nuevos**) para actuar con la información del individuo, de su entorno, de la transmisión del aprendizaje y del problema. Uso de conocimiento



Información memética (que se transmite):

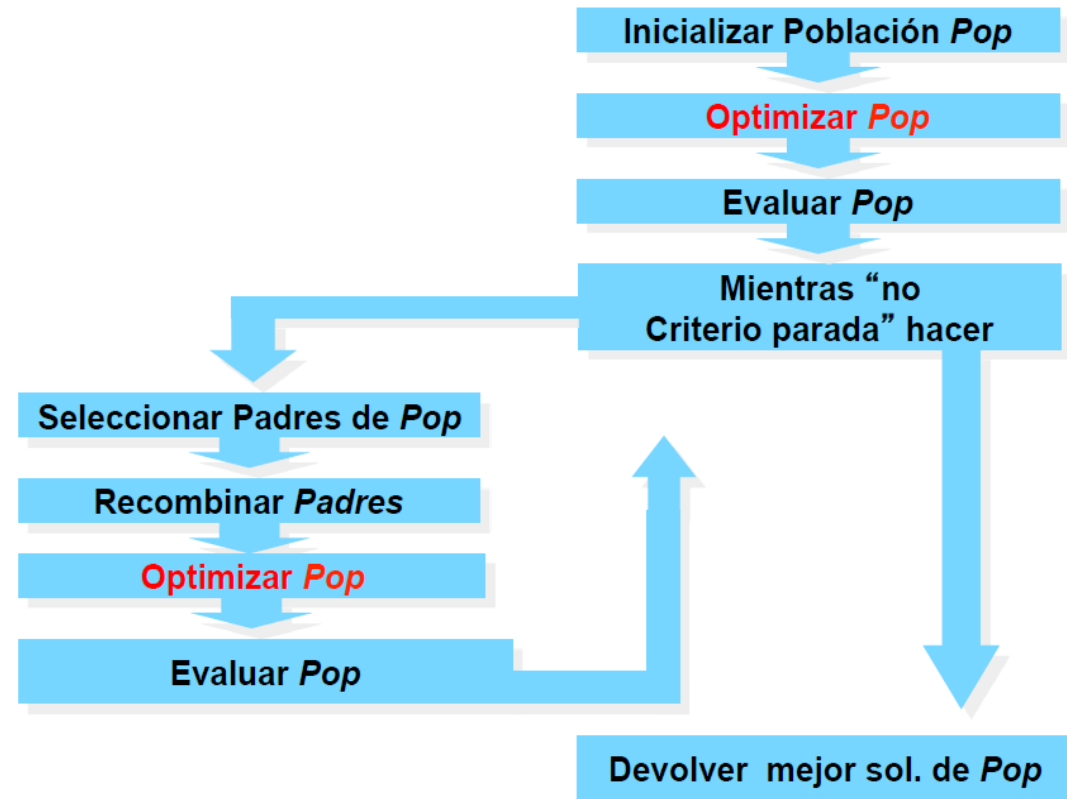
- Es importante ser de un solo color
- Es mejor optimizar el tamaño de las orejas
- Afinar el grado de curvatura de los colmillos
- etc.

Características del proceso de Mejora Local a transmitir:

- ✓ Esfuerzo en la búsqueda (competición recursos)
- ✓ Sobre qué individuos
- ✓ Definición de la vecindad
- ✓ Tipo metaheurística de mejora local, Parámetros
- ✓ Etc.

GA	
<i>codificación</i>	esquemas, cadenas lineales, alfabetos predefinidos
<i>individuo</i>	solución al problema
<i>cruce</i>	intercambio no-guiado de información
<i>mutación</i>	introducción aleatoria de nueva información

MA	
<i>representación</i>	formas, no-linealidad, cercanía al problema
<i>agente</i>	solución/ones al problema + mecanismo mejora local
<i>recombinación</i>	intercambio guiado de información
<i>mutación</i>	introducción sensible de nueva información
<i>mejora local</i>	aprendizaje lamarckiano



UN AM puede caracterizarse como

- una colección de agentes que realizan exploraciones autónomas del espacio de búsqueda
- cooperando ocasionalmente a través de la recombinación, y
- compitiendo continuamente por los recursos computacionales a través de la selección y el reemplazo

Esquema de un Algoritmo Memético (hay multiples variantes)

Inicialización: Generar una población inicial de agentes/soluciones: genotipos + memes;

Mientras No condición-de-Parada, hacer

*Población inicial de Agentes:
{Soluciones}, {Características}, {Procesos mejora local}*

1 Evaluar los agentes (fenotipos) de la población (fitness);

2 Paso Generacional: Obtener una nueva población

Padres = Selección (Población, Fitness);

Nueva_Población = Reproducir (Padres, **Cruce**);

Nueva_Población = Mutación (Población, **Mutación**);

Población = Actualizar (Población, Nueva_Población, **Reemplazo**, Fitness);

*Información a heredar: Genes + Memes:
No exclusivamente genes, sino características
inferidas de buenas soluciones
¿Cómo aplicar buenos procesos de mejora?
Cruces basados en información del problema*

*Incluye mutación genes y memes
Posible Proceso de Mejora*

3 Seleccionar un conjunto de agentes (W) que deben someterse a mejora individual;

*Incluye genes
y memes*

4 Para cada agente de W, hacer:

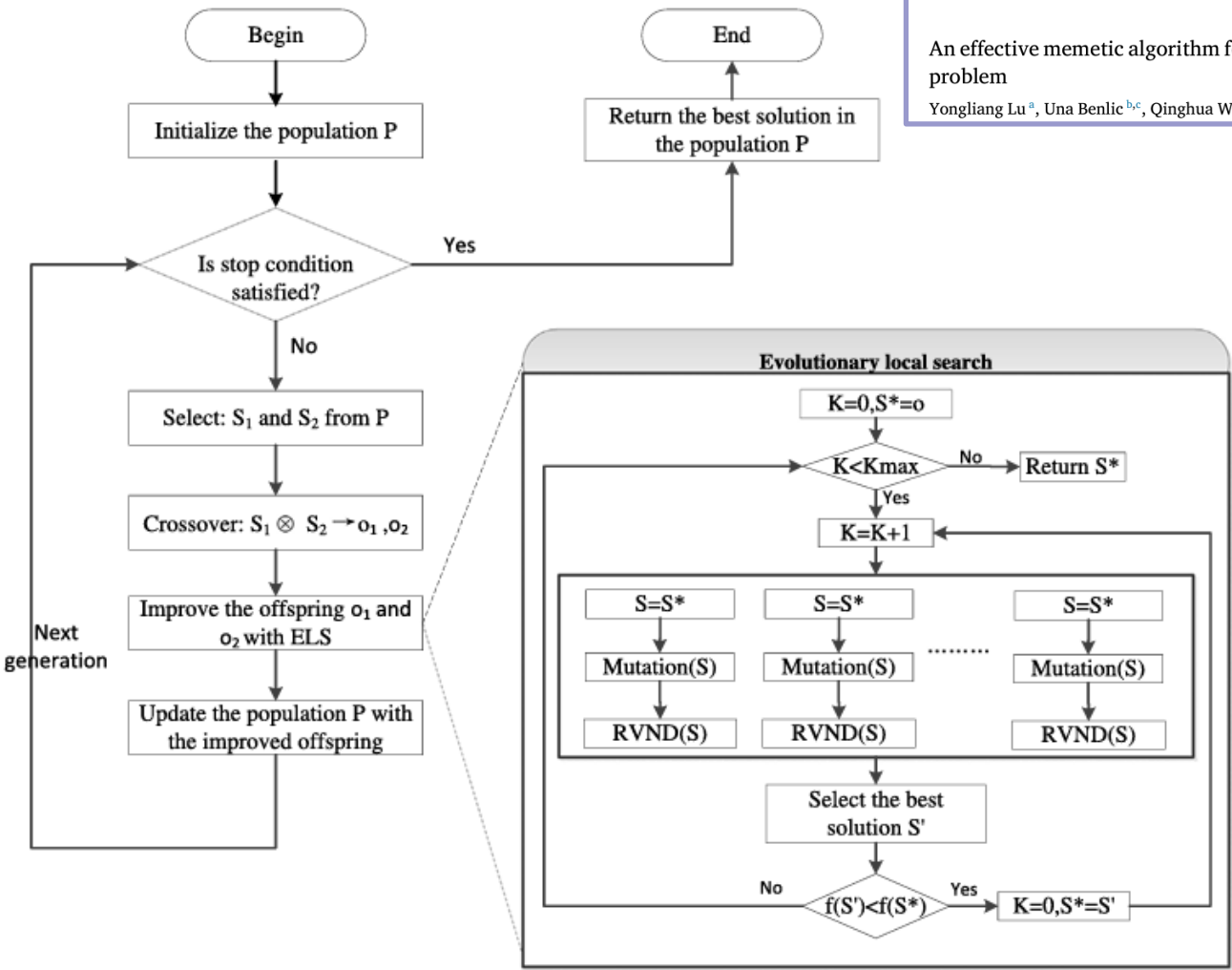
Nuevo agente W = Proceso_Mejora (W, **Búsqueda Local**);

*En la Mejora Local, el agente hace
uso de información heredada del
padre (meme)*

Fin Mientras

Los meta-operadores de Selección, Cruce, Mutación y Reemplazo trabajan con información añadida del problema para hacer un proceso no exclusivamente sintáctico

Ejemplo. Distribución y reparto de bicicletas (artículo en Poliformat)



Engineering Applications of Artificial Intelligence 95 (2020) 103890

Contents lists available at [ScienceDirect](#)

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

An effective memetic algorithm for the generalized bike-sharing rebalancing problem

Yongliang Lu ^a, Una Benlic ^{b,c}, Qinghua Wu ^{a,*}

Decisiones de diseño de un Algoritmo Memético

- **¿Cuándo se aplica el Algoritmo de Búsqueda Local?:** ¿tras cada cruce?, ¿al final de cada ciclo? ¿cada k-generaciones? ¿decisión estocástica?
- **¿Sobre qué agentes (W) se aplica?:** ¿Sobre toda la población? ¿Sobre un subconjunto (mejores, representantes, aleatorios, etc.)?
- **¿Qué uso se hace del agente optimizado?:** ¿Sustituye al agente origen (aprendizaje Lamarkiano)? ¿El agente origen se mantiene y solo recibe el fitness mejorado (aprendizaje Baldwiniano)?
- **¿Qué Algoritmo de Búsqueda Local se utiliza?**
- **¿Cómo se aplica la optimización local?** (Intensidad y frecuencia)

Por ejemplo:

En cada generación, aplicar la BL sobre los **N/10 mejores** cromosomas de la población (N es el tamaño de la población)

En cada generación, aplicar la BL sobre **todos** los cromosomas de la población

Cada 10 generaciones, aplicar la BL sobre los N/10 mejores cromosomas

Cada 10 generaciones, aplicar la BL a N/10 individuos seleccionados estocásticamente (ruleta, rango...)

El AM puede verse como una colección de agentes que:

1. realizan exploraciones de mejora en su espacio de búsqueda (mutaciones),
2. cooperan ocasionalmente con el resto de agentes (soluciones) por medio de operaciones de cruce, y
3. compiten por los recursos computacionales (espacio) con procesos de selección y reemplazo

La aportación fundamental frente a un AG es:

- La representación del individuo (solución) suele ser más compleja, estructurada, en vez de cadenas lineales (representado las soluciones de forma más estructurada)
- Un agente puede contener diversas soluciones e incorpora heurísticas de mejora local
- Se incorpora un proceso de mejora local del agente, guiado por heurísticas propias. Esta mejora del padre (aprendizaje en el meme) se traslada y hereda a los hijos

Equilibrio en un AM:

- Los AM deben encontrar un equilibrio entre la diversidad de la población (AG) y la convergencia en vecindarios (búsqueda local)

Los AM no son un paradigma “ortodoxo” y sistemático. Hay muchos grados de libertad para el usuario. Existen muchas variantes y alternativas de diseño. **Un AM no es solo un AG + búsqueda local.** Muchas veces están *disfrazados* como AG o algoritmos evolutivos

En general, son más eficaces que los AG (pero más complejos de diseñar)

Búsqueda dispersa. (Scatter Search, Glover 1997)

“Heuristics for Integer programming Using Surrogate Constraints”. Decision Sciences 8 (1): 156–166, 1977

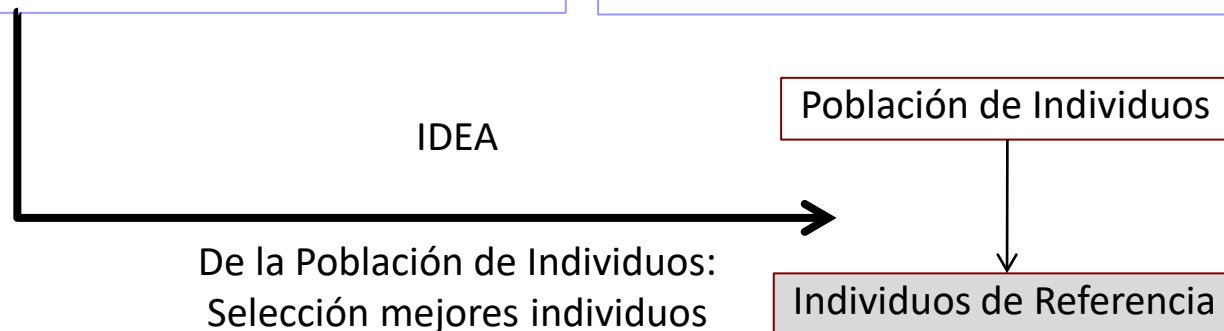
- Evolución de los algoritmos genéticos (como algoritmo poblacional, que construye soluciones mediante la combinación de otras soluciones)
- Integración de la Búsqueda en Haz (*Beam Search*) en un AG

Búsqueda por haz local (beam search)

- Comienza con k estados (soluciones) generados aleatoriamente
- En cada paso se generan todos los sucesores (b) de los k estados: $k*b$ nuevos estados
- Se seleccionan los k mejores sucesores de la lista completa y se repite el proceso

Algoritmo Genético

- Comienza con n individuos (soluciones) generados aleatoriamente
- En cada paso se cruzan y mutan, generando nuevos individuos
- Los nuevos individuos se añaden (o reemplazan) a la población inicial y se repite el proceso



Algoritmo poblacional que construye soluciones mediante la combinación de otras soluciones

- Opera sobre un conjunto de soluciones, llamadas **soluciones de referencia**, que son una selección de “buenas soluciones”, extraída del conjunto de soluciones

Población: {Soluciones} \Rightarrow Soluciones de referencia: {Mejores-Soluciones}

Población reducida. Típicamente, $|\text{Sol_Ref}| \leq |\text{Población}|/10$

- Las Soluciones de Referencia son combinadas para obtener nuevas soluciones.

Las combinaciones son “*formas*” de combinaciones lineales de (dos o más) soluciones:

$$\text{Solución-Nueva} = \alpha_1 S_1 + \alpha_2 S_2 + \dots + \alpha_n S_n, \quad \textit{Similar a Cruce y Selección}$$

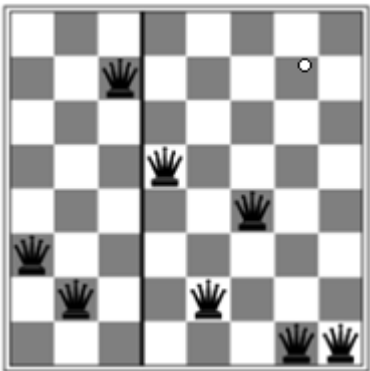
con procesos adaptativos para garantizar la factibilidad

Este proceso (combinación lineal ponderada de soluciones previas) es una forma más estructurada de ‘cruce’ y ‘selección’

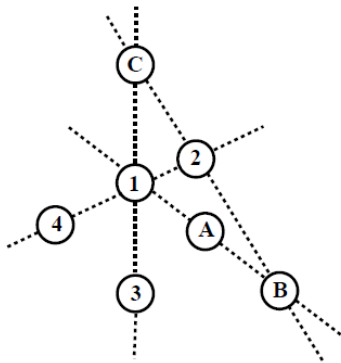
- Las nuevas soluciones suelen tener un posterior proceso de **mejora local**
- De las antiguas soluciones de referencia, y las nuevas soluciones obtenidas, se obtiene el nuevo conjunto de soluciones de referencia (proceso iterativo)

Ejemplo 1: 8 reinas

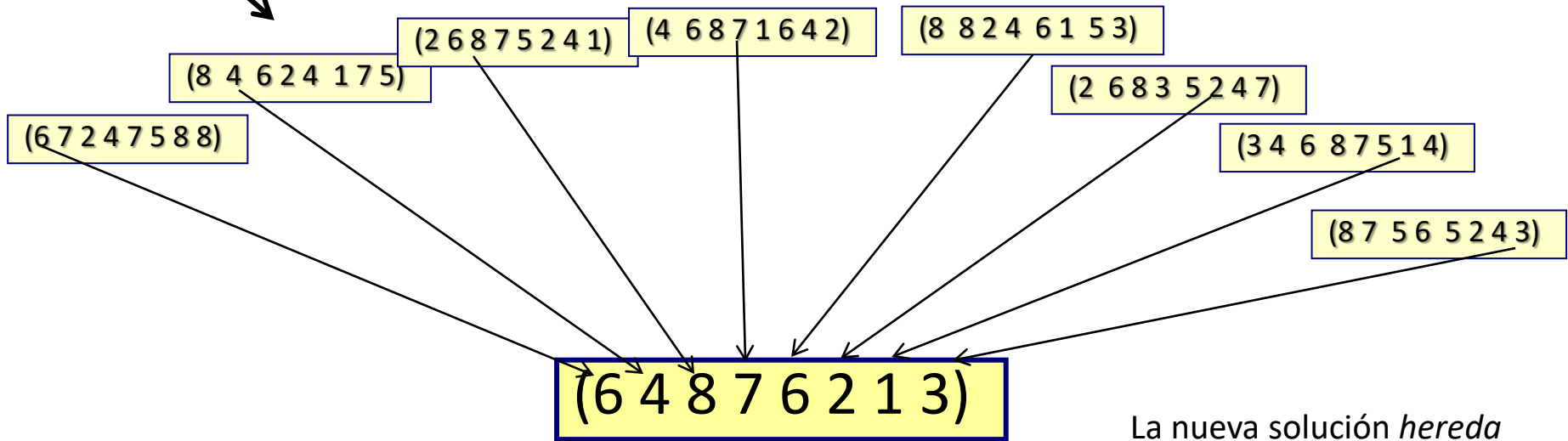
Soluciones de Referencia



- Las nuevas soluciones se obtienen mediante combinación lineal de soluciones de referencia
- El objetivo es transmitir las buenas propiedades que tienen las mejores soluciones (generando soluciones en el entorno de buenas soluciones)



Soluciones de Referencia



Nueva solución

La nueva solución *hereda* propiedades del conjunto de soluciones de referencia (múltiples padres)

$$\text{Solución-Nueva} = \alpha_1 S_1 + \alpha_2 S_2 + \dots + \alpha_n S_n$$

Nueva solución: Obtiene fila_i de la solución_i

Ejemplo 2: Generar rutas de dos autobuses, a través de 10 nodos, para recoger a estudiantes. (Corberán et al. Tech report TR08-2000, D.E.I.O., UV.2000)

Supongamos dos soluciones: **Solución A:** $A1 = \{ 4, 2, 7, 1 \}$, $A2 = \{ 5, 10, 6, 9, 3, 8 \}$

Solución B: $B1 = \{ 2, 6, 8, 10, 9 \}$, $B2 = \{ 3, 4, 7, 1, 5 \}$

Combinando A y B, obtenemos dos nuevas soluciones: $N1 = \text{Combina}(A1, B2)$, $N2 = \text{Combina}(A2, B1)$

La combinación es **más compleja** que un simple cruce

COMBINACIÓN DE SOLUCIONES.

Paso	Par	Voto 1	Voto 2	Asignación	Regla de selección
1	(A_1, B_2)	4	3	$N_1 = \{ 4 \}$	azar
2	(A_2, B_1)	5	2	$N_2 = \{ 2 \}$	azar
3	(A_1, B_2)	7	3	$N_1 = \{ 4, 3 \}$	3 antes 7
4	(A_2, B_1)	5	6	$N_2 = \{ 2, 5 \}$	5 antes 6
5	(A_1, B_2)	7	7	$N_1 = \{ 4, 3, 7 \}$	igual
6	(A_2, B_1)	10	6	$N_2 = \{ 2, 5, 6 \}$	azar
7	(A_1, B_2)	1	1	$N_1 = \{ 4, 3, 7, 1 \}$	igual
8	(A_2, B_1)	10	8	$N_2 = \{ 2, 5, 6, 10 \}$	10 antes 8
9		9	8	$N_2 = \{ 2, 5, 6, 10, 8 \}$	8 antes 9
10		9	9	$N_2 = \{ 2, 5, 6, 10, 8, 9 \}$	igual

Casos:

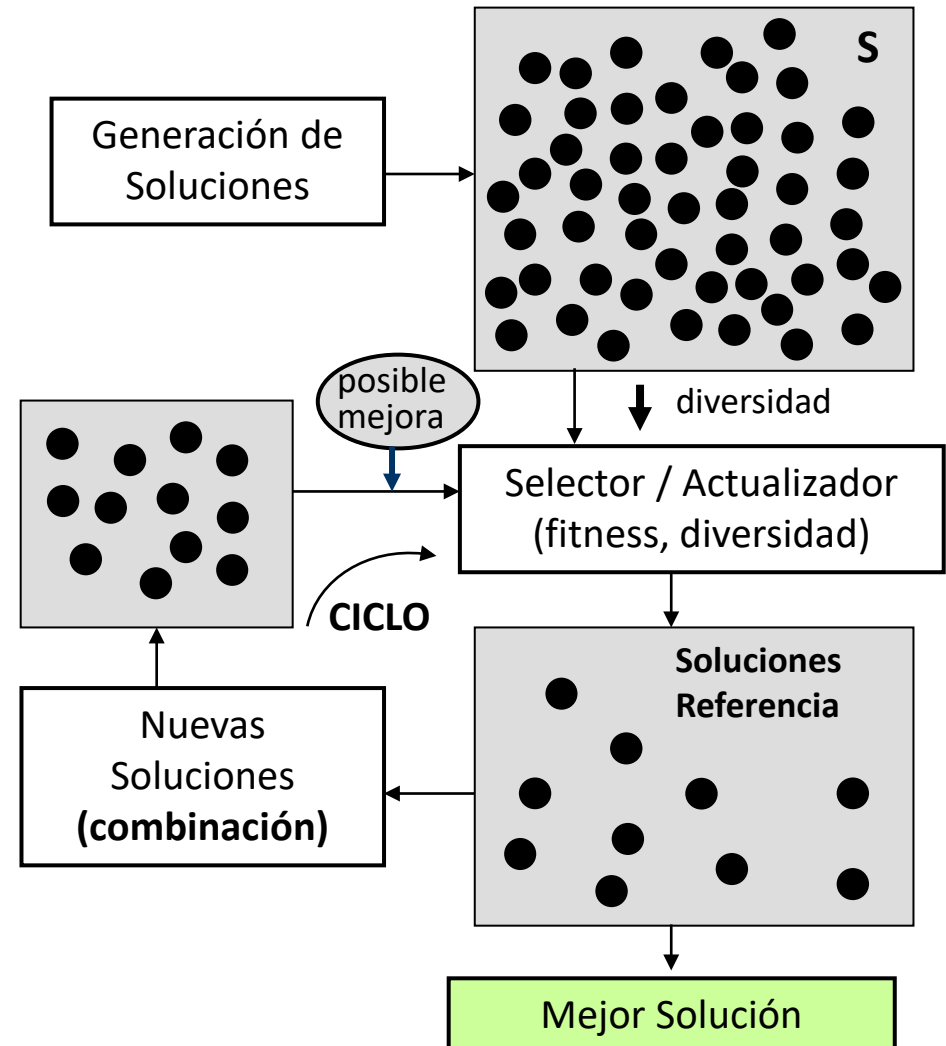
- En $N1$ se añade $\{4, \mathbf{3}\}$, porque '2' ya en $N2$ y '3' precede a '7' en $B2$
- En $N2$ se añade $\{2, \mathbf{5}\}$, porque 5 precede a 6 en $A2$
- Etc.

Obtenemos: $N1 = \{ 4, 3, 7, 1 \}$ y $N2 = \{ 2, 5, 6, 10, 8, 9 \}$

Esquema General Búsqueda Dispersa

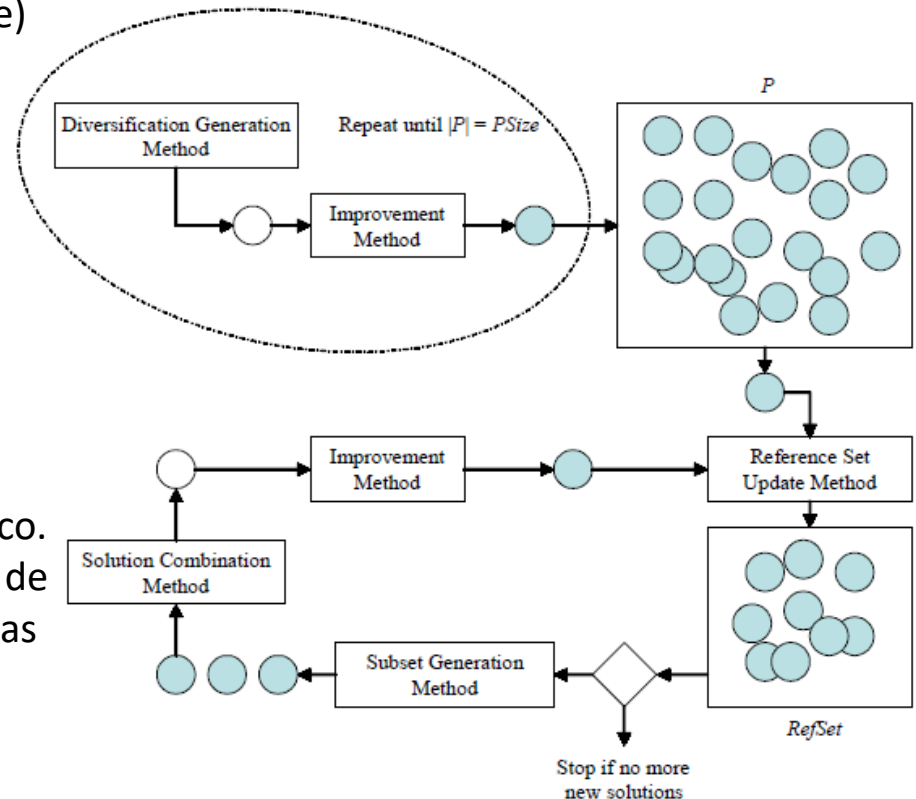
1. Generar conjunto inicial de soluciones que garantice la diversidad (semillas) – soluciones de alta calidad (élite)
2. Seleccionar un subconjunto de las mejores soluciones (criterios: función objetivo y diversidad). Soluciones de Referencia
3. Crear nuevas soluciones combinando soluciones de referencia. Se adaptan para hacerlas factibles
4. Mejorar las nuevas soluciones (búsqueda local y heurísticas)
5. Extraer las mejores soluciones y se añaden al conjunto de soluciones de referencia
6. Eliminar del conjunto de referencia las menos optimizadas/diversas
7. Tras varias iteraciones, se extrae la mejor solución

Si en alguna iteración no se consigue mejorar el conjunto de soluciones de referencia, se añaden nuevas soluciones de S que añadan diversidad



La búsqueda dispersa fundamentalmente introduce:

1. Soluciones de referencia (haz), como selección de de las mejores soluciones en la población (élite)
2. Método de combinación de soluciones de referencia
3. Mejora local de nuevas soluciones
4. Revisión del nuevo conjunto de soluciones de referencia (haz)
5. Es menos aleatorio que un AG y más sistemático. Es más difícil de diseñar debido a la necesidad de encontrar estructuras de referencia y heurísticas específicas



Scatter Search: Diseño Básico y Estrategias Avanzadas. Rafael Martí y Manuel Laguna (en Poliformat)

Adecuado para problemas de los que se tengan soluciones iniciales optimizadas