Metaheurísticas

Seminario 3. Problemas de optimización con técnicas basadas en poblaciones

1. Estructura de un Algoritmo Genético y Aspectos de Implementación

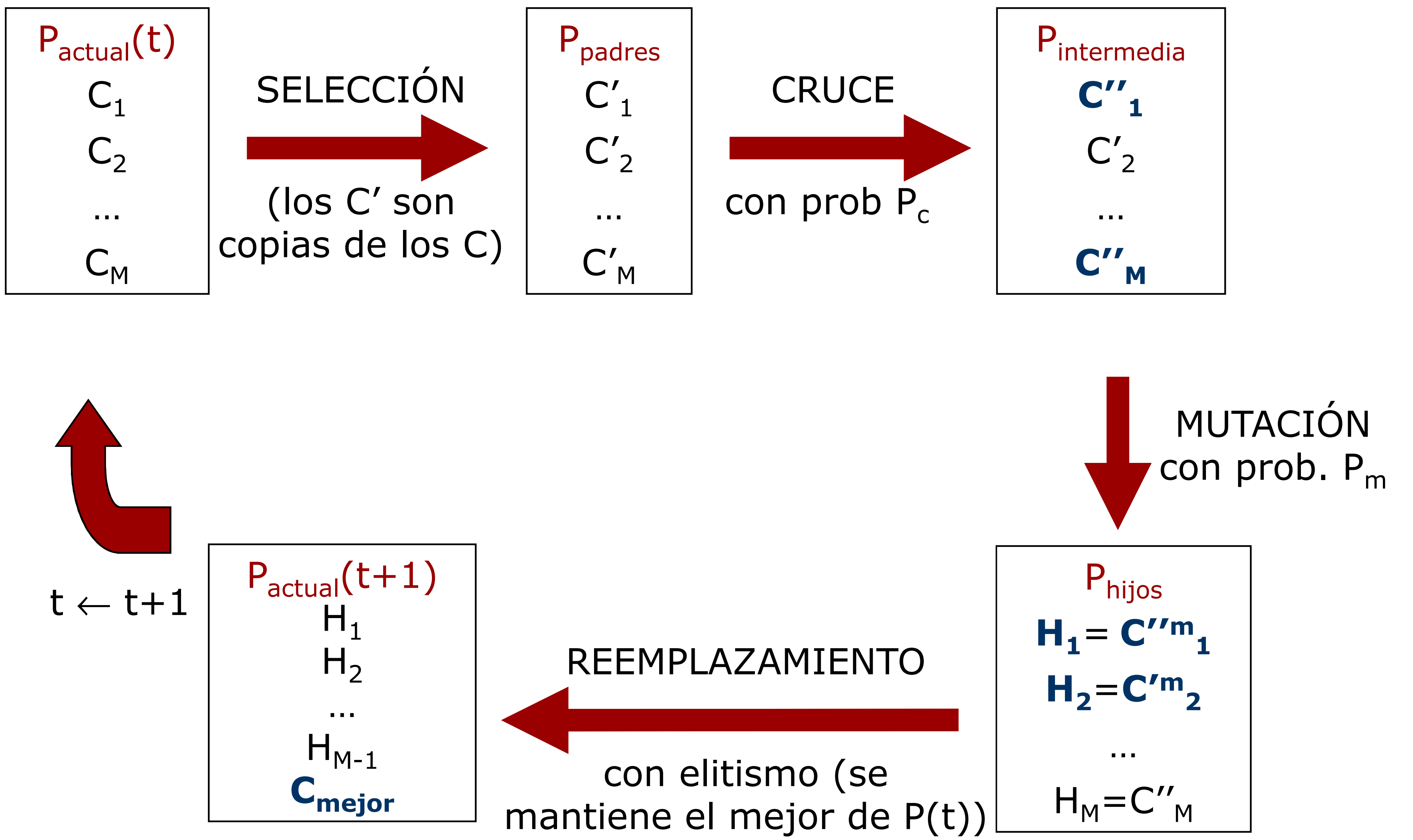
2. Problemas de Optimización con Algoritmos Genéticos
   - Asignación Cuadrática
   - Selección de Características

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 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 $u \leq P_c$, se estima a priori (al principio del algoritmo) el número de cruces a hacer en cada generación (esperanza matemática):

$$\textit{Nº esperado cruces} = P_c \cdot M/2$$

- ✔ Por ejemplo, con una población de 60 cromosomas (30 parejas) y una P_c de 0.6, cruzarán 0,6*30= 18 parejas

- ✔ De nuevo, consideramos la aleatoriedad que ya aplica el mecanismo de selección y cruzamos siempre las *Nº 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·n, es muy grande

- ✔ La P_m, definida a nivel de gen, suele ser muy baja (p.e. P_m=0.01). Eso provoca que se generen muchos números aleatorios para finalmente realizar muy pocas mutaciones

- ✔ Por ejemplo, con una población de 60 cromosomas de 100 genes cada uno tenemos 6000 genes de los cuales mutarían unos 60 (*Nº esperado mutaciones* = P_m · *nº genes población*, esperanza matemática)

- ✔ Generar 6000 números aleatorios en cada generación para hacer sólo 60 mutaciones (en media) es un gasto inútil. Para evitarlo, haremos siempre exactamente *Nº esperado mutaciones* en cada generación

Modelo Generacional: Aspectos de Implementación

- ✔ Aparte de hacer un número fijo de mutaciones, hay que decidir cuáles son los genes que mutan

- ✔ Normalmente, eso se hace también generando números aleatorios, en concreto dos, un entero en {1, …, M} para escoger el cromosoma y otro en {1, …, n} para el gen

- ✔ Existen también mecanismos más avanzados que permiten escoger el gen a mutar generando un único número real en [0,1] y haciendo unas operaciones matemáticas (ver código entregado en prácticas)

Problema de Asignación Cuadrática (QAP)

- **Problema de la asignación cuadrática, *QAP*:**

  Dadas n unidades y n localizaciones posibles, el problema consiste en determinar la asignación óptima de las unidades en las localizaciones conociendo el flujo existente entre las primeras y la distancia entre las segundas

$$QAP = \min_{S \in \Pi_N} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{S(i)S(j)} \right)$$

donde:

- ✔ *S* es una solución candidata (una posible asignación de unidades a localizaciones) representada por una permutación de *n* elementos
- ✔ $f_{ij} \cdot d_{S(i)S(j)}$ es el coste de la asignación de la unidad u_i a la localización $S(i)$ y u_j a $S(j)$, calculado como el coste del recorrido del flujo que circula entre esas dos unidades i y j cuando están situadas en las localizaciones S(i) y S(j)

Algoritmo Genético para el QAP

- Representación de orden: permutación $\pi=[\pi(1), \ldots, \pi(n)]$ en el que las posiciones del vector $i=1,\ldots,n$ representan las unidades y los valores $\pi(1), \ldots, \pi(n)$ contenidos en ellas las localizaciones

- Generación de la población inicial: 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 binario

- Operador de cruce: El basado en posición y otro a escoger: OX o PMX

- Operador de mutación: Intercambio (operador de vecino de la BL de la Práctica 1). Se generará otra posición aleatoria con la que intercambiar el contenido del gen a mutar

Algoritmo Genético para el QAP

Cruce para representación de orden basado en posición

- Genera un hijo a partir de dos padres
- Aquellas posiciones que contengan el mismo valor en ambos padres se mantienen en el hijo (para preservar las asignaciones prometedoras)
- Las asignaciones restantes se seleccionan en un orden aleatorio para completar el hijo

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

Restos: {1, 2, 4, 5, 8, 9} $\rightarrow$ Orden aleatorio: {9, 1, 2, 4, 8, 5}
Hijo = (9 1 3 2 4 7 6 8 5)

Algoritmo Genético para el QAP

Cruce para representación de orden PMX

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

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

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

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

Algoritmo Genético para la Selección de Características

- Representación binaria: un vector binario $s=(s_1, \ldots, s_n)$ en el que cada posición *i* representa una característica y su valor 0/1 indica si está o no seleccionada
- Generación de la población inicial: 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 binario
- Operador de cruce: Cruce clásico en dos puntos
- Operador de mutación: Intercambio de pertenencia de la característica correspondiente al gen a mutar (*Flip*(*s*,*i*)). (operador de vecino de la BL de la Práctica 1)