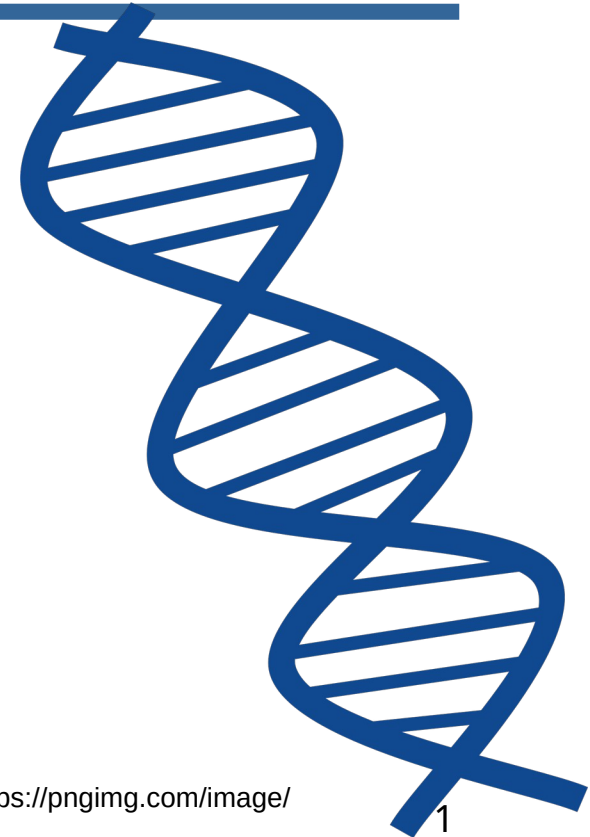

Metaheurísticas

Seminario 3. Problemas de optimización con técnicas basadas en poblaciones

1. Estructura de un Algoritmo Genético/Memético y Aspectos de Implementación
2. Problemas de Optimización con Algoritmos Genéticos y Meméticos
 - Problema de Mínima Dispersión
 - Problema de Selección de Influyentes



Estructura de un Algoritmo Genético

Procedimiento Algoritmo Genético

Inicio (1)

$t = 0$;

inicializar $P(t)$

evaluar $P(t)$

Mientras (no se cumpla la condición de parada) hacer

Inicio(2)

$t = t + 1$

seleccionar P' desde $P(t-1)$

recombinar P'

mutar P'

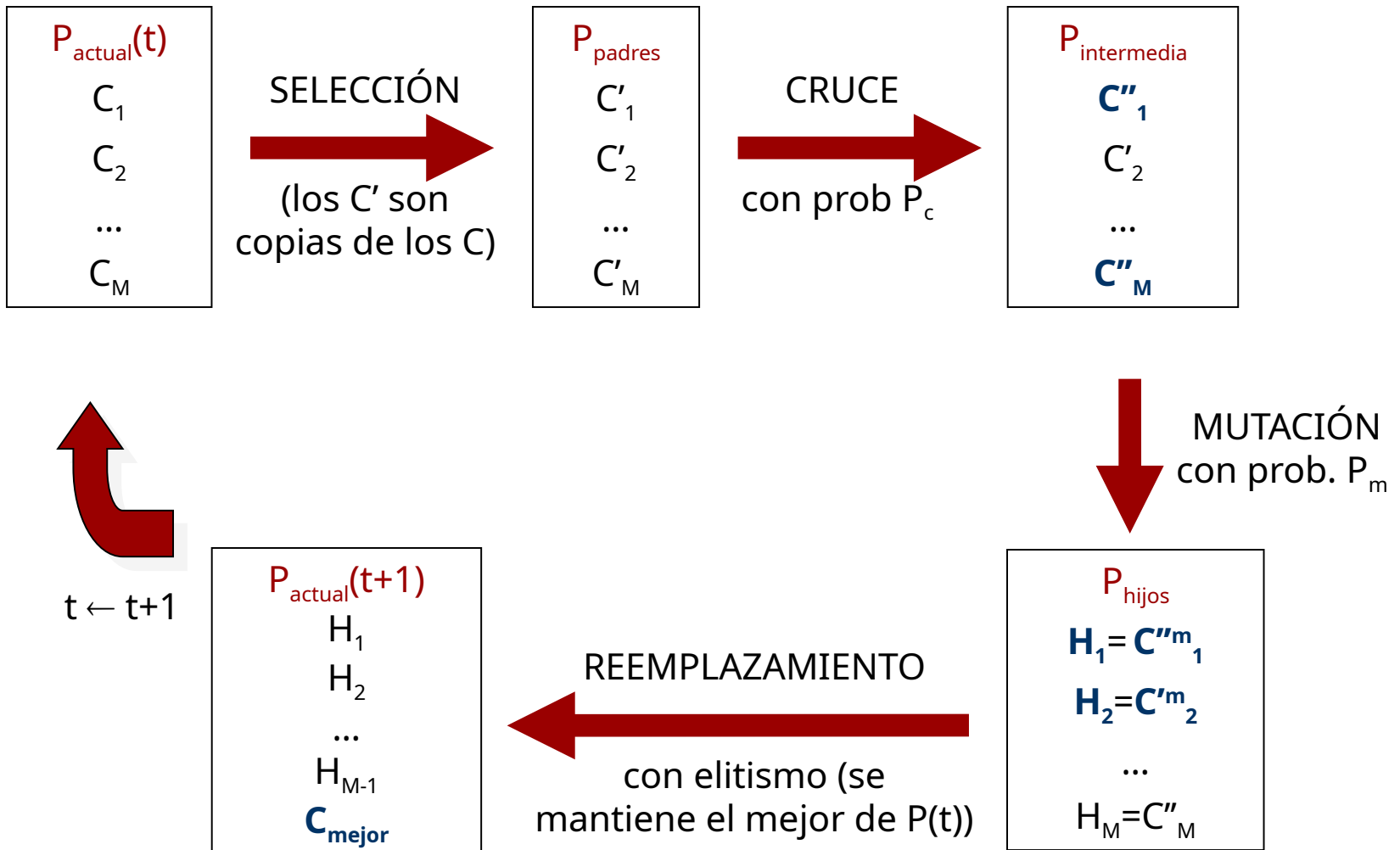
reemplazar $P(t)$ a partir de $P(t-1)$ y P'

evaluar $P(t)$

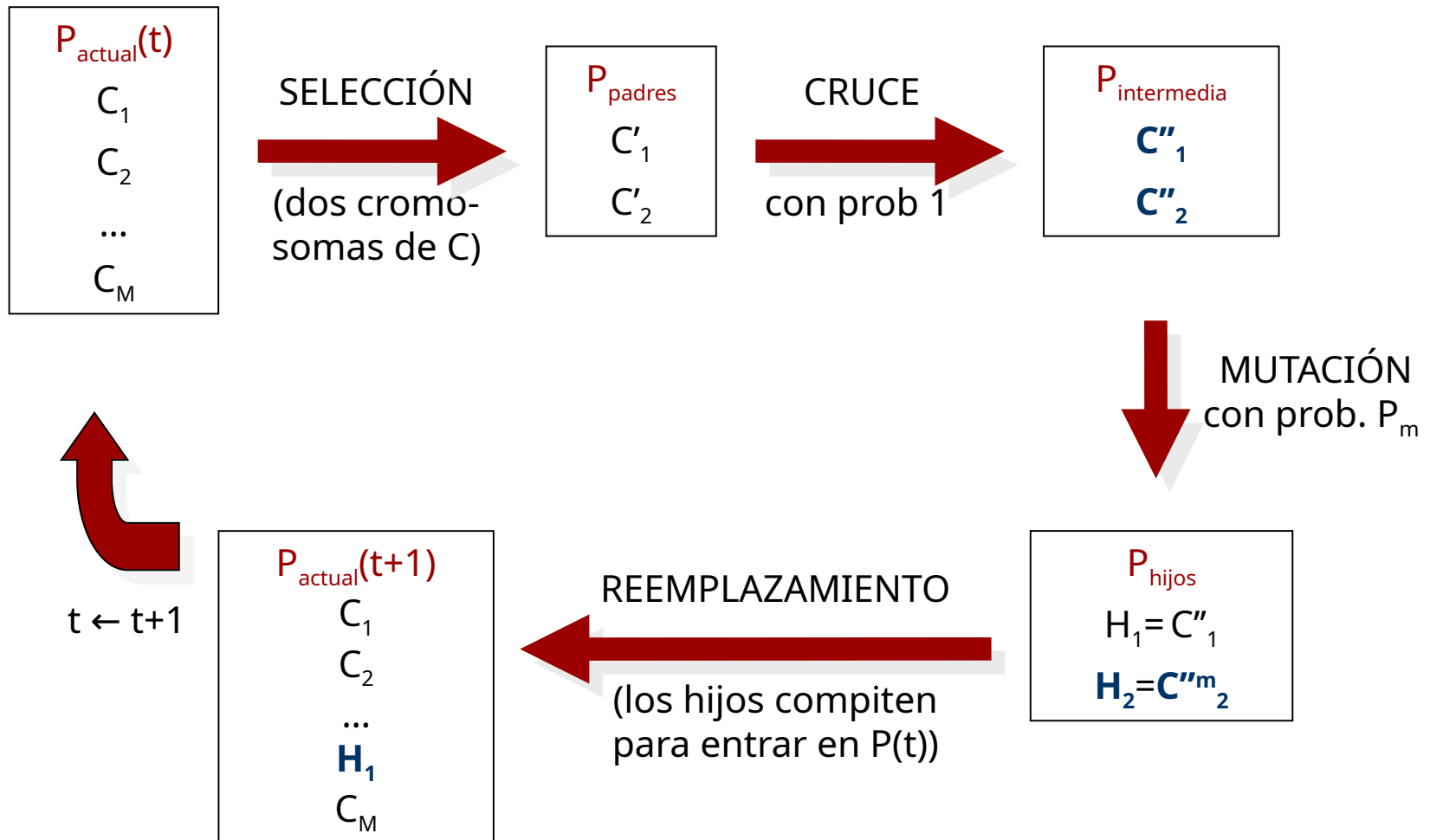
Final(2)

Final(1)

Modelo Generacional



Modelo Estacionario



Modelo Generacional:

Aspectos de Implementación

- ✓ Lo mas costoso en tiempo de ejecución de un Algoritmo Genético es la generación de números aleatorios para:
 - ✓ Aplicar el mecanismo de selección
 - ✓ Emparejar las parejas de padres para el cruce
 - ✓ Decidir si una pareja de padres cruza o no de acuerdo a P_c
 - ✓ **Decidir si cada gen muta o no de acuerdo a P_m**
- ✓ Se pueden diseñar implementaciones eficientes que reduzcan en gran medida la cantidad de números aleatorios necesaria:
 - ✓ Emparejar las parejas para el cruce: Como el mecanismo de selección ya tiene una componente aleatoria, se aplica siempre un emparejamiento fijo: el primero con el segundo, el tercero con el cuarto, etc.

Modelo Generacional:

Aspectos de Implementación

- ✓ Decidir si una pareja de padres cruza: En vez de generar un aleatorio u en $[0,1]$ para cada pareja y cruzarla si $u < P_c$, se estima a priori (al principio del algoritmo) el número de cruces a hacer en cada generación (**esperanza matemática**):

$$N^{\circ} \text{ esperado cruces} = \lceil P_c \cdot M/2 \rceil$$

- ✓ Por ejemplo, con una población de 50 cromosomas (25 parejas) y una P_c de 0.7, cruzarán $0,7 \cdot 25 = 17.5$ parejas \Rightarrow 18 parejas (36 cromosomas)
- ✓ De nuevo, consideramos la aleatoriedad que ya aplica el mecanismo de selección y cruzamos siempre las $N^{\circ} \text{ esperado cruces}$ primeras parejas de la población intermedia

Modelo Generacional:

Aspectos de Implementación

- ✓ Decidir si cada gen muta: El problema es similar al del cruce, pero mucho mas acusado
- ✓ Normalmente, tanto el tamaño de población M como el de los cromosomas n es grande. Por tanto, el número de genes de la población, $M \cdot n$, es muy grande
- ✓ La P_m , definida a nivel de gen, suele ser muy baja (p.e. $P_m < 1\%$). Eso provoca que se generen muchos números aleatorios para finalmente realizar muy pocas mutaciones
- ✓ Por ejemplo, con una población de 50 cromosomas de 10 genes cada uno tenemos 500 genes de los cuales mutarían unos 5 ($N^\circ \text{ esperado mutaciones} = P_m \cdot n^\circ \text{ genes población}$, **esperanza matemática**).

Modelo Generacional:

Aspectos de Implementación

- ✓ Generar 500 números aleatorios en cada generación para hacer sólo 5 mutaciones (en media) es un gasto inútil. Para evitarlo, haremos siempre exactamente *N° esperado mutaciones* en cada generación
- ✓ Planteamos una probabilidad fija de mutación por Individuo en vez de por gen (ej: 10%), con lo que se sabe le número de mutaciones por generación (5 individuos), y se genera un entero en $\{1, \dots, \text{popsize}\}$ para escoger el cromosoma.
- ✓ Aparte de hacer un número fijo de mutaciones, hay que decidir cuáles son los genes que mutan.
- ✓ Normalmente, eso se hace también generando números aleatorio para escoger el gen $\{1, \dots, m\}$.

Aspectos de Diseño de los Algoritmos Meméticos

- Una decisión fundamental en el diseño de un Algoritmo Memético (AM) es la definición del equilibrio entre
 - la exploración desarrollada por el algoritmo de búsqueda global (el Algoritmo Genético (AG)) y
 - la explotación desarrollada por el algoritmo de búsqueda local (BL)
- La especificación de este **equilibrio entre exploración y explotación** se basa principalmente en dos decisiones:
 - ¿Cuándo se aplica el optimizador local
 - En cada generación del AG o
 - cada cierto número de generaciones
 - y sobre qué agentes?
 - Sólo sobre el mejor individuo de la población en la generación actual o
 - Sobre un subconjunto de individuos escogidos de forma fija (los m mejores de la población) o variable (de acuerdo a una probabilidad de aplicación p_{LS})

Aspectos de Diseño de los Algoritmos Meméticos

2. ¿Sobre qué agentes se aplica (anchura de la BL) y con qué intensidad (profundidad de la BL)?
 - ✓ AMs baja intensidad (alta frecuencia de aplicación de la BL/pocas iteraciones)
 - ✓ AMs alta intensidad (baja frecuencia de la BL/muchas iteraciones)

Algoritmo Genético para el MDD

- **Representación:** binaria durante el cruce, en el que las posiciones del vector $i=1, \dots, n$ representan los objetivos y los valores $\pi(1), \dots, \pi(n)$ contenidos en ellas si es elegido (1) o no (0).
- **Generación de la población inicial:** como define el guion, de forma totalmente aleatoria.
- **Modelos de evolución:** 2 variantes: generacional con elitismo / estacionario con 2 hijos que compiten con los dos peores de la población
- **Mecanismo de selección:** torneo de 3 soluciones permitiendo repetir.
- **Operador de mutación:** Se intercambia un nodo elegido por otro disponible.
- **Operador de cruce:** El cruce de intercambio, y el cruce uniforme con reparación.

Algoritmo Genético para el MDD

Cruce uniforme con reparación

- Genera dos hijos a partir de dos padres
- Los valores comunes son mantenidos en ambos hijos.
- Para el resto de posiciones se dividen aleatoriamente para una solución hija o la otra.

Padre₁ = (0 1 0 1 0 1 0 1 0)
Padre₂ = (1 0 0 1 1 1 0 0 0)

→

Hijo₁ = (0 0 0 1 0 1 0 1 0)
Hijo₂ = (1 1 0 1 1 1 0 0 0)

- Al terminar el número de nodos puede no ser correcto, será necesario añadir o eliminar hasta tener el tamaño correcto. Debéis definir vuestro propio criterio reparación, incluso puede usar información del problema (mediante una heurística simple).

Algoritmo Genético para el MDD

Cruce de intercambio

- Genera dos hijos a partir de dos padres
- Los valores comunes son mantenidos en ambos hijos.
- El resto de unos se reparte aleatoriamente en el resto de posiciones.

Shuffle([0, 1, 0, 1, 0]) => [1 0 0 1 0]

Padre₁ = (0 1 0 1 0 1 0) → Hijo₁ = (1 0 0 1 0 1 0)

Padre₂ = (1 0 0 1 1 1 0 0 0) → Hijo₂ = (1 1 0 1 0 1 0 0 0)

Shuffle([0, 1, 0, 1, 0]) => [1 1 0 0 0]

- Al terminar el número de nodos será correcto, por lo que no será necesario hacer ningún otro cambio.

Algoritmo Genético para el SNIMP


- **Representación**: entera incluso durante el operador de cruce.
- **Generación de la población inicial**: como define el guion, de forma totalmente aleatoria.
- **Modelos de evolución**: 2 variantes: generacional con elitismo / estacionario con 2 hijos que compiten con los dos peores de la población
- **Mecanismo de selección**: torneo de 3 soluciones permitiendo repetir.
- **Operador de mutación**: Se intercambia un nodo elegido por otro disponible.
- **Operador de cruce**: El cruce ordenado, y el cruce en dos puntos con reparación.

Algoritmo Genético para el SNIMP

Cruce en dos puntos con reparación

- Genera dos hijos a partir de dos padres.
- Elige aleatoriamente un segmento a intercambiar.
- El hijo₁ es como el padre₁ excepto en el segmento, en donde es similar al padre₂.
- El hijo₂ es como el padre₂ excepto en el segmento, en donde es similar al padre₁.

Padre₁ = (30 2 8 5 6)
Padre₂ = (12 10 4 3 2)



Hijo₁ = (30 10 4 3 6)
Hijo₂ = (12 2 8 5 2 6)

- Al terminar el número de nodos puede no ser correcto por repetición, en ese caso se cambiarán por otros valores del otro padre que no genere repetición. Debeis definir vuestro propio criterio de selección, incluso puede usar información del problema (mediante una heurística simple).

Algoritmo Genético para el SNIMP

Cruce con orden

- Genera dos hijos a partir de dos padres.
- Al principio combina los valores de ambos padres de forma ordenada.
- El hijo₁ se construye a partir de las posiciones impares de la lista ordenada.
- El hijo₂ se construye a partir de las posiciones pares de la lista ordenada.

Padre₁ = (30 2 8 5 6)

Padre₂ = (12 10 4 3 2)

Lista ordenada = (2 2 3 4 5 6 8 10 12 30)

Hijo₁ = (2 3 5 8 12)

Hijo₂ = (2 4 6 10 30)

Problemas de Optimización con Algoritmos Meméticos

- En los dos problemas (MDD y SNIMP), emplearemos un AM consistente en un AG generacional que aplica una BL (Seminario 2) a cierto número de cromosomas cada cierto tiempo.
- Se estudiarán las siguientes tres posibilidades de hibridación:
 - **AM-(10,1.0)**: Cada **10** generaciones, aplicar la BL sobre **todos los cromosomas** de la población
 - **AM-(10,0.1)**: Cada **10** generaciones, aplicar la BL sobre un **subconjunto de cromosomas** de la población seleccionado aleatoriamente con probabilidad p_{LS} igual a **0.1** para cada cromosoma
 - **AM-(10,0.1mej)**: Cada **10** generaciones, aplicar la BL sobre los **0.1·N mejores** cromosomas de la población actual (N es el tamaño de ésta)
- Se aplicará **una BL de baja intensidad**. En QKP se evaluarán sólo m vecinos en cada aplicación (siendo m el tamaño del problema), y en APC se evaluarán $2 \cdot n$ vecinos en cada aplicación, dos por cada componente.

NOTA SOBRE LOS TIEMPOS DE EJECUCIÓN

En los AGs de la segunda práctica no es posible emplear la factorización de la función objetivo empleada en la BL de la primera práctica.

Esto se debe a que los operadores de cruce empleados provocan un cambio significativo en las soluciones candidatas generadas y es necesario evaluarlas de forma estándar.

Eso provoca que las ejecuciones de los Ags puede ser mucho más lentas que las de la BL en ese caso. Hay que tener este hecho en cuenta y no dejar las ejecuciones para el último momento