ESCUELA
POLITÉCNICA
NACIONAL

Computación Evolutiva

Jose Aguilar

aguilar@ula.ve

COMPUTACION EVOLUTIVA

- ENFOQUES ALTERNATIVOS DE BUSQUEDA Y APRENDIZAJE BASADOS EN MODELOS COMPUTACIONALES DE PROCESOS EVOLUTIVOS
- *IDEA:* BUSQUEDA ESTOCASTICA EVOLUCIONANDO A UN CONJUNTO DE ESTRUCTURAS Y SELECCIONANDO DE MODO ITERATIVO A LAS MAS APTAS

***FINALIDAD*: SUPERVIVENCIA DEL MAS APTO**

***MODO*: ADAPTACION AL ENTORNO**

COMPUTACION EVOLUTIVA

- Los esfuerzos de la CE han consistido en relacionar algunos conceptos de la teoría de la evolución con un conjunto de técnicas computacionales

- El principal aporte de la CE ha sido el uso de mecanismos de selección de soluciones potenciales y de construcción de nuevas candidatas por recombinación de características de otras ya existentes.

COMPUTACION EVOLUTIVA

- EMULACION DE PROCESOS EVOLUTIVOS:
  - POBLACION DE POSIBLES SOLUCIONES => INDIVIDUOS
  - PROCESO DE SELECCIÓN => APTITUD DE LOS INDIVIDUOS
  - PROCESO DE TRANSFORMACION => GENERACION DE NUEVOS INDIVIDUOS

•Aspectos generales de la computación evolutiva

El propósito genérico de los algoritmos en la CE es guiar una **búsqueda estocástica** haciendo evolucionar un conjunto de estructuras, seleccionando de modo iterativo las mas adecuadas.

COMPUTACION EVOLUTIVA

- FENOMENOS NATURALES SIMULADOS:
  - HERENCIA GENETICA
  - LUCHA POR LA SUPERVIVENCIA
- INSPIRACIONES TEORICAS:
  - EVOLUCION DE DARWIN
  - SELECCIÓN DE WEISMANN
  - GENETICA DE MENDELL

COMPUTACION EVOLUTIVA

- EVOLUCION DE DARWIN:

Darwin propone la selección natural como uno de los mecanismos fundamentales capaces de explicar la génesis de nuevos tipos de individuos.

- TEORÍA NEODARWINISTA:

Combina la teoría de Darwin con los aportes de la Genética, para postular el principio,

la unidad sobre la que actúa la evolución no es el individuo, sino otra de orden superior: la población.

COMPUTACION EVOLUTIVA

- EVOLUCION DE DARWIN:
  - **No todos los individuos** de una misma especie **son iguales**.
  - Algunas de estas **variaciones podrán resultar favorables** en determinados ambientes.
  - En cada generación, el **número de individuos** en una poblaciones permanece **casi constante**.
  - Los individuos mejor adaptados tendrán **más posibilidades de dejar descendientes.**
  - Como las características favorables se van acumulando (agregando), con el tiempo surgen **grandes diferencias entre el grupo original y los individuos finales**.

COMPUTACION EVOLUTIVA

- GENETICA DE MENDELL
  - En cada organismo existe un par de factores que regulan la aparición de una determinada característica (**genes**).
  - El organismo **obtiene** tales factores de **sus padres.**
  - Si un organismo posee **dos factores diferentes** para una característica dada, **uno de ellos debe expresarse** y excluir totalmente al otro.

Características de la CE

- Se trabaja con la **codificación de un conjunto de parámetros** (o un subconjunto de ellos) y no con los parámetros en si.

- Se usa una **función objetivo o función de evaluación** para definir lo que se desea optimizar.

Esta función permite evaluar la calidad de los individuos en cada generación

=> función de aptitud

Características de la CE

- Se trabaja con una **población del problema**, no con un punto simple.

- Se usan **reglas de transición probabilísticas**, no reglas deterministas

COMPUTACION EVOLUTIVA

- OBJETIVOS CONFLICTIVOS QUE SE SIGUEN:
  - EXPLOTAR LAS MEJORES SOLUCIONES
  - EXPLORAR EL ESPACIO DE BUSQUEDA
- PARADIGMAS:
  - ALGORITMOS GENETICOS (HOLLAND)
  - PROGRAMAS EVOLUTIVOS (MICHALEWICZ)
  - PROGRAMACION GENETICA (KOZA)
  - ESTRATEGIAS EVOLUTIVAS (RECHENBERG SCHWEFEL)
  - PROGRAMACION EVOLUTIVA (FOGEL)

COMPUTACION EVOLUTIVA

MACROALGORITMO:

1.- POBLACION INICIAL

2.- EVALUACION DE LOS INDIVIDUOS

3.- REPRODUCCION INICIAL

4.- REEMPLAZO

5.- CONDICION DE FINALIZACION O REGRESA A PASO 2

COMPUTACION EVOLUTIVA

ASPECTOS DE IMPLEMENTACION:

- REPRESENTACION DE LOS INDIVIDUOS
- MEDIDAS DE ADAPTACION (FUNCION ADAPTATIVA)
- MECANISMOS DE SELECCIÓN Y REMPLAZO
- INTERPRETACION DE LOS RESULTADOS
- PARAMETROS Y VARIABLES QUE CONTROLAN EL PROCESO
- OPERADORES GENETICOS A UTILIZAR

Representación de los Individuos

- **Tipos**
  - Cadenas binarias => 2^l soluciones en vector de longitud l
  - Vector de valores reales: incluye vector de desviación standard
  - Elementos del problema
  - Estados Finitos
  - Arboles

Operadores Genéticos

- Mecanismos de transformación que se utilizan para recorrer el espacio de las soluciones de un problema.

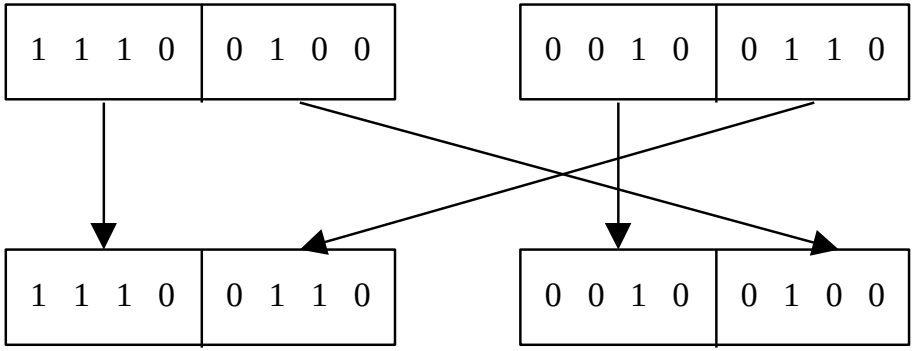
  realizan los cambios de la población durante la ejecución de un algoritmo evolutivo.

- En general, se basan en los operadores genéticos biológicos.
- Operadores genéticos clásicos:
  - Mutación
  - Cruce

Operador Cruce

- Operador, normalmente **binario**,
- Permite representar el proceso de **apareamiento natural**
- **Intercambia pedazos de información** entre diversos individuos para producir dos nuevos individuos.
- El **punto de corte** es escogido aleatoriamente.

Operador Cruce

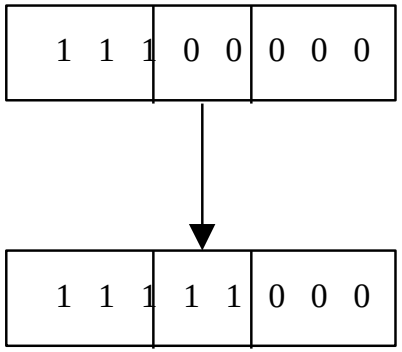


- **Tipos:**
  - Cruce multipunto
  - Cruce segmentado
  - Cruce uniforme

Operador de Mutación

- Operador **unario** que simula el proceso evolutivo que ocurre en los individuos cuando cambian su estructura genética.
- Se **seleccionan aleatoriamente componentes (genes) de un individuo para ser modificados,** también de forma aleatoria.
- A través del operador de mutación se puedan **producir cambios** en la estructura de un individuo

Operador de Mutación



- **Tipos**
  - Mutaciones sobre genes
  - Mutaciones no estacionarias
  - Mutaciones no uniformes

Operadores Avanzados

- Dominación
- Segregación
- Inversión
- Duplicación

Mecanismos de funcionamiento de los algoritmos evolutivos

- **Estrategias de selección** de los individuos susceptibles a reproducirse.
- **Estrategia de apareamiento** de los individuos en la fase de reproducción.
- **Estrategia de reemplazo.**

Mecanismos de Selección

Políticas empleadas para la conformación de los progenitores.

- **Tipos:**
  - elitesca
  - aleatoria
  - Por sorteo
  - universal (por ruleta)
  - Por torneos

Muestreo por sorteo

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

- Se calculan las probabilidades acumulativas (q_i):
  - $q_0 = 0$.
  - $q_i = q_{i-1} + p_i$.
- Se genera un numero aleatorio α entre 0 y 1.
- Se elige el individuo *i* que cumpla con la condición:

$$q_{i-1} < \alpha < q_i.$$

Muestreo universal

Se genera un numero aleatorio α y se utiliza este para crear los números:

$$\alpha_i = \frac{\alpha + i - 1}{k} \qquad \forall i = 1, k$$

Una vez obtenidos estos números, se aplica k-veces el paso 3 del muestreo por sorteo.

Muestreo por torneos

Se toman al azar *m* individuos de la población, y el mejor de estos *m* individuos formara parte de la muestra.

Mecanismos de Reemplazo

- Tipos:
  - Padre por hijo, o Reemplazo directo
  - Por similitud
  - Por inserción (,), p.e., peores por mejores
  - Por inclusión (+)

Estrategia de Apareamiento

Qué individuos con qué individuos aparear en la fase de reproducción

- Tipos:
  - Aleatorio,
  - Autofertilizacion,
  - etc.

ALGUNOS PARAMETROS Y VARIABLES QUE CONTROLAN EL PROCESO

- **Probabilidades** de uso de los operadores

- **Criterios de parada**
  - Por ejemplo, Número de generaciones

- **Tamaño y diversidad** de la población

Revisión Procedimiento general de los algoritmos evolutivos

- Generar una población inicial.
- Evaluar los individuos.
- Seleccionar, a través de algún mecanismo, ciertos individuos de la población.
- Modificar los genes de los padres seleccionados usando los operadores genéticos.
- Evaluar los nuevos individuos.
- Generar una nueva población con la existente y los individuos generados en el paso 4.
- Verificar el criterio de convergencia, o regresar al paso 3.

COMPUTACION EVOLUTIVA

- OPTIMIZACION
- ADAPTACION
- MAQUINAS INTELIGENTES
- BIOLOGIA
- CONSTRUCCION DE PATRONES

Aplicaciones

- Planeación: enrutamiento, planificación, empaquetamiento
- Diseño: sistemas digitales y electrónicos, redes neuronales, reactores, estructuras, aviones, compt. paralela
- Simulación e Identificación: modelo-comportamiento, determinar modelos
- Control: controlador, optimizador,
- Clasificación: economía, biología, procesamiento de imágenes, juegos

Algoritmos Genéticos

Algoritmos Genéticos (AGs)

METODO ESTOCASTICO DE BUSQUEDA INFORMADA DE SOLUCIONES CUASI-OPTIMAS. SE MANTIENE UNA POBLACION QUE REPRESENTA POSIBLES SOLUCIONES, LA CUAL ES SOMETIDA A CIERTAS TRANSFORMACIONES PARA OBTENER NUEVOS CANDIDATOS, Y A UN PROCESO DE SELECCIÓN SESGADO A FAVOR DE LAS MEJORES SOLUCIONES

Algoritmos Genéticos (AGs)

- **Algoritmos de búsqueda** que **emulan la evolución biológica** en el computador, siguiendo un **proceso evolutivo inteligente** sobre individuos,
- Fueron propuestos por John Holland a mediados de los **años 70**
- El objetivo final es encontrar la **mejor solución posible**, partiendo de soluciones escogidas aleatoriamente.

Algoritmos Genéticos (AGs)

Es la más conocida.

- Razones:
  - **Reúnen,** de manera natural, todas las **ideas fundamentales de la CE.**
  - **Poseen la mayor base teórica**, la cual es sencilla y con mayores posibilidades de ampliación.
  - **Requieren menor conocimiento** especifico del problema, lo que les da mayor versatilidad.

Características de los AGs

- Búsqueda **Informada.**
- Búsqueda **Basada en Población**.
- Operadores **Aleatorios**.

Generalidades de los AGs

- El procedimiento general consiste en **mantener una población de potenciales soluciones** a través de las generaciones.
- Los individuos de esa población son **cadenas de caracteres.**
- Se utilizan **operadores evolutivos como mecanismos de búsqueda**.
- El procedimiento continua hasta cumplirse algún **criterio de convergencia o parada**.
- Son el paradigma mas usado de la CE, y junto con las Redes Neuronales Artificiales, **los más usados de toda el área de Computación Inteligente**.

Implementación de los AGs

- Seleccionar la Codificación: Si suponemos que un individuo *Y* posee *m* genes se requieren para la codificación L bits

$$L = \sum_{i=0}^{m} L_i \qquad L_i = \log_2 a_i$$

gen *i* tiene a_i posibles valores (alelos).

- Seleccionar la población inicial y su tamaño.

Lo mas variada posible.

Implementación de los AGs

- Definir criterio de parada
  - *número de iteraciones* (generaciones).
  - *similitud de las estructuras*
  - *similitud de contenido*.
- Función de Evaluación

  De manera ideal, la función de evaluación coincide con el objetivo a maximizar o minimizar
- Manejar los individuos no factibles
  - *penalización*,
  - *reparación o corrección*,
  - *decodificación*

Implementación de los AGs

- **Seleccionar los operadores genéticos**

- La influencia de los operadores genéticos es fundamental en el funcionamiento de los AGs; por lo tanto, es importante hacer una **buena selección de los mismos.**
- **Habitualmente, se utilizan dos operadores**, el de cruce y el de mutación.
- No obstante, cuando se tiene conocimiento especifico del problema es posible **incorporar otros operadores genéticos, o modificar los operadores ya existentes.**

Ejemplos de Aplicación

Búsqueda en un espacio de soluciones

- Supongamos que se desea **maximizar la función:** F(x) = Sen(xπ/256) en el intervalo [0,255].
- ***Codificación.*** Como las soluciones varían entre 0 y 255, se pueden utilizar cadenas binarias de longitud 8. Por ejemplo, 00010100.

Ejemplos de Aplicación

- *Función de evaluación*. La función de evaluación es la misma función dada, y se utilizar a como función de aptitud la diferencia entre el valor máximo (1) y el valor para la función dada: g(x)=1-F(x).
- *Población inicial*.

Individuo	**x**	**F(x)**
10111101	189	0.733
11011000	216	0.471
01100011	99	0.937
11101100	236	0.243

Ejemplos de Aplicación

Problema de optimización combinatoria

- ***El Agente Viajero:*** Dadas N ciudades, el viajero de comercio debe **visitar cada ciudad una vez**, teniendo en cuenta que el **costo total del recorrido debe ser mínimo**.

G=(N, A)

N={1, ...,n}: conjunto de *n* nodos (vértices)

A={a_{ij}}: matriz de adyacencia.

d_{ij} = { ∞ Si $a_{ij} = 0$

Lij Si $a_{ij} = 1$

Lij: distancia entre las ciudades i y j.

Ejemplos de Aplicación

Formas de representar soluciones

- Suponiendo que las ciudades son numeradas desde 1 hasta n, una **solución** al problema puede expresarse a través de una **matriz de estado *E***

e_{ij} = { 1 Si la ciudad *j* fue la i^{esima} ciudad visitada

0 En otro caso

- La matriz E permite definir un **arreglo unidimensional V de dimensión *n***;

$v_{ij} = i$ Si la ciudad i fue la j^{esima} ciudad visitada

Ejemplos de Aplicación

- Función objetivo

$$F1 = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} L_{ik} e_{ij} e_{kj+1}$$

$$F2 = C\left(\left|\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij} - n\right| + \sum_{i=1}^{n}\left|\sum_{j=1}^{n} e_{ij} - 1\right| + \sum_{j=1}^{n}\left|\sum_{i=1}^{n} e_{ij} - 1\right|\right)$$

$$FC = F1 + F2$$

C = n * Max(L_{ik}) factor de penalización.

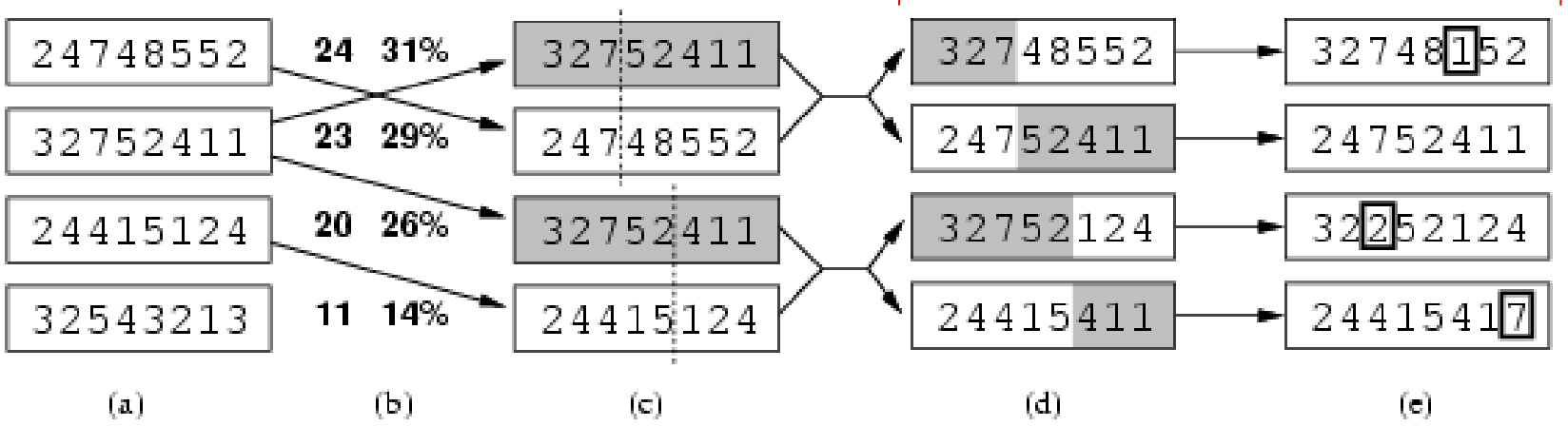
Ejemplos de Aplicación

- Codificación de los individuos:
  - Una solución viene dada como un recorrido que cumple con las restricciones del problema.
  - Se puede realizar por medio del arreglo unidimensional V de longitud *n*
- Función Objetivo: F1 o FC
- Operadores genéticos:

Si es F1 => Inversión.

Algoritmos Genéticos (AGs):
Ejemplo de evolución

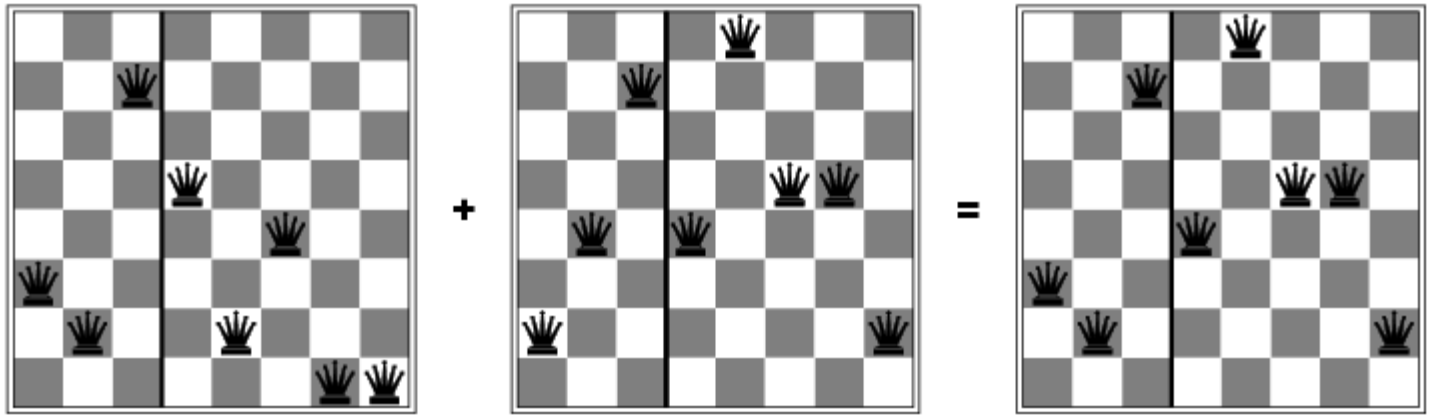
Población Inicial **Selección** **Reproducción**



Función de Aptitud **Cruce** **Mutación**

Algoritmos Genéticos (AGs):

Ejemplo de crce



Puedo tener operadores particulares a un problema dado

=> Cumplen con las restricciones del problema

LIMITACIONES

- **FUNCION DE APTITUD** SIEMPRE **POSITIVA**
- NUMERO DE **GENERACIONES FINITO**
- **CODIFICACION BINARIA** ES LA COMUN
- NO CONSIDERA **RESTRICCIONES**
- **PROBLEMAS DE DIVERSIDAD:** SUPERINDIVIDUOS, CONVERGENCIA PREMATURA, PRESION SELECTIVA, MECANISMOS DE ESCISION, CONTROLANDO EDADES, ETC.
- **PROBLEMAS DE REPRESENTACION:** COMPLETITUD, COHERENCIA, UNIFORMIDAD, ETC.

AVANCES

- **FUNCIONES MULTIOBJETIVOS**

  =>conjunto de Paretos

- **AG PARALELOS**
- **META-ALGORITMOS GENETICOS**

  =>APRENDIZAJE

- **NICHOS Y ESPECIES**

NICHOS Y ESPECIES

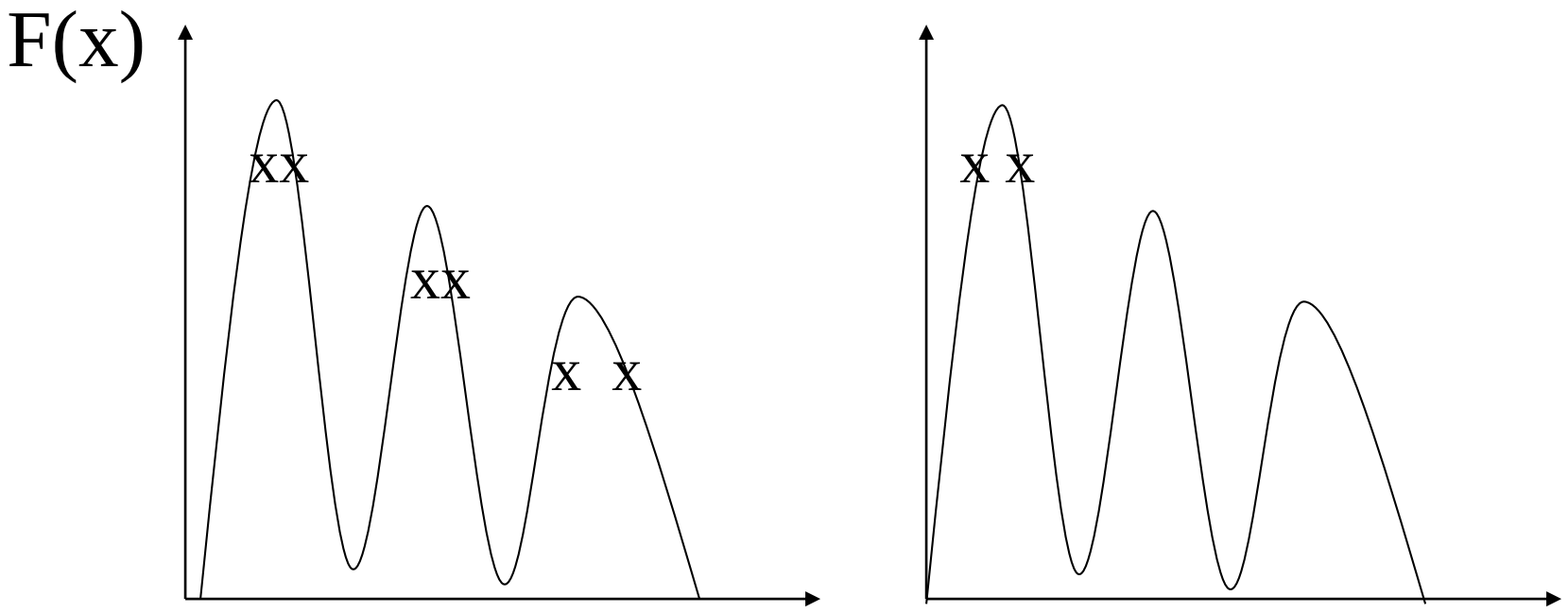
- Estables subpoblaciones: especies

- Sirven a diferentes subdominios de una función: nichos

- Metafora de compartir: vecinos

$$Fs(x_i)=F(x_i)/\sum_j s(d(x_i,x_j))$$

NICHOS Y ESPECIES



PROBLEMAS MULTIOBJETIVOS

- Muchos problemas de optimización se reducen a la consideración de **un solo criterio**, mientras que en otros problemas es necesario evaluar **varios criterios**.

  problemas multicriterios o multiobjetivos

- Las técnicas de optimización convencional, tales como los métodos basados en gradiente, y unos menos convencionales, tales como recocido simulado, son **difíciles de extender a casos verdaderamente multiobjetivos**, ya que estos no fueron diseñados para evaluar esto.

PROBLEMAS MULTIOBJETIVOS

En el mundo real, la mayor parte de los problemas tienen **varios objetivos** (posiblemente en conflicto entre sí) que se desea sean **satisfechos de manera simultánea**.

Muchos de estos problemas suelen convertirse a **mono-objetivo** transformando todos los objetivos originales, menos uno, en restricciones.

Hay tres tipos de situaciones que pueden presentarse en un problema multiobjetivo:

- Minimizar todas las funciones objetivo.
- Maximizar todas las funciones objetivo.
- Minimizar algunas funciones y maximizar otras.

PROBLEMAS MULTIOBJETIVOS

- Ejemplo: fabrica de cauchos que deben minimizar dos costos:
  - **numero de accidentes y**
  - **costo de producir cauchos**

- Escenarios:

  a = (2,10)

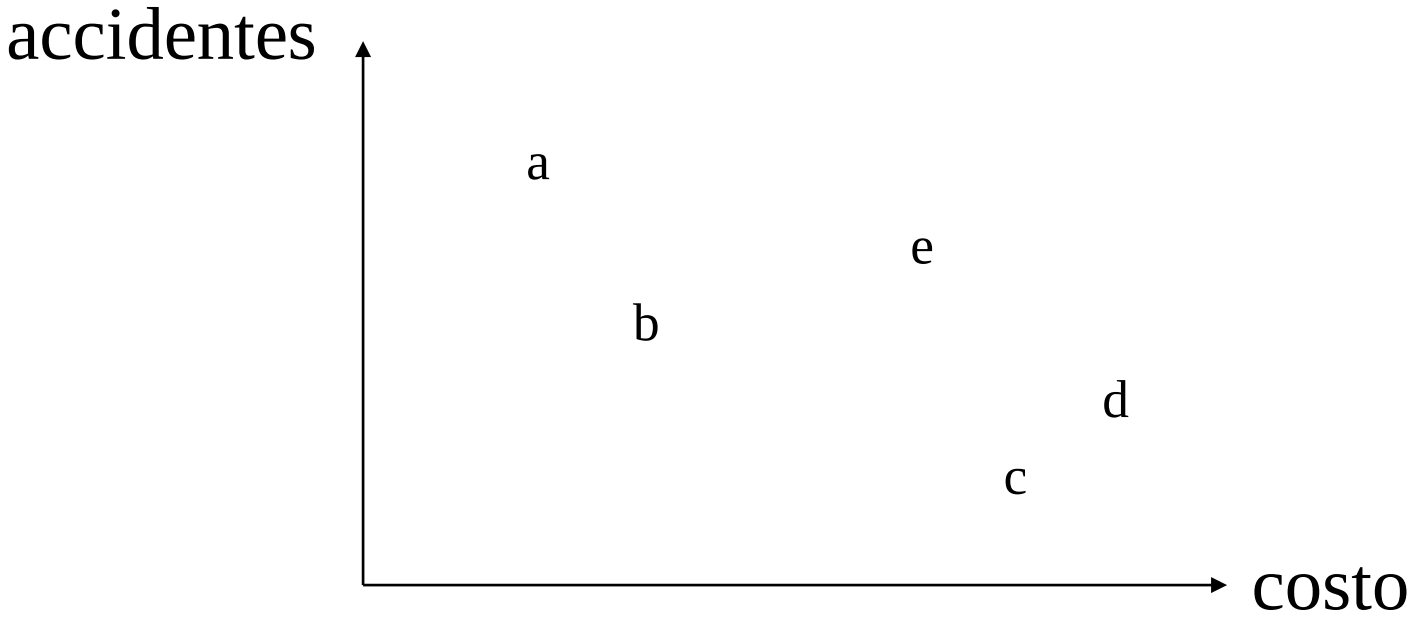
  b = (4,6)

  c = (8,4)

  d = (9,5)

  e = (7,8)

PROBLEMAS MULTIOBJETIVOS



Individuos **no dominados**: a, b, c!!!

Problemas de Localización Multiobjetivo

Escoger un conjunto de lugares para situar varias instalaciones que prestarán su servicio a un conjunto de clientes:

- La función objetivo es generalmente el costo de operación (fijo y variable)

- En algunos contextos es necesario considerar otras funciones objetivo: distancia, cobertura, tiempo de respuesta, equidad, etc.

- Criterios en conflicto
  - Costo
  - Cobertura

Problemas de Localización Multiobjetivo

Variables del modelo

I,i : Conjunto lugares de localización de las instalaciones
m instalaciones

J,j : Conjunto e índice de los lugares de demanda, n clientes

f_i : Costo fijo de operar una instalación en el lugar i.

c_{ij} : Costo de atender toda la demanda del lugar j desde la instalación i.

d_j : Demanda del cliente j.

h_{ij} : Distancia entre i y j

COMPONENTES DEL PROBLEMA DE COBERTURA

D_{max} : Distancia máxima de cobertura.

Q_j : $\{i: h_{ij} \leq D_{max}\}$, conjunto de instalaciones que pueden atender la demanda del nodo j cumpliendo con la distancia máxima de cobertura.

Problemas de Localización Multiobjetivo

Variables de decisión

$$y_i = \begin{cases} 1, & \text{si se abre la bodega } i, \\ 0, & \text{de lo contrario.} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{si el cliente } j \text{ es atendido por la bodega } i, \\ 0, & \text{de lo contrario.} \end{cases}$$

Problemas de Localización Multiobjetivo

Función objetivo

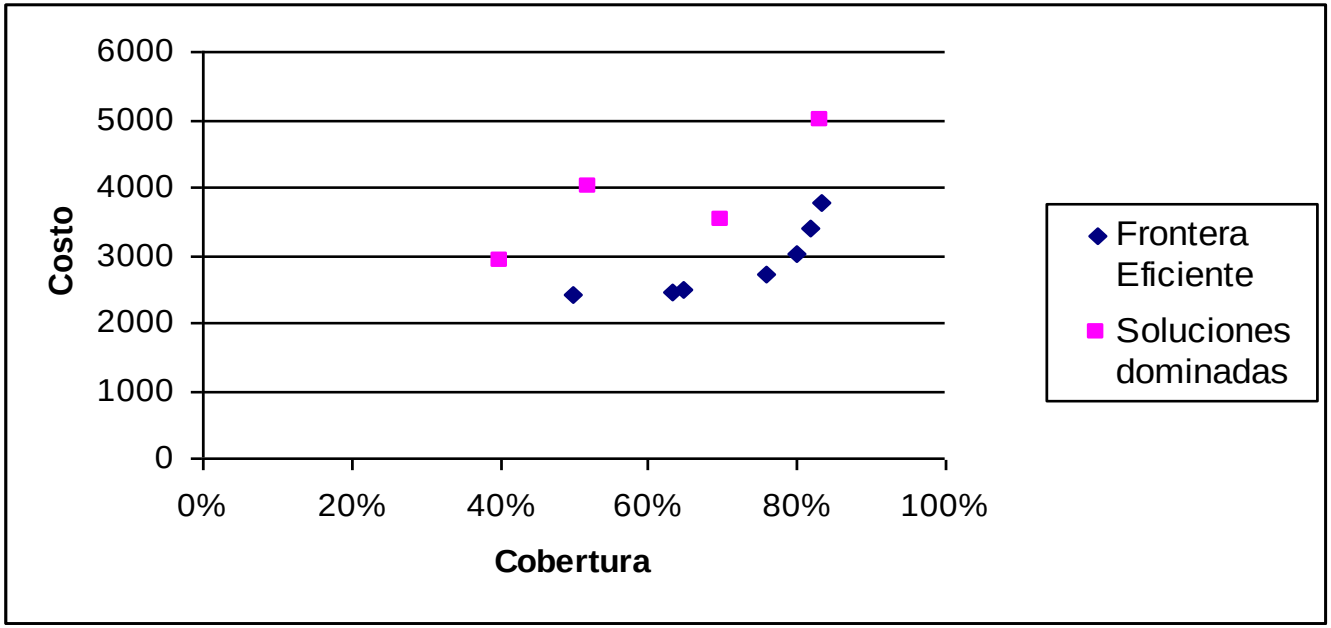
$$\begin{aligned} \text{mín} \quad z_1 &= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}x_{ij} + \sum_{i\in\mathcal{I}} f_i y_i \\ \text{máx} \quad z_2 &= \sum_{j\in\mathcal{J}} d_j \sum_{i\in Q_j} x_{ij} \end{aligned}$$

Restricciones

$$\begin{aligned} \sum_{i\in\mathcal{I}} x_{ij} &= 1 \quad , \quad j \in \mathcal{J} \\ x_{ij} &\leq y_i \quad , \quad j \in \mathcal{J}, \quad i \in \mathcal{I} \\ x_{ij} &\in \{0,1\}, \quad j \in \mathcal{J}, \quad i \in \mathcal{I} \\ y_i &\in \{0,1\}, \quad i \in \mathcal{I} \end{aligned}$$

Generalidades Optimización Multiobjetivo

- Objetivos en conflicto.
- Se busca un **conjunto de soluciones eficientes**, en las cuales no sea posible **mejorar el valor de una de las funciones objetivo sin deteriorar el desempeño de la otra.**



PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

- A diferencia de la optimización de un único objetivo, la solución a un problema multiobjetivo no es un único punto, sino una familia de puntos conocida como el "Conjunto Optimo de Pareto".
- Cada punto en esa superficie es óptimo en el sentido de que no pueden realizarse mejoras en un componente del vector de costo que no conduzca a la degradación en al menos uno de los componentes restantes.
- La noción de optimalidad de Pareto es sólo un primer paso hacia la solución práctica de un problema multiobjetivos; también envuelve la escogencia de una única solución compromiso del "Conjunto Optimo de Pareto" de acuerdo a alguna información de preferencia.

PROBLEMAS MULTIOBJETIVOS: TEORIA DE PARETO

Decimos que un vector de variables de decisión $x^* \in \alpha$ (α es la zona factible) es un **óptimo de Pareto** si no existe otro $x \in \alpha$ tal que

$$f_i(x) \leq f_i(x^*) \text{ para toda } i = 1, \ldots, k$$

x^* es un óptimo de Pareto si no existe ningún vector factible de variables de decisión $x^* \in \alpha$ que **decremente algún criterio sin causar un incremento simultáneo en al menos un criterio.**

Desafortunadamente, este concepto casi siempre produce no una solución única sino un conjunto de ellas a las que se les llama **conjunto de de Pareto.**

Los vectores x^* correspondientes a las soluciones incluidas en el conjunto de óptimos de Pareto son llamados **no dominados**.

La gráfica de las funciones objetivo cuyos vectores no dominados se encuentran en el conjunto de óptimos de Pareto se denominan **frente de Pareto.**

PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

"Conjunto Optimo de Pareto" esta compuesto por elementos que son soluciones **no-inferiores (no-dominados)** para el problema multiobjetivos:

Concepto de Inferioridad: Un vector **u** = $(u_1,...,u_n)$ se dice es inferior a **v** = $(v_1,...,v_n)$ si **v** es parcialmente menor que **u** (**v** p < **u**); es decir:

$i = 1,...,n$, $v_i \ \ u_i$ $\quad$ $i = 1,...,n : v_i < u_i$

Concepto de Superioridad: Un vector **u** = $(u_1,...,u_n)$ se dice es superior a **v** = (v_1,....,v_n) si **v** es inferior a **u**.

ESQUEMAS RESOLUCION PROBLEMAS MULTIOBJETIVOS

- No basadas en Pareto
  - Suma lineal de pesos
  - VEGA
  - Método ε-constraint
  - Teoría de juegos
- Basadas en Pareto
  - Jerarquización de Pareto “pura”
  - PAES (Pareto Archived Evolution Strategy)
  - NSGA y NSGA II (Nondominated Sorting Genetic Algorithm)
  - SPEA (Strength Pareto Evolutionary Algorithm).

PROBLEMAS MULTIOBJETIVOS

ENFOQUES BASADOS EN LA DIVISIÓN DE LA POBLACIÓN

1.- Dividir población en *n* subpoblaciones $S_1,\ldots, S_n$.

% se va a iterar un AG con una FO distinta en c/u.

2.- Definir vector de prioridad para las FOs.

3.- Repetir Hasta (Evaluar todas las FOs)

3.1 Para cada S_i, $\forall$ i = 1,...,*n*, usar la FO correspondiente según el vector de prioridad.

3.1.1 Iterar el AG para esa FO_i según un número *y* dado de iteraciones en cada S_i.

3.1.2 Reemplazar *X* % de individuos en cada población S_i que cumplen mal la FO actual.

PROBLEMAS MULTIOBJETIVOS

ENFOQUES BASADOS EN POBLACIÓN TOTAL

1.- Inicio.

2.- Repetir Hasta (Que c/FO sea escogida un mínimo número de veces ó población converja).

2.1 Escoger una FO (aleatoriamente o cíclicamente).

2.2 Iterar el AG en la población usando la FO escogida.

PROBLEMAS MULTIOBJETIVOS

ENFOQUES BASADOS EN LA TEORIA DE PARETO

Para obtener soluciones no-inferiores se pueden usar los AGs:

- manteniendo una población de soluciones, los AGs pueden buscar muchas soluciones no-inferiores en paralelo.

ENFOQUES BASADOS EN LA TEORIA DE PARETO

Enfoque en dos fases:

1. Se clasifican los individuos en **no-inferiores o no (no-dominados o dominados)**, para cada una de las FOs que conforman la FMO.
2. Se utilizan los individuos no-dominados como los **padrotes** para la fase de reproducción.

Macro Algoritmo:

1.- Generar población.

2.- Repetir Hasta que población generada sea igual a la anterior o sea homogénea.

2.1.- Fase de Clasificación: Agrupar individuos en dominados y no-dominados para c/FO.

2.2.- Fase de Optimización: Generar nueva población según un AG clásico.

ENFOQUES BASADOS EN LA TEORIA DE PARETO

Fase de Clasificación:

1.- Generar población inicial.

2.- Evaluar los individuos para c/FO y establecer su clasificación en c/FO.

3.- Seleccionar individuos no-dominados (Según c/u de las FOs).

4.- Asignar rango 1 a los individuos no-dominados y remover de la disputa, luego se encuentra un nuevo conjunto de individuos no-dominados con rango 2, y así sucesivamente. Esto se hace para c/FOs.

Fase de Optimización:

1.- Seleccionar los individuos para reproducción por algún mecanismo clásico (ruleta, elitesco).

2.- Reproducir con operadores clásicos, usando los individuos no-dominados como los padrotes.

ESQUEMAS RESOLUCION PROBLEMAS MULTIOBJETIVOS

Input : N: population size

N': archive size

T: maximum number of generations

Output: A: nondominated set

1. **Initialization :** Generate an initial population P_0 and create the empty archive (external set) $P_0' = ;$. Fix t = 0 (i.e. first generation).
2. **Fitness Assignment**: Compute fitness values for individuals in P_t and P_t'.
3. **Environmental Selection**: Find the nondominated individuals in P_t and P_0' and copy them in P_{t+1}'. If the size of P_{t+1}' exceeds N', then reduce P_{t+1}' by a truncation operator. Conversely, if the size of P_{t+1}' is smaller than N', then fill P_{t+1}' with dominated individuals in P_1 and P_t'.
4. **Termination:** If t >= T or any other stopping criteria is satisfied, then A is the set of nondominated individuals, i.e. optimal solutions. **Stop**.
5. **Mating Selection:** perform binary tournament selection with replacement on P_{t+1}' to fill the mating pool.
6. **Variation:** apply crossover and mutation to the mating pool, generating population P_{t+1}'. Increase the generation counter and **go to step 2**.

Paralelismo Implícito

- La eficacia a los AGs se basa en que, aunque el AG sólo procesa n estructuras en cada generación, se puede probar que, bajo hipótesis muy generales, se procesan de modo útil al menos n^3 esquemas.
- Este paralelismo implícito se consigue sin ningún dispositivo o memoria adicionales, sólo con la propia población.
- Paralelismo proporcional al tamaño de la población
- Problemas en performance generados por sincronización y envío de mensajes

Paralelismo en AG

Repetir

- Para cada individuo i
  - Evaluar f(i)
  - Transmitir f(i) para todos los individuos j en el vecindario
  - Elegir un individuo j para combinar basado en fitness
  - Pedir el individuo j
  - Reproducir usando los individuos i y j

Hasta que la variación en la población es pequeña

Modelos isla

- 6400 individuos en 64 procesadores, por ejemplo
  - Dividir la población total en subpoblaciones de 100 individuos cada una
- Cada subpoblación ejecuta un algoritmo genético
- Cada x cantidad de generaciones, las subpoblaciones intercambian algunos individuos
- Si un gran número de individuos migra en cada generación, se pierden las diferencias entre las islas
- Si la migración es poco frecuente, podría llevar a que cada población converja prematuramente

PROGRAMACIÓN GENÉTICA

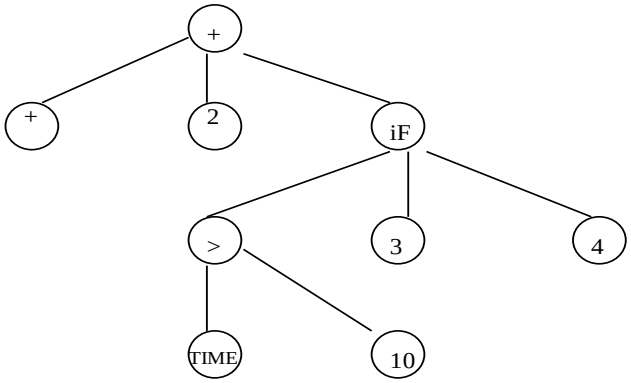
Consiste en utilizar la metodología de CE no para encontrar soluciones a problemas, sino para encontrar el **mejor procedimiento para resolverlos**.

Las estructuras que se someten a evolución **codifican los algoritmos o programas** que, al ser ejecutados, **determinan soluciones.**

PROGRAMACIÓN GENÉTICA

La representación de los individuos se hace a través de *árboles de análisis.*

En la practica, se usa el lenguaje LISP: Por ejemplo, la siguiente expresión: *(+ 1 2 (IF (> TIME 10) 3 4)),* su árbol es:



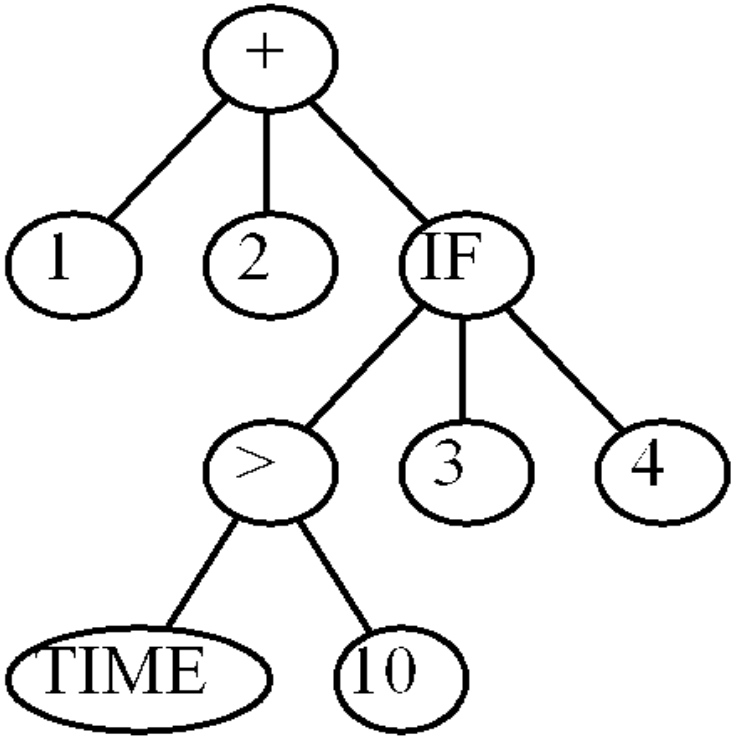
PROGRAMACIÓN GENÉTICA

```
int foo (int time)
{
    int temp1, temp2;
    if (time > 10)
        temp1 = 3;
    else
        temp1 = 4;
    temp2 = temp1 + 1 + 2;
    return (temp2);
}
```

Time	Output
0	**6**
1	**6**
2	**6**
3	**6**
4	**6**
5	**6**
6	**6**
7	**6**
8	**6**
9	**6**
10	**6**
11	**7**
12	**7**

PROGRAMACIÓN GENÉTICA

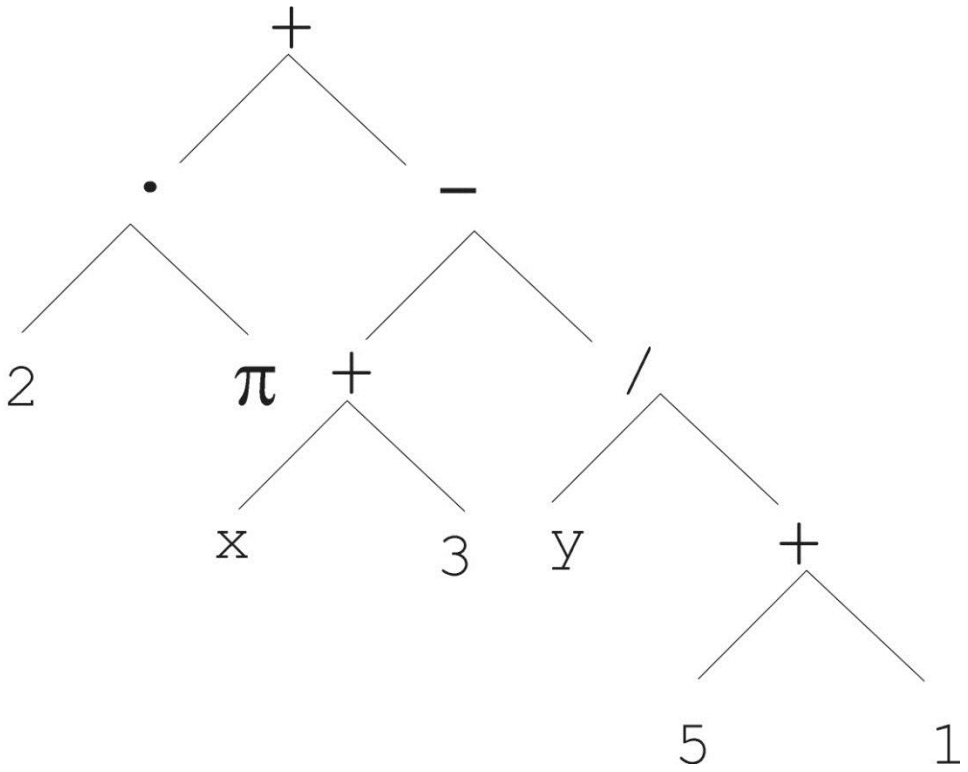
Representación de un programa




```
(+ 1 2 (IF (> TIME 10) 3 4))
```

PROGRAMACIÓN GENÉTICA

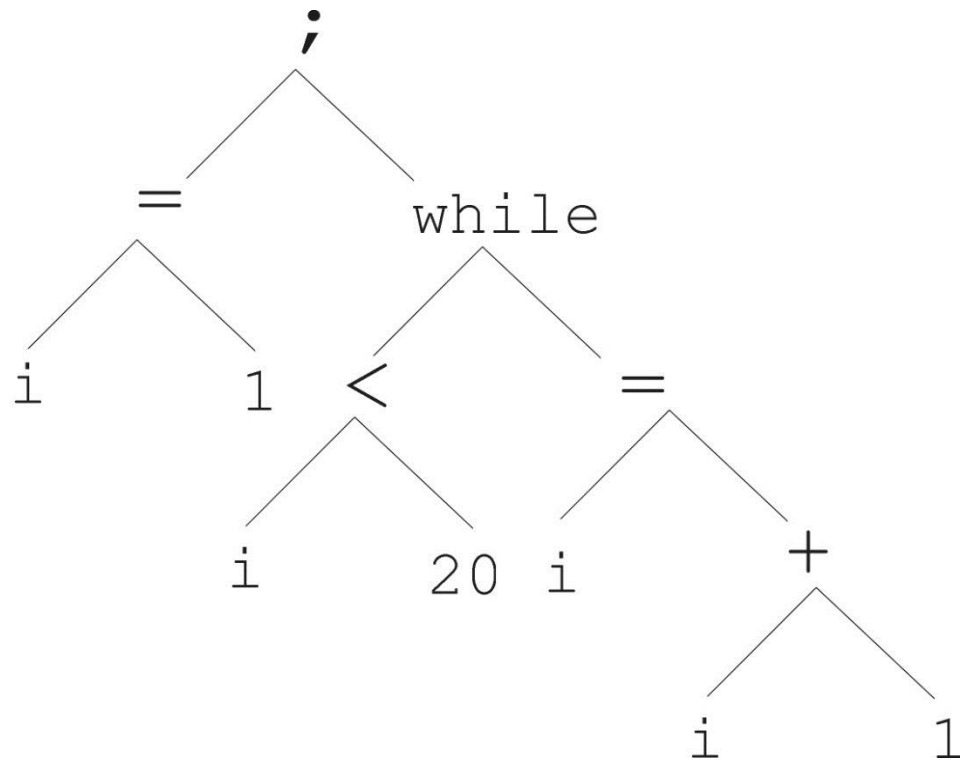
Representaciones de una fórmula



$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

PROGRAMACIÓN GENÉTICA

Representaciones basadas en árboles




```
i =1;
while (i < 20)
{
        i = i +1
}
```

PROGRAMACIÓN GENÉTICA

La codificación utiliza un conjunto de *símbolos de funciones* y otro de *símbolos terminales* (átomos) adecuados para la solución del problema dado.

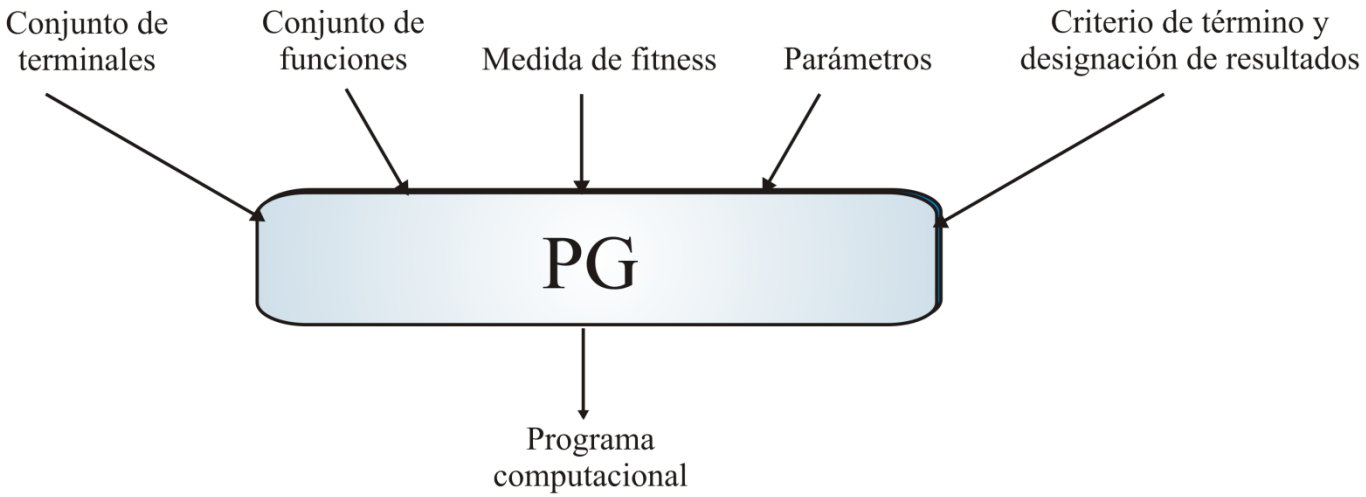
Las funciones pueden ser operaciones aritméticas, funciones matemáticas, funciones lógicas, o funciones especificas del dominio.

PROGRAMACIÓN GENÉTICA

ELEMENTOS:

- CONJUNTO DE **TERMINOS**
- CONJUNTO DE **FUNCIONES**
- MEDIDA DE **APTITUD**
- **PARAMETROS** DE CONTROL
- CRITERIO DE **CULMINACION**

PROGRAMACIÓN GENÉTICA



Determinar

- El **conjunto de terminales**
- El **conjunto de funciones**
- La **medición del fitness**
- Los **parámetros para la ejecución**
- El **método para designar un resultado** y
- Un criterio para **terminar una ejecución**

Conjuntos de Funciones y Terminales

Los nodos en el árbol pueden ser:

- **Conjunto de Funciones F**
  - El conjunto de todos los posibles nodos internos en el árbol
  - Todas las funciones tienen una ariedad ≥ 1.

- **Conjunto de terminales T**
  - El conjunto de todos los posibles nodos hoja en el árbol

El espacio de búsqueda del problema son todas las posibles combinaciones de funciones y terminales

Parámetros de control

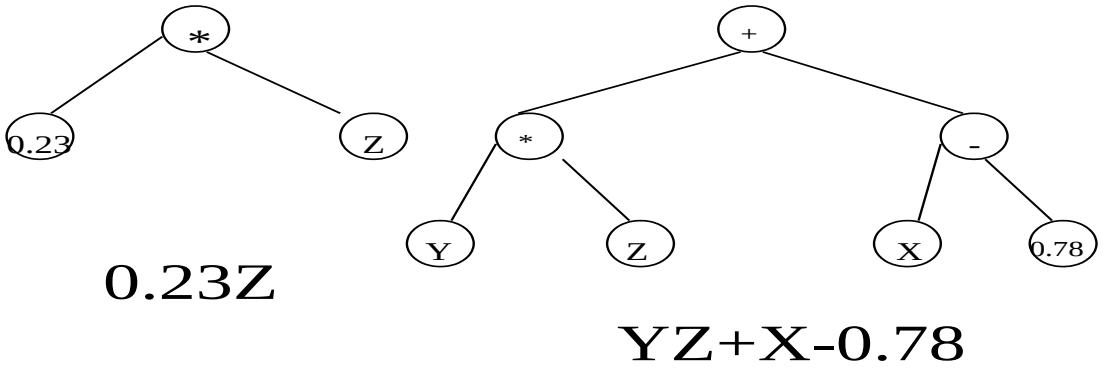
- **G:** número máximo **de generaciones** a producir
- **M:** tamaño de la **población**
- **P_C:** probabilidad de **cruzamiento**
- **P_M:** probabilidad de **mutación**
- **D_{max} tamaño** máximo (medido por **profundidad**) de los individuos creados por las operaciones de evolución darwiniana
- **$D_{inicial}$ tamaño** (medido por profundidad) de los individuos creados en la población inicial

PROGRAMACIÓN GENÉTICA

EJEMPLO:

CONJ. TERMINOS: {X,Y,Z, REALES}

CONJ. FUNCIONES: {+,-,*,IF}



Función de Evaluación

Función de Fitness "de Error"

$$f_p = \sum_{i=1}^{n} |p_i - o_i|$$

p : Algoritmo
p_i : la salida del programa p en el $i^{ésimo}$ caso de fitness
o_i : la salida del $i^{ésimo}$ ejemplo en el caso de fitness
f_p : el fitness de p.
n : número de casos de fitness

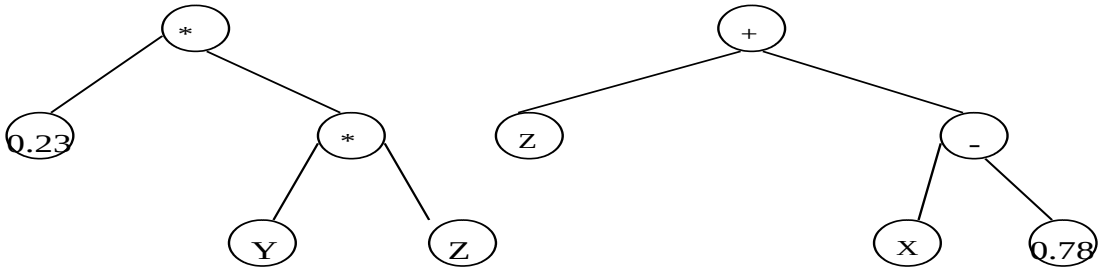
- Función de evaluación de error cuadrático $f_p = \sum_{i=1}^{n} (p_i - o_i)^2$

PROGRAMACIÓN GENÉTICA

OPERADOR CRUCE

0.23Z YZ+X-0.78

=> 0.23YZ Z+X-0.78

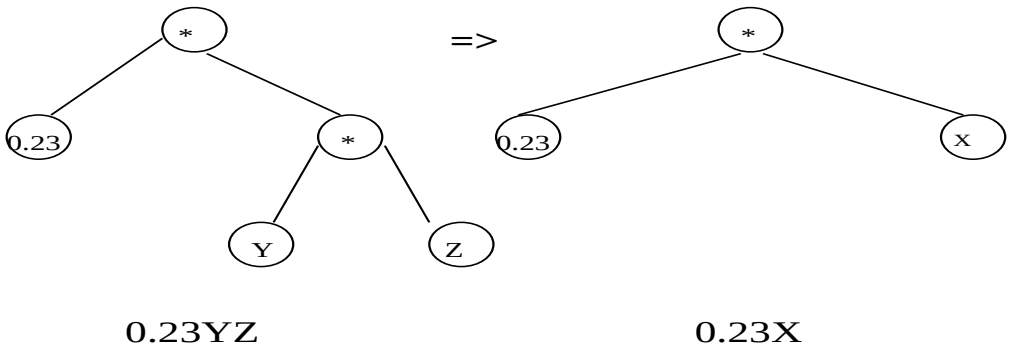


0.23YZ Z+X-0.78

PROGRAMACIÓN GENÉTICA

OPERADOR MUTACION

0.23YZ



PROGRAMACIÓN GENÉTICA

EJEMPLO: PROBLEMA DE REGRESION DE FUNCION DESCONOCIDA

VAR. IND: X VAR DEP: Y

X	Y
-1	0.178
-0.9	-0.169
…	
0	0.00
…	

PROGRAMACIÓN GENÉTICA

PROBLEMA: ENCONTRAR Y=F(X)

TERMINOS:X

FUNCIONES: +,-,*,SIN COS, EXP, LOG

FUNC. ADAPTAT.: ERROR

GENER 0:	X+ LOG(2X)+X	**fa: 6.05**
	COS(COS(+(-(*XX))))	**fa: 23,67**
GENER 34:	$X^4+X^3+X^2+X$	fa: 0

PROGRAMACIÓN GENÉTICA

Problema de la mochila

$$Max\ Z = \sum_{j=1}^{n} p_j x_j$$

Sujeto a

$$\sum_{j=1}^{n} w_j x_j \leq W$$

$$x_j \in \{0,1\}, j = 1,2,3 \ldots . n$$

Funciones Terminal	Descripción
FALSO/VERDADERO/NULO	**Valores lógicos**
GT_W	**Selecciona elemento de mayor Peso.**
LO_W	**Selecciona elemento de menor Peso.**
GT_P	**Selecciona elemento de mayor Beneficio.**
LO_P	**Selecciona elemento de menor Beneficio.**
LO_R	**Selecciona elemento de menor Eficiencia.**
ANY	**Selecciona elemento aleatorio.**
Seq (subárbol P1, subárbol P2)	**Operador que recibe y ejecuta dos subárboles (sub-programas) en secuencia, no retorna resultados.**
While(subárbol Cond, subárbol P1)	**Operador que produce una ejecución iterativa sobre el segundo subárbol, según se cumpla la condición establecida en el primer subárbol. Se maneja la condición con dos variables extras, las cuales son, número de veces que se ejecutó el ciclo "while" sin tener un efecto real en la configuración de la instancia, denominado ITER_SIN_EFECTO. Este valor se fijó en *n* (número de objetos de la mochila) y número máximo de iteraciones de ciclo "while", denominado ITER_MAX. Este valor fue fijado en 2*n.**
Put (subárbol P1)	**Agrega un elemento no seleccionado a la mochila, si el elemento no es válido o ya fue seleccionado, esta función no realiza ninguna acción.**
Quitar (subárbol P1)	**Remueve un elemento que ha sido seleccionado previamente**
Probar (subárbol P1)	**Verifica si la mochila continúa siendo válida al intentar insertar un elemento a la mochila.**
SiMejora(subárbol P1)	**Intercambia el objeto de menor eficiencia insertado en la mochila LO_R por P1, siempre que se cumplan las siguientes condiciones, que al intercambiar LO_R por P1, la mochila sigue estando válida o P1 es mejor que LO_R.**

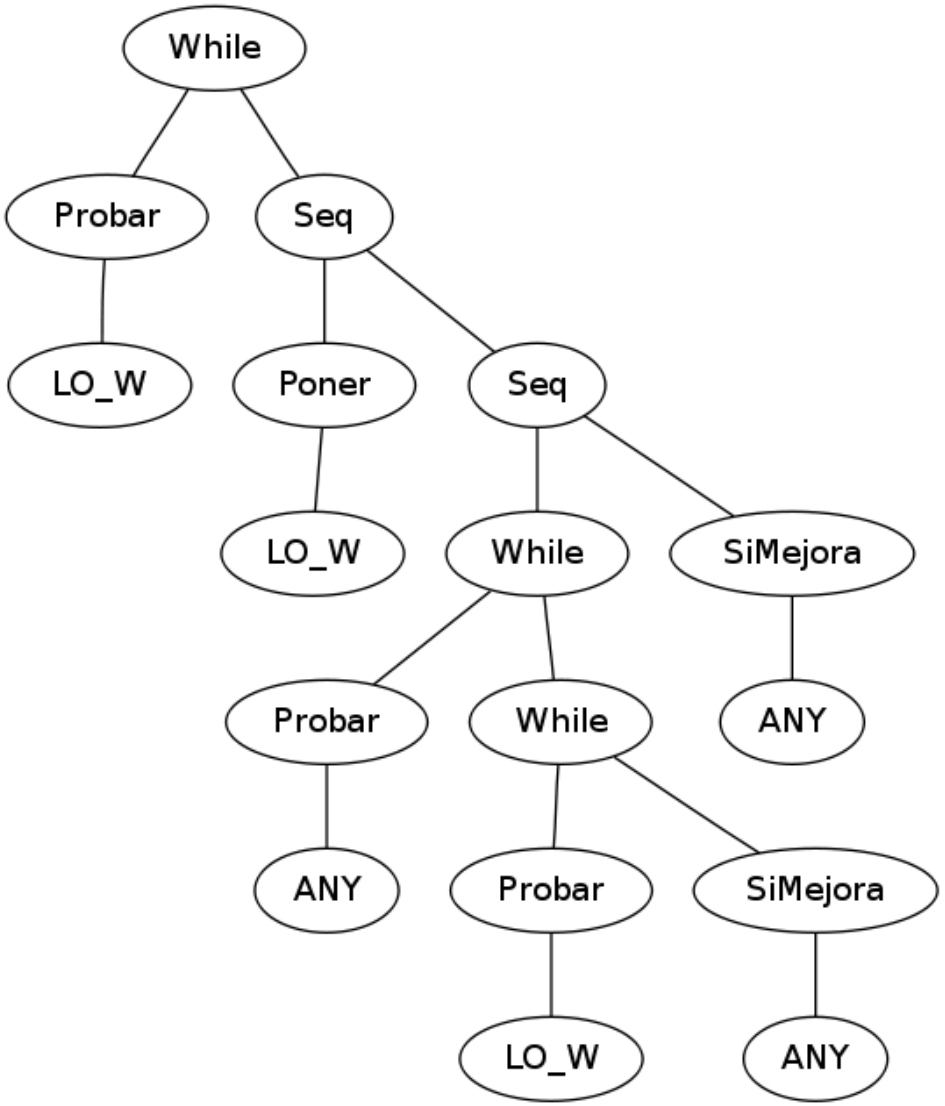
PROGRAMACIÓN GENÉTICA

Problema de la mochila

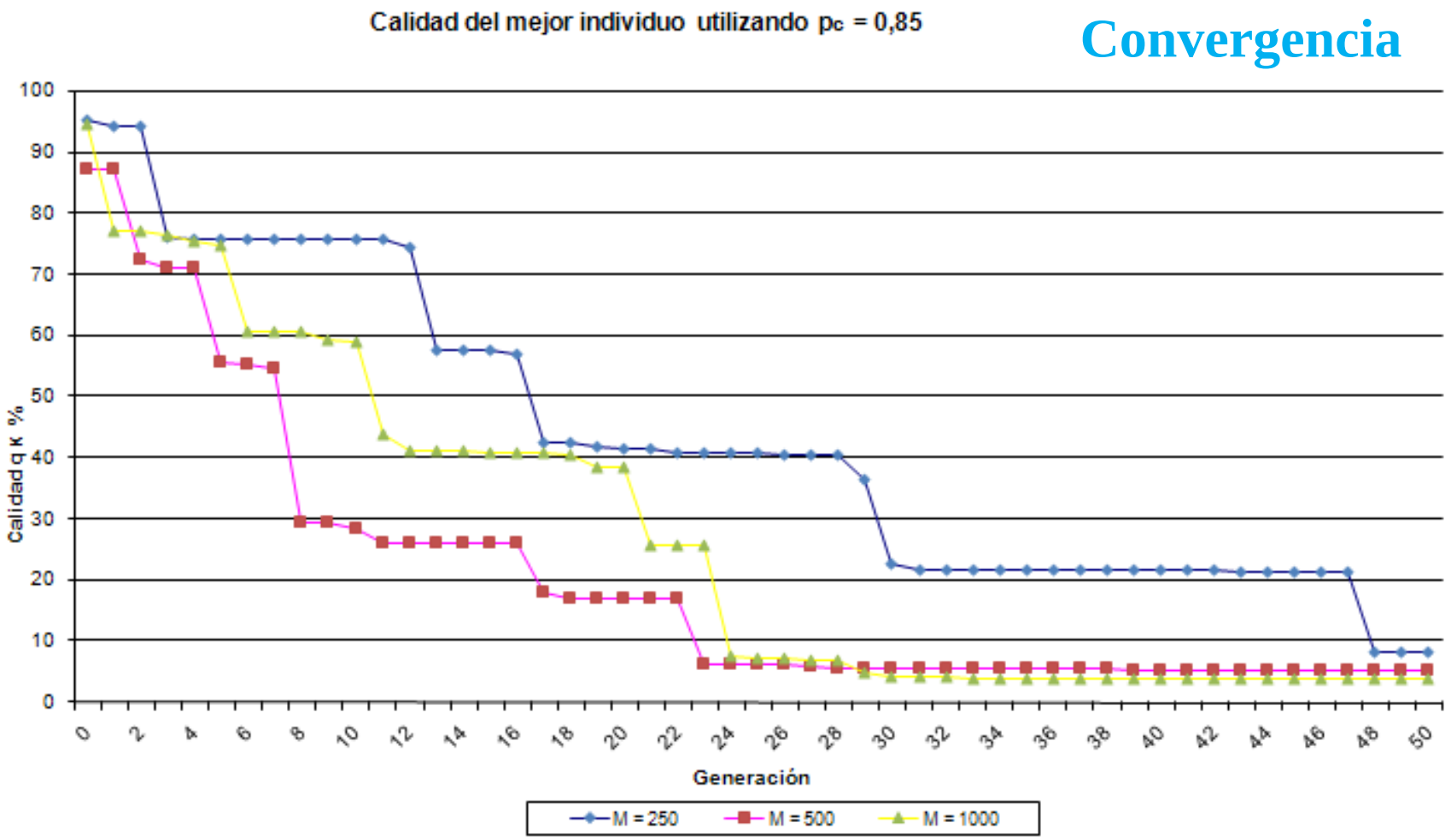
Tamaño de la población (M)	**500**
Numero máximo de generaciones (G)	**50**
Probabilidad de cruzamiento (Pc)	**85%**
Probabilidad de mutación (Pm)	**0%**

Problema de la mochila

Código Inferido



Problema de la mochila



Convergencia

PROGRAMACIÓN GENÉTICA

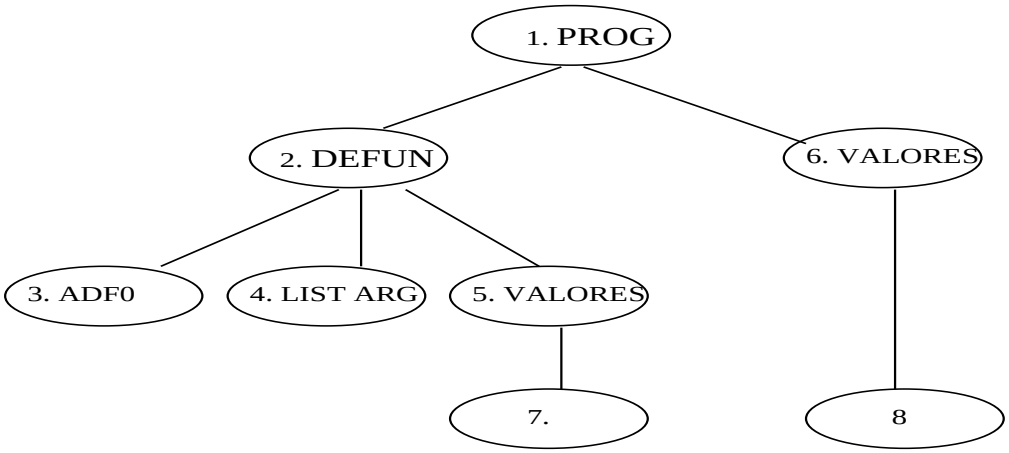
ADF (Automatically Defined Functions)

- DESCUBRIMIENTO AUTOMATICO DE UNIDADES FUNCIONALES
- **INSPIRACION:** HUMANOS ESCRIBEN FUNCIONES/RUTINAS QUE AGRUPAN PORCIONES DE INSTRUCCIONES QUE PUEDEN SER LLAMADAS VARIAS VECES DESDE PROGRAMAS
  - DESCOMPOSICION DE PROBLEMAS
  - RESOLUCION DE SUBPROBLEMAS
  - RECUPERACION DE LAS SOLUCIONES PARCIALES

PROGRAMACIÓN GENÉTICA

ADF (Automatically Defined Functions)

- DEFINICION DE UN ADF

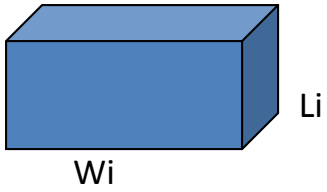


1. RAIZ
2. INICIO RAMA DEF. FUNCION
3. NOMBRE ADF
4. LISTA ARGUMENTOS
5. VALORES A REGRESAR
6. VARIAB. QUE GUARDAN RES.
7. CUERPO ADF
8. CUERPO PROGRAMA

PROGRAMACIÓN GENÉTICA

EJEMPLO ADF

- PROGRAMA CALCULA DIF ENTRE 2 CUBOS



ENFOQUE CLASICO: (-(*(* W0 L0)) H0) (* (* W1 L1) H1))

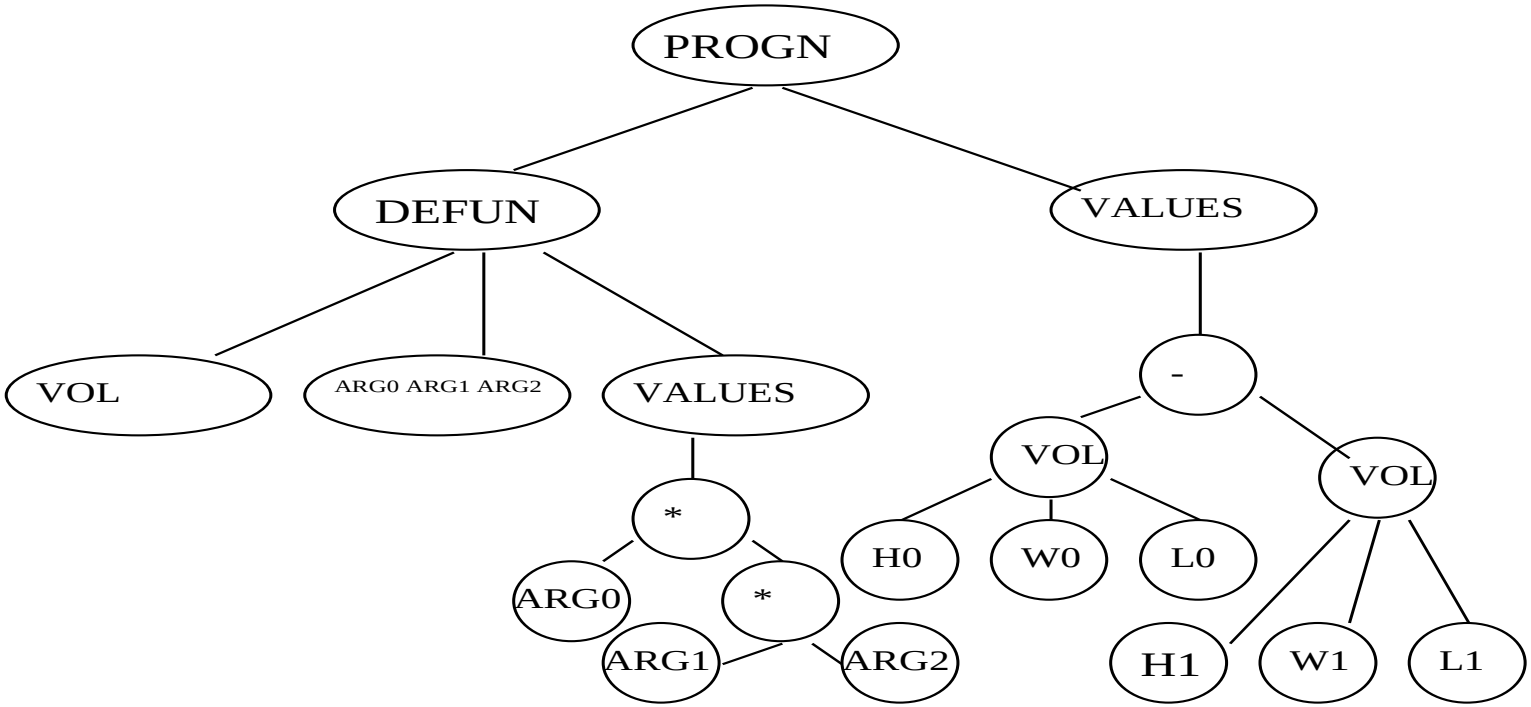
ENFOQUE CON ADF: (PROG (DEF (VOL ARG0 AR1 ARG2)

VALUES (*ARG0 (* ARG1 ARG2))))

(VALUES (- VOL L0 W0 H0)

(VOL L1 W1 H1))))

PROGRAMACIÓN GENÉTICA

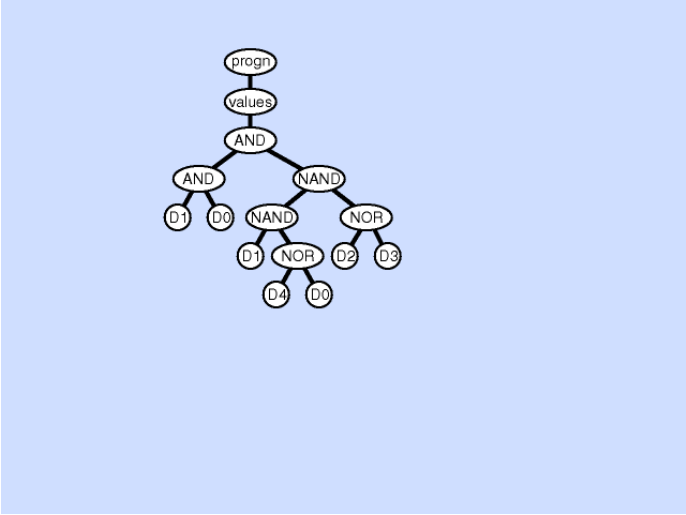


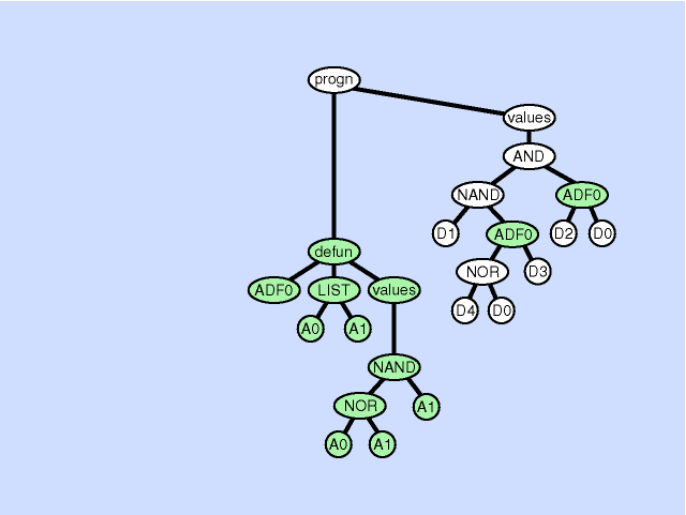
PROGRAMACIÓN GENÉTICA

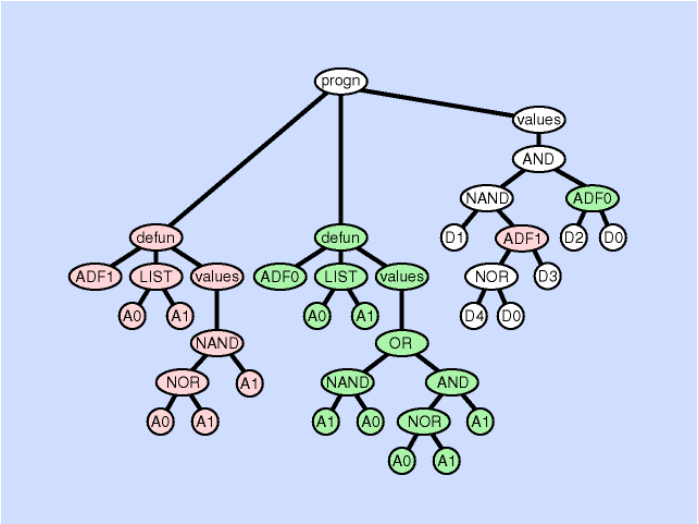
ELEMENTOS DE UN ADF

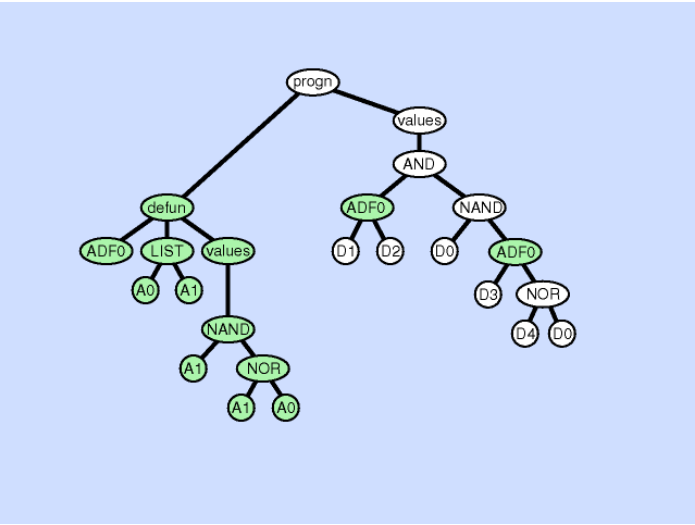
- NUMEROS DE ADFs
- SUS ARGUMENTOS
- LAS REFERENCIAS JERARQUICAS
- OPERADORES
  - **CREACION:** ADF O INVOCACION
  - **DUPLICACION:** ADF O INVOCACION
  - **ELIMINACION:** ADF O INVOCACION

Operaciones que alteran la Arquitectura









PROGRAMACIÓN GENÉTICA

ADL (AUTOMATICALLY DEFINED LOOP)

REPETICION DE OPERACIONES PREESTABLECIDAS

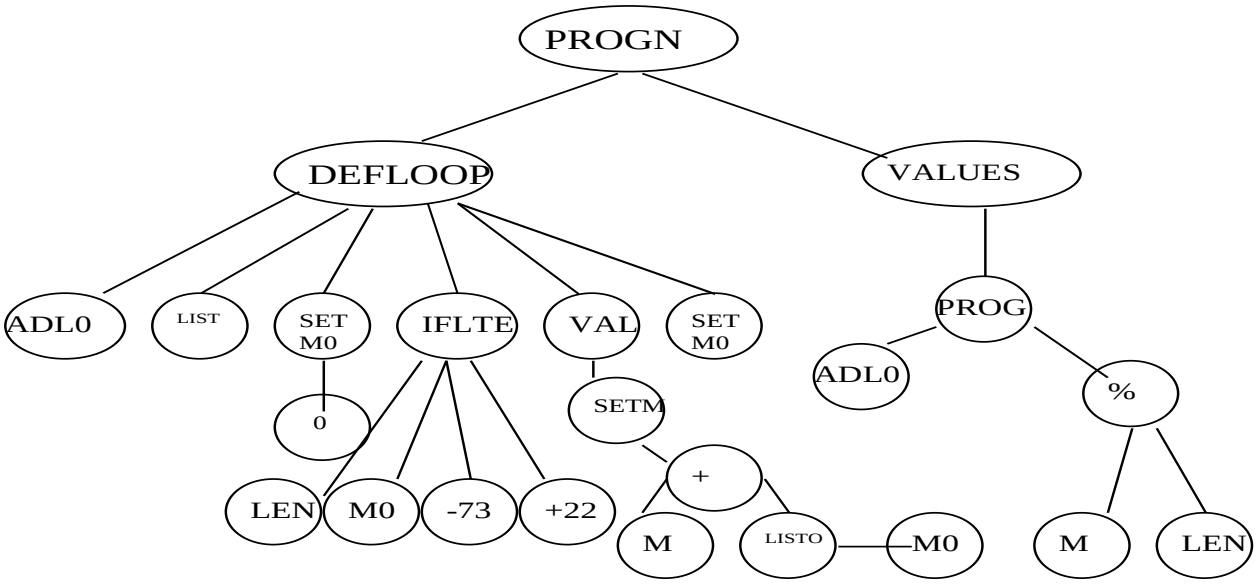
```
for (i=0; i<len; i++)
    m=m+v[i];
```

```
for (LIB; LCB; LUB)
    LBB;
```

PROGRAMACIÓN GENÉTICA

ADL

- rama de inicio lazo
- rama con cuerpo
- ramas con condición del lazo
- ramas con actualización lazo



PROGRAMACIÓN GENÉTICA

ADL

```
M0=0;
for (i=0; i<LEN;i++
    M0=M0+V[i]
AVERAGE=M0/LEN
```

Pueden haber múltiples ADL en un programa (no anidados entre ellos)

No poseen argumentos y pueden llamar a ADF

PROGRAMACIÓN GENÉTICA

ADL

Operadores

Creación

=> Tomar un pedazo del código y crear ADL

Eliminación

=> regeneración aleatoria del pedazo

=> modificación conjunto terminal y funciones

Duplicación

=> Escoger un ADL existente y modificar nombre, etc.

=> Llamados al ADL viejo son aleatoriamente modificados

PROGRAMACIÓN GENÉTICA

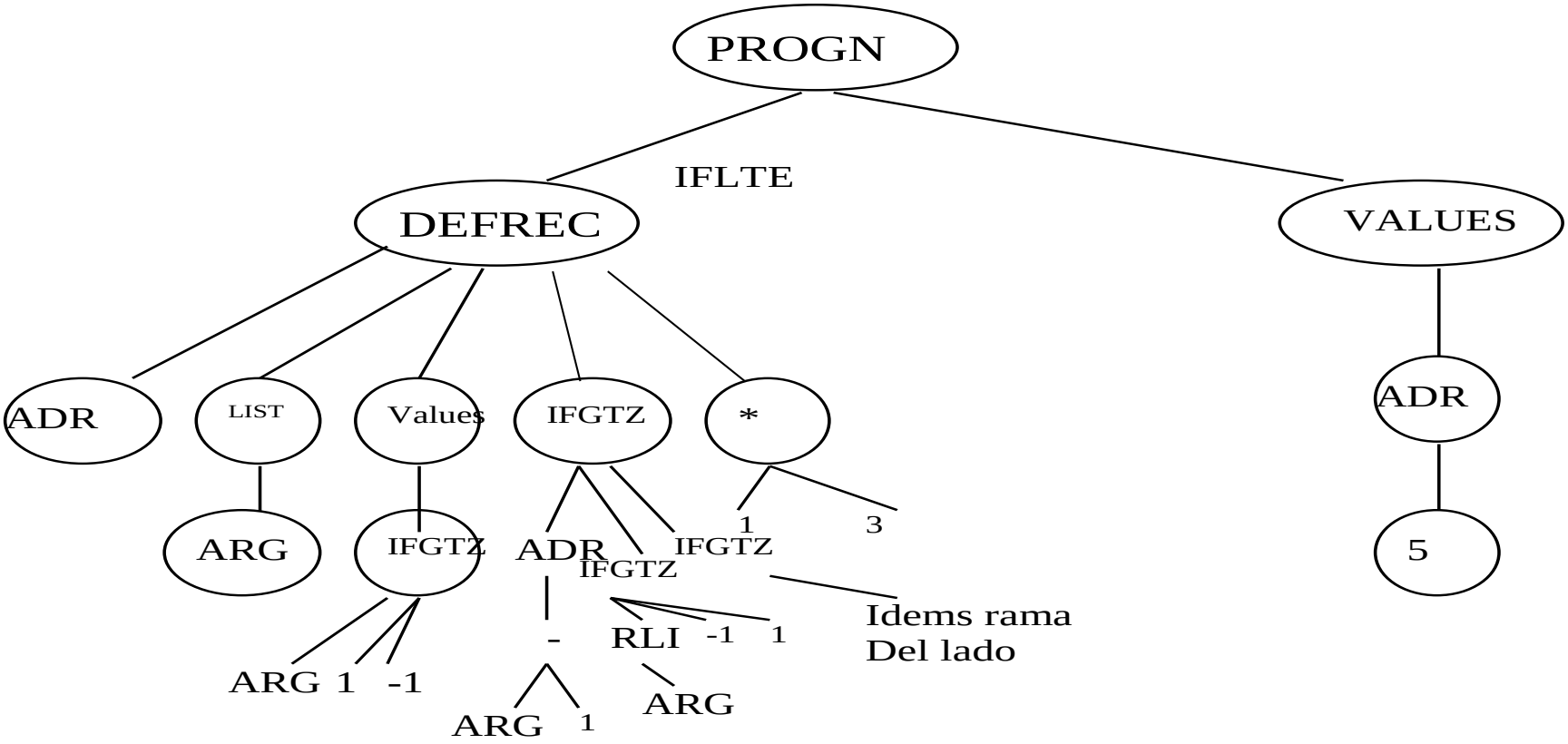
ADR

Partes

- Rama Condicional (RCB)
- Rama cuerpo positivo (RBB)
- Rama cuerpo negativo (RGB)

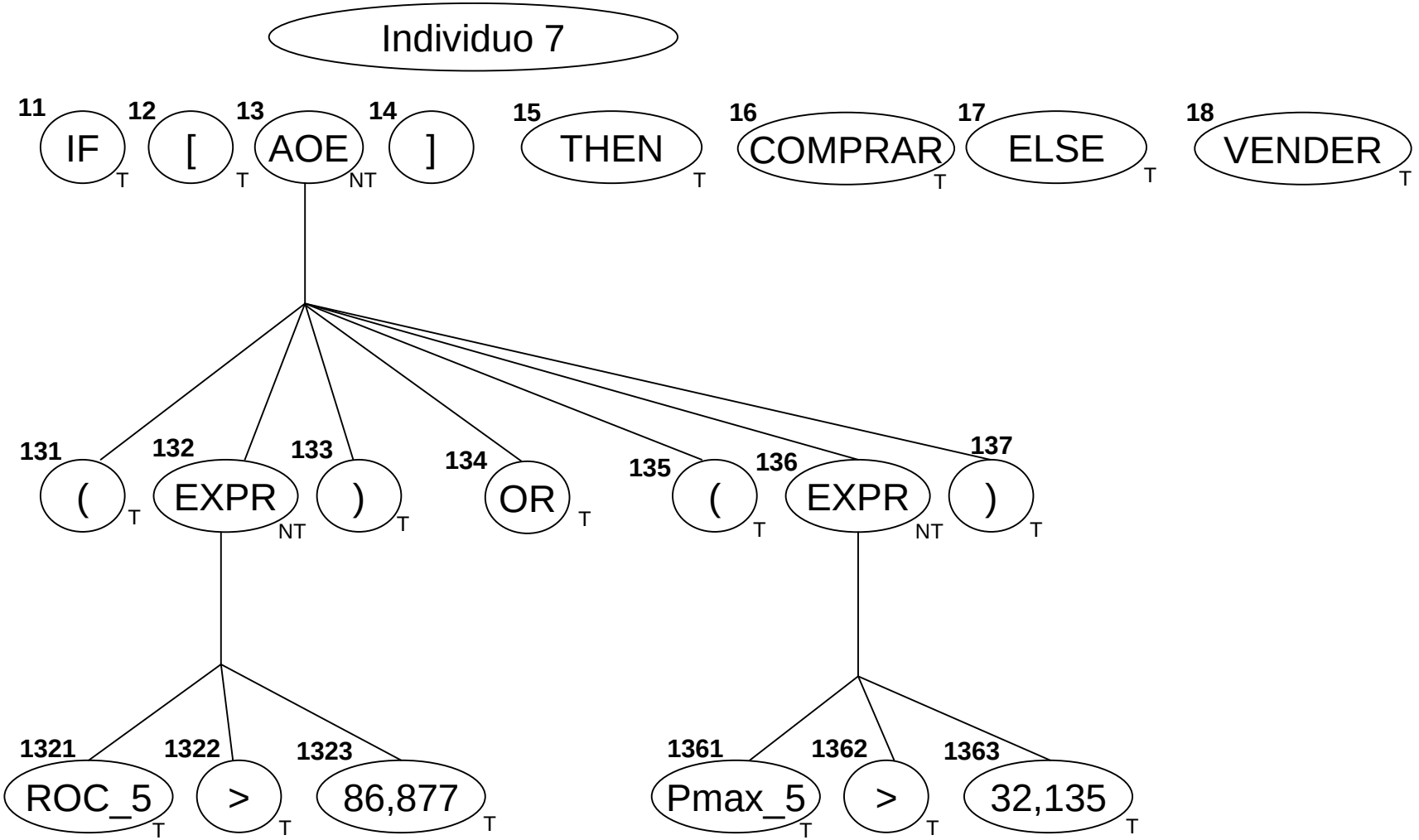
```
float ADR (float ARG)
{    if (RCB (ARG) > 0)
            result=RBB(ARG);            /* llamada a ADR*/
     else
            result=RGB(ARG);
return(result); }
```

PROGRAMACIÓN GENÉTICA



ADR

IF [(ROC_5 > 86,8) OR (Pmax_5 > 32,1)] THEN COMPRAR ELSE VENDER



PROGRAMACIÓN GENÉTICA

ADR

Operadores

Creación

=> Tomar un pedazo del código y crear ADR

Eliminación

=> regeneración aleatoria del pedazo

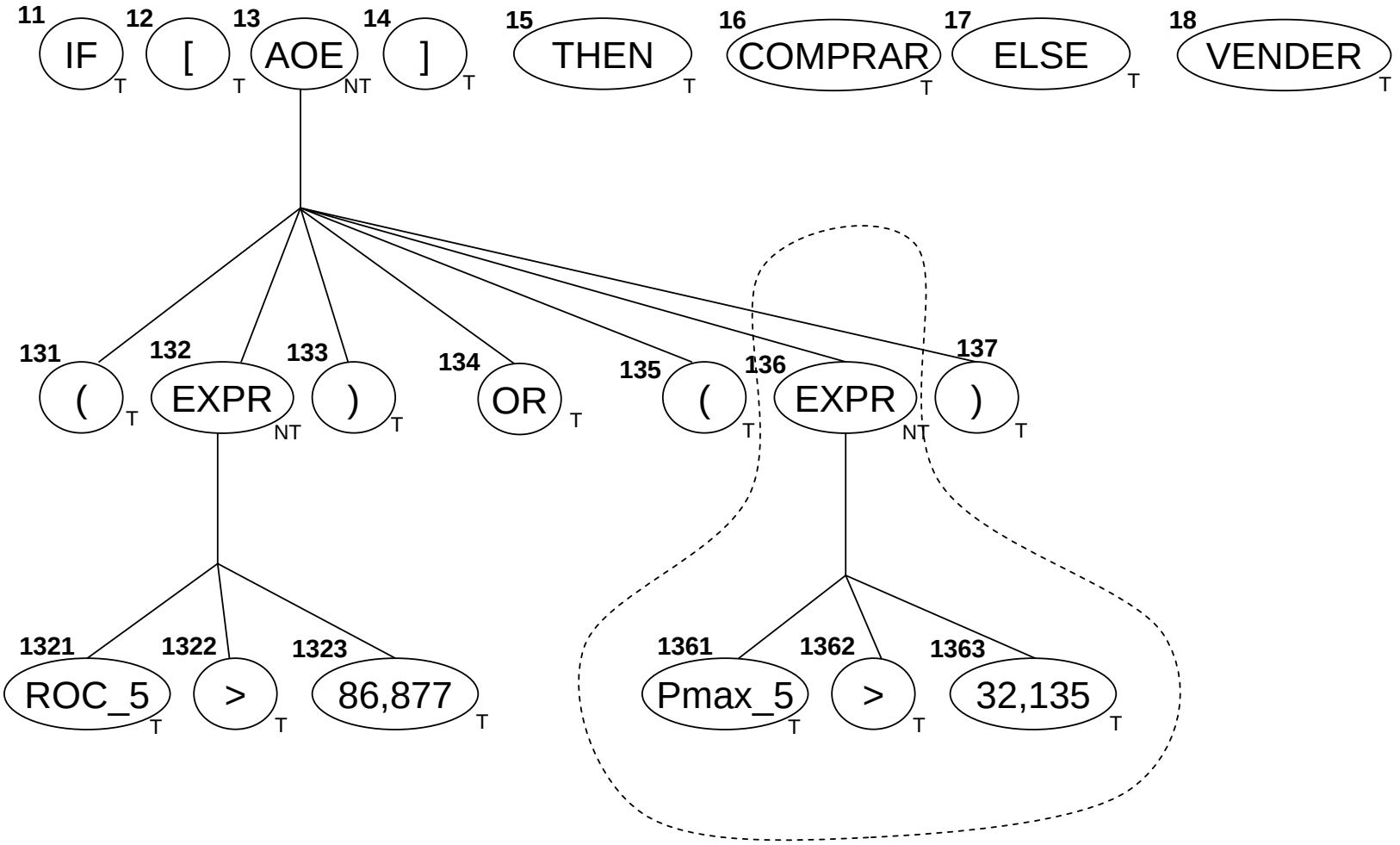
=> modificación conjunto terminal y funciones

Duplicación

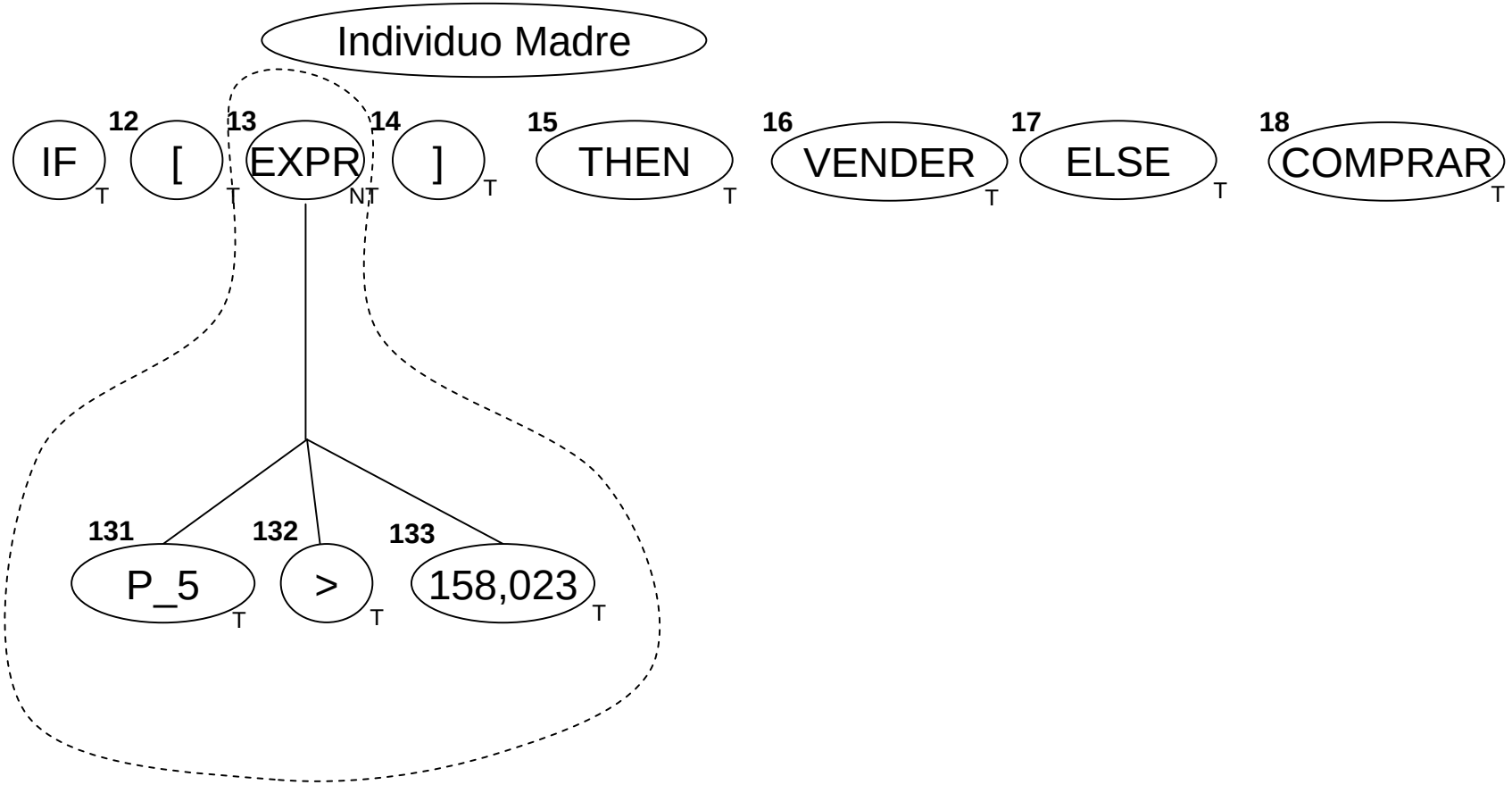
=> Escoger un ADL existente y modificar nombre, etc.

=> Llamados al ADL viejo son aleatoriamente modificados

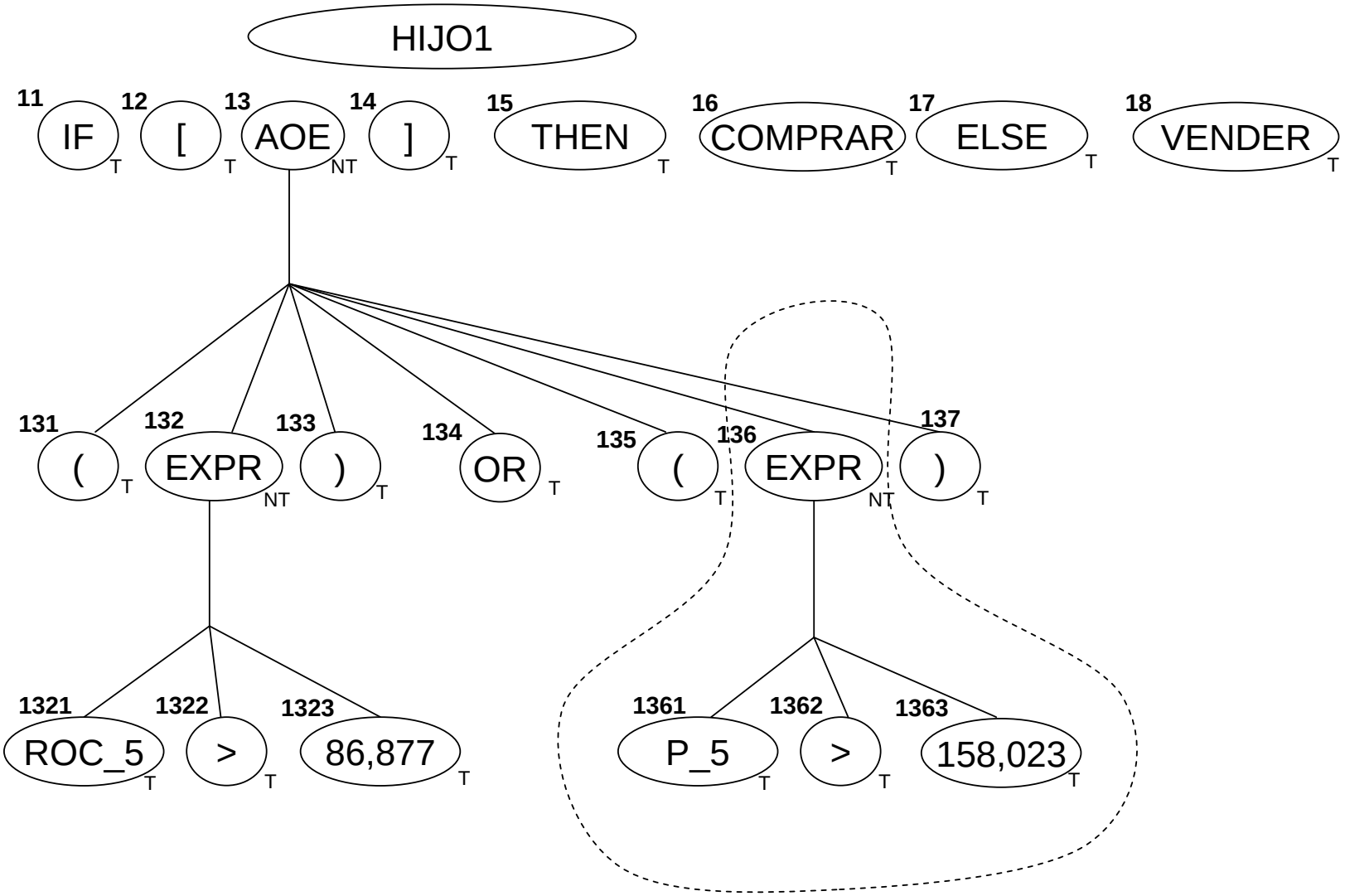
Individuo Padre: IF [(ROC_5 > 86,877) OR (Pmax_5 > 32,135)]
THEN COMPRAR ELSE VENDER



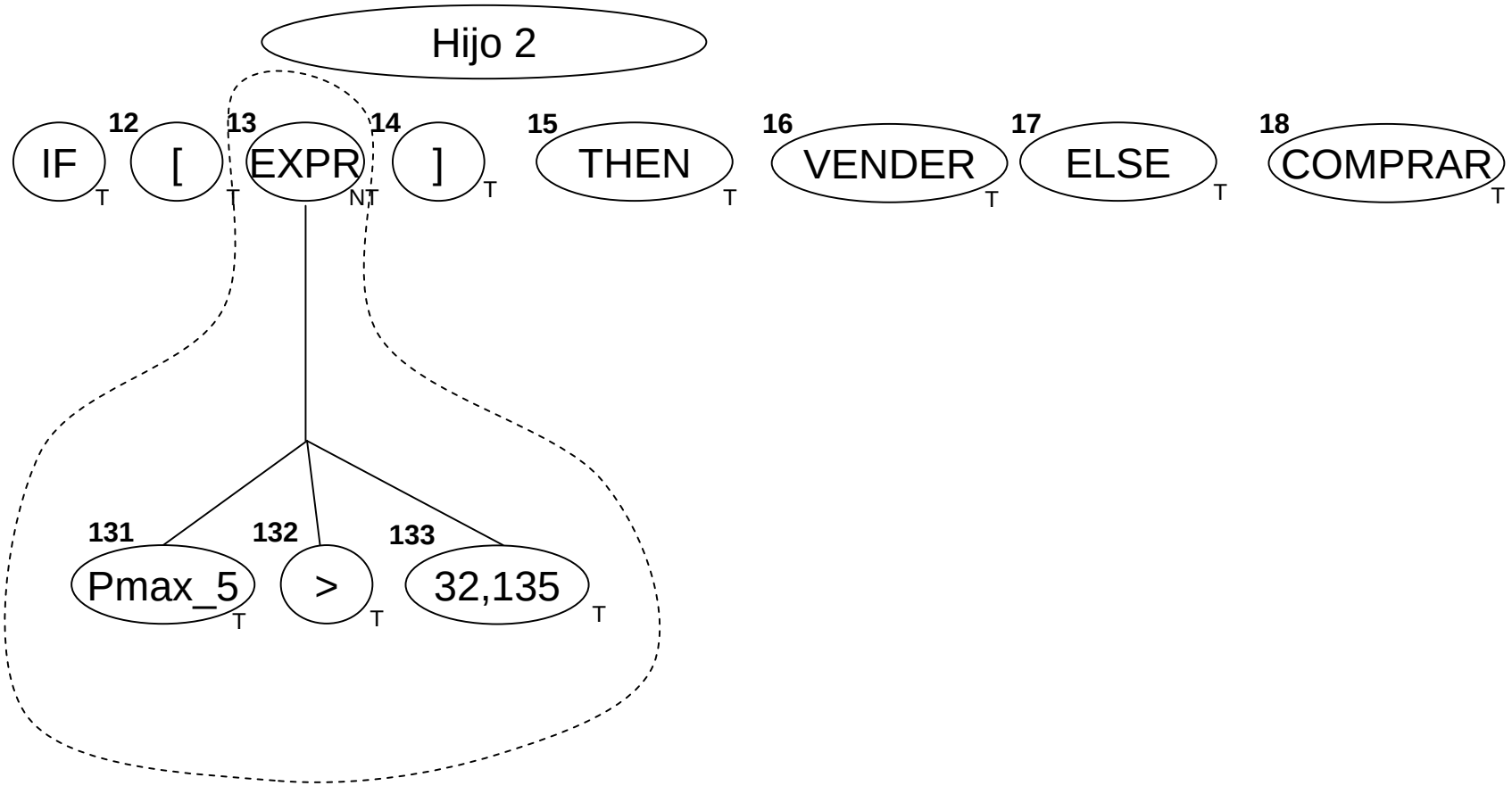
Individuo Madre: IF [P_5 > 158,023] THEN VENDER ELSE COMPRAR



Individuo Hijo1: IF [(ROC_5 > 86,877) OR (P_5 > 158,023)] THEN COMPRAR ELSE VENDER



Individuo Hijo2: IF [Pmax_5 > 32,135] THEN VENDER ELSE COMPRAR



PROGRAMACIÓN GENÉTICA

AREAS DE APLICACIÓN

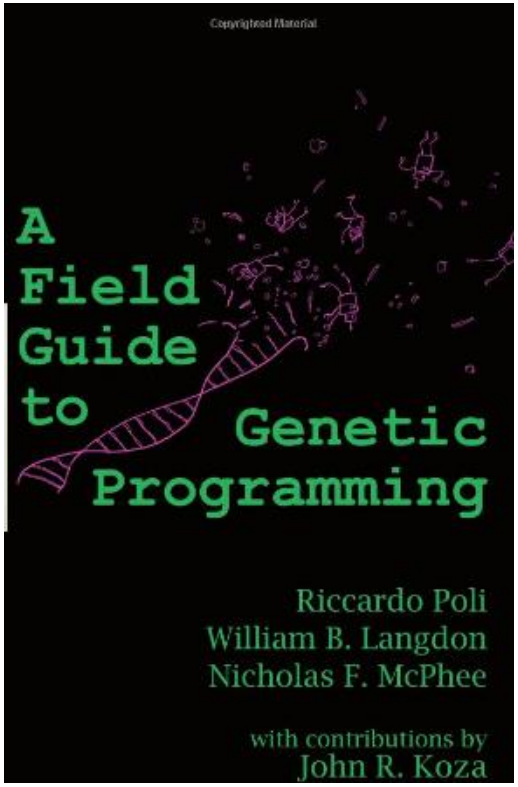
- CONTROL: IDENTIFICACION, ETC.
- MINERIA DE DATOS
- SISTEMAS MULTIAGENTES
- HARDWARE EVOLUTIVO

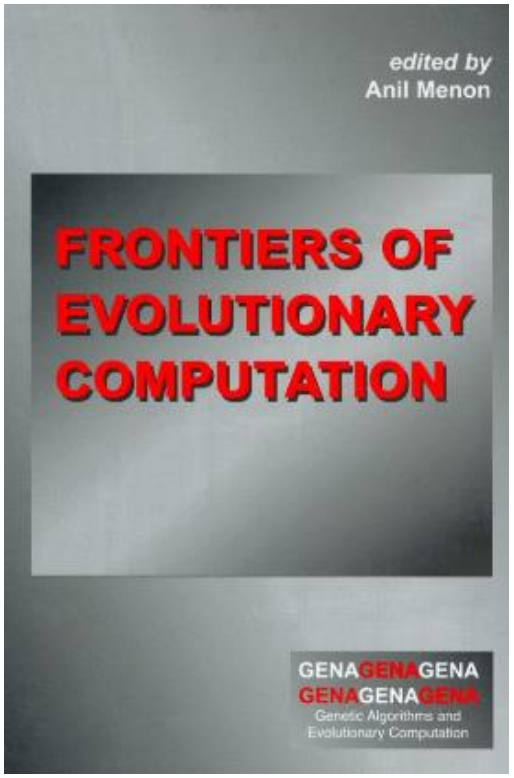
www.aic.nrl.navy.mil/galist

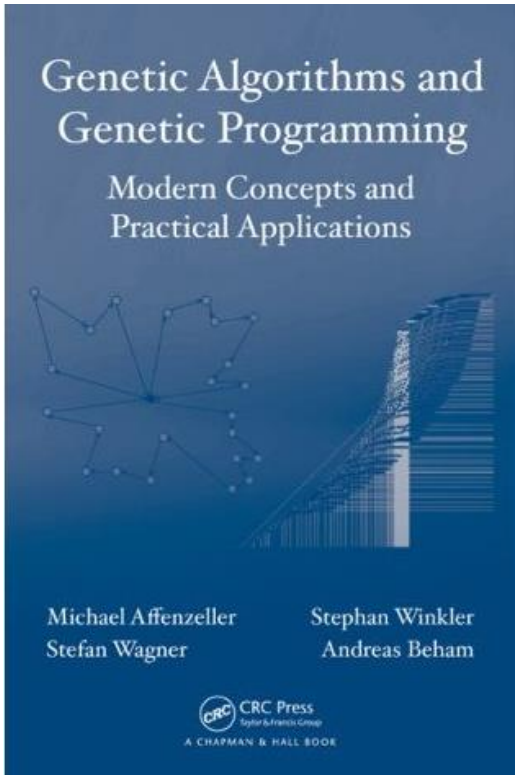
Kernel GPC++

Es una biblioteca de clases escritas en el lenguaje de programación orientado a objetos C++, que permite utilizar la **técnica de Programación Genética** con un esfuerzo mínimo, dado que el *kernel* proporciona el conjunto de estructuras, clases y algoritmos necesarios para utilizar la técnica.

Algunas Referencias







ESCUELA
POLITÉCNICA
NACIONAL

PROGRAMACIÓN EVOLUTIVA

PROGRAMACIÓN EVOLUTIVA

Evoluciona algoritmo que opera secuencia de símbolos tomados de un alfabeto finito

Símbolo de salida maximiza rendimiento si algoritmo predice próximo símbolo

Énfasis en el comportamiento

Por ejemplo: un conjunto de *maquinas de estado finito* se expone a un ambiente.

PROGRAMACIÓN EVOLUTIVA

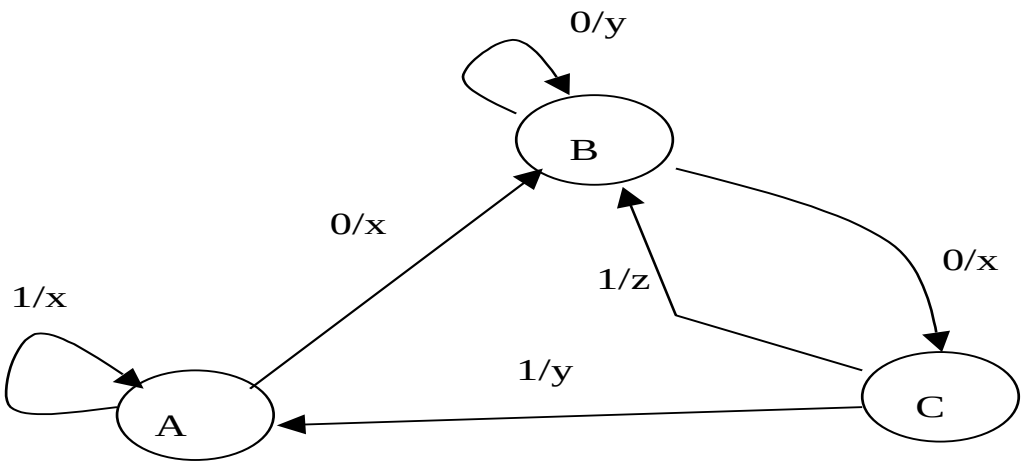
Para cada Máquina de Estados Finitos, su predicción es medida con respecto al ambiente, como una función error.

Después que se hace la ultima predicción, un error promedio para cada máquina es calculado para indicar la aptitud de ella.

Las máquinas que presenten el mejor resultado (mínimo error) se retienen para ser padres en la próxima generación.

PROGRAMACIÓN EVOLUTIVA

Edo. Presente	C	B	C	A	A	B
Entrada	0	1	1	1	0	1
Prox.	B	C	A	A	B	C
Salida	x	z	y	x	x	z



PROGRAMACIÓN EVOLUTIVA

MACROALGORITMO

1. DEFINIR SECUENCIA DE SIMBOLOS Y FA
2. INICIAR POBLACION MEF
3. INTRODUCIR SIMBOLO ENTRADA Y VER SU SALIDA
4. EVALUAR SI PREDIJO BIEN
5. CREAR NUEVOS HIJOS USANDO MUTACION: **CAMBIAR SIMBOLOS, ESTADO DE TRANSICION, O EDO. INICIAL, ANADIR O ELIMINAR SIMBOLO**
6. SELECCIONAR MEJORES PADRES
7. REGRESAR A 3

Esquema de generación de descendientes

Por cada maquina padre se obtiene un descendiente aplicando sobre esta una mutación aleatoria:

- Cambiar un símbolo de salida.
- Cambiar un estado de transición.
- Agregar un estado.
- Eliminar un estado.
- Cambiar el estado inicial.

PROGRAMACIÓN EVOLUTIVA

EJEMPLO: PREDICCION DE TIEMPO

{SOL, LLUVIA, SOL, LLUVIA, TORMENTA, ...}

REPRESENTADO POR {0, 1, 0, 1, 2, ...}

MATRIZ DE PAGO

	0	1	2
0	1	-10	-100
1	-10	5	-40
2	-25	-5	10

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

ESCUELA POLITÉCNICA NACIONAL

Estrategias Evolutivas

ESTRATEGIAS EVOLUTIVAS

Las EEs pueden definirse como **algoritmos evolutivos enfocados hacia la optimización paramétrica** que utilizan una representación a través de **vectores reales**, una **selección determinista**, operadores específicos de **mutación y cruce** (Cruce Numérico) y **restricciones implícitas** en la representación u operadores.

Las estrategias evolutivas pueden dividirse en:

Simples.

Múltiples.

ESTRATEGIAS EVOLUTIVAS

Las EE simples son consideradas como procedimientos estocásticos de optimización paramétrica con paso adaptativo (similares al temple simulado).

Se hace evolucionar un solo individuo usando únicamente un operador genético, la mutación.

ESTRATEGIAS EVOLUTIVAS

Se denotan como (1+1)-EE:

- El individuo (v) se representa a través de un par de vectores reales v = (x;s).
- El vector *x* representa una solución, y *s* es un vector de desviaciones típicas usado para la mutación.

$$x[t+1] = \begin{cases} x[t] + N(0, s[k]) & \text{ssi } x[t+1] > x[t] \text{ y } x[t+1] \in \psi \\ x[t] & \text{de lo contrario} \end{cases}$$

  - N(0,s): Representa un vector de números aleatorios gausianos independientes con medias 0 y un vector de desviaciones típicas *s*, donde cada uno de los elementos del vector *s* es la desviación típica a usar por cada uno de los elementos del vector *x.*
  - ψ: Representa el espacio de soluciones.

ESTRATEGIAS EVOLUTIVAS

Las EEs múltiples surgen como respuesta a las debilidades de las EEs simples.

En las EEs múltiples existen múltiples individuos (población), y se producen en cada generación varios nuevos individuos, usando tanto mutación como cruce.

ESTRATEGIAS EVOLUTIVAS

- Se denotan como (λ^+, μ)-EE, donde λ es el tamaño de la población y μ es el tamaño de la descendencia.

- Los símbolos (+ ,) representan dos posibles mecanismos de reemplazo:
  - Por inclusión (+): padres e hijos compiten
  - Por inserción (,): solo sobreviven hijos y compiten entre ellos

ESTRATEGIAS EVOLUTIVAS

Operadores de cruce y mutación

Cruce intermedio:

Padres: $\left\{ \begin{array}{l} \langle X1, \ldots, Xm \rangle, \langle S1, \ldots, Sm \rangle \\ \langle Y1, \ldots, Ym \rangle, \langle S'1, \ldots, S'm \rangle \end{array} \right.$

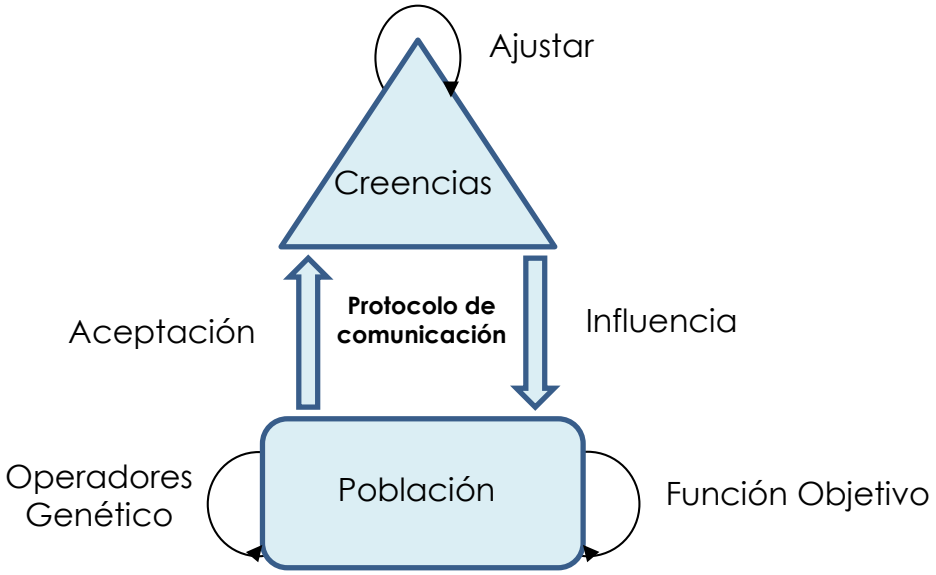
resultado: $\langle 1/2\,(X1+Y1), \ldots, 1/2(Xm+Ym);$

$\mathrm{sqrt}(S1^2+S'1^2), \ldots, \mathrm{sqrt}(S1^2+S'1^2)\rangle$

Algoritmos culturales (AC):

Basado en la **evolución cultural**, vista como un proceso de herencia a dos niveles:

- **el nivel micro-evolutivo**, que consiste en el ***material genético hered***ado por los padres a sus descendientes, y
- **el nivel macro-evolutivo**, que es el conocimiento adquirido por los individuos a través de las generaciones, y que una vez codificado y almacenado, sirve para ***guiar el comportamiento*** de los individuos que pertenecen a una población



Begin
$t = 0$;
Iniciar P^t
Iniciar B^t
Repetir
Evaluar P^t
Ajuste (B^t, Acepte (P^t))
Variación (P^t, influencia (B^t))
$t = t + 1$;
Seleccionar P^t de P^{t-1}
Hasta (haber alcanzado la condición de finalizar)

End

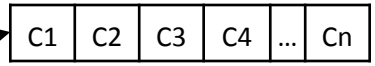
Pseudo-código de un AC

Algoritmos culturales (AC):

Caracterizar los componentes del AC

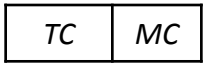
- Población
- Espacio de creencias
- Protocolo de comunicación

Individuo

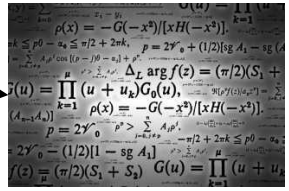


Conversaciones y sub-conversaciones

Conocimiento



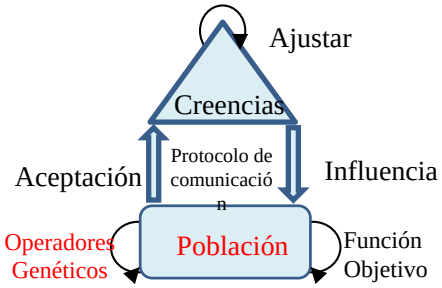
Tipos de conversación y mecanismos de coordinación

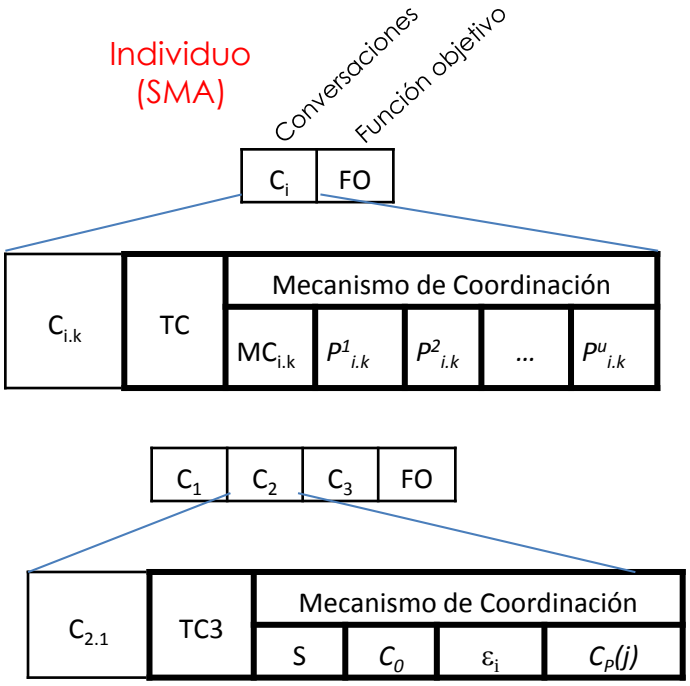


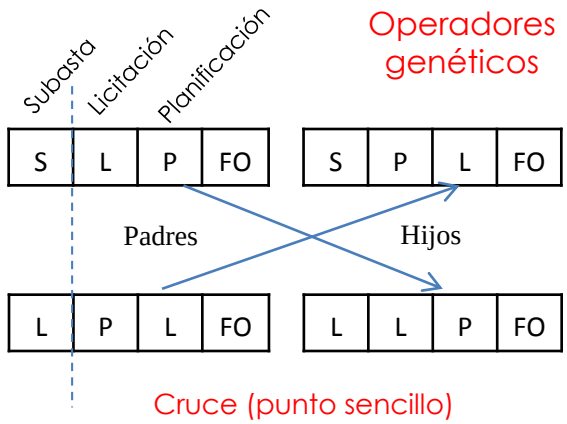
Ecuaciones

Algoritmos culturales (AC):

Población







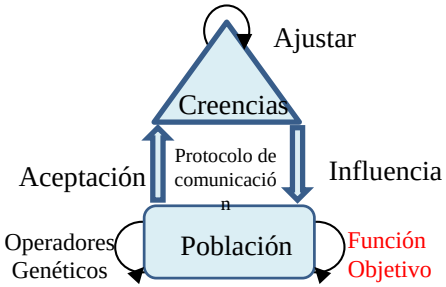
Algoritmos culturales (AC):

Función objetivo

Permite evaluar el performance o desempeño de los individuos, esta basada en el costo de procesamiento (CP) y en el costo comunicacional (CC) de cada mecanismo de coordinación usado por el individuo (la idea es minimizar la función objetivo)

$$FO = \Sigma^{n}_{i=1} \Sigma^{m}_{i\ k=1} (a * CP_{i.k} + b * CC_{i.k})$$

- Donde a y b son constantes definidas por el usuario y permiten normalizar las unidades de la función
- Conversaciones i = {1 … n}
- Sub-conversaciones k = {1 … m_i}



- CP y CC

Costo de Procesamiento

$$CP_{i.k} = PI_k + PE_k + \Sigma^{j}_{l=1} \Sigma^{nj}_{q=1} A_{l,q}$$

- *PI* fijación del precio inicial; especificación de condiciones en las que se requiere un servicio, generación de planes
- *PE* proceso de selección del agente ganador, asignación de planes
- $A_{l,q}$ tiempo para preparar propuestas, tiempo para generar planes parciales

Costo de Comunicación

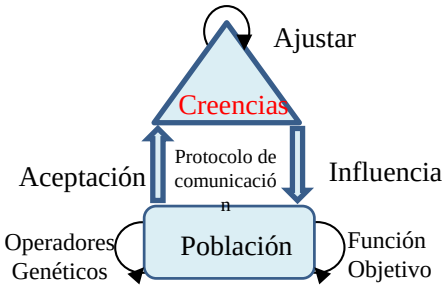
$$CC_{i.k} = \Sigma^{j}_{l=1} (\Sigma^{N-1}_{r=1} CEP_{l,r} + \Sigma^{nj}_{s=1} CEO_{l,s}) + \Sigma^{N-1}_{r=1} CS_r$$

- *CEP* costo de envío de propuesta
- *CEO* costo de envío de ofertas, envío de planes parciales
- *CS* costo de informar al ganador, de enviar el plan global

Algoritmos culturales (AC):

Espacio de creencias

- **Conocimiento normativo:** colección de rangos de valores deseables para los individuos de la población
- **Conocimiento del dominio específico**: información acerca del dominio del problema
- **Conocimiento situacional**: ejemplos específicos de eventos importantes,
- **Conocimiento temporal**: historial del espacio de búsqueda (registra lo recorrido del espacio de búsqueda), por ejemplo, patrones temporales del proceso de búsqueda
- **Conocimiento espacial**: información acerca de la topografía del entorno o espacio de búsqueda.



Conocimiento Circunstancial

Ejemplos específicos de eventos importantes, e. g., experiencias exitosas

TC	*MC*	$IO_{(TC, MC, t-1)}$	$TO_{(TC)}$

- *IO* es el índice de ocurrencias
- *TO* es el total de ocurrencias

Conocimiento Normativo

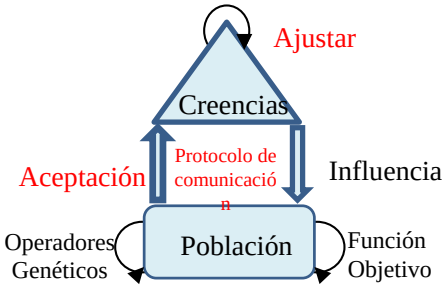
Son rangos de valores idóneos de cada una de las variables que integran al mecanismo de coordinación usado por los individuos

MC	P^1		P^2		...	P^u	
	LI	*LS*	*LI*	*LS*		*LI*	*LS*

- *LI, LS* son los límites inferiores y superiores de cada variable P^u

Algoritmos culturales (AC):

Protocolo de comunicación: función aceptación



Conocimiento Circunstancial

$$IO_{(TC,MC,t)} = IO_{(TC,MC,t-1)} + \left(\frac{NO_{(TC,MC,t)}}{TO_{(TC)}} \right)$$

$$TO_{(TC)} = TO_{(TC)} + \sum_{i=1}^{k} NO_{(TC,MC,t)i}$$

TC, tipo de conversación
MC, mecanismo de coordinación
t, estado actual

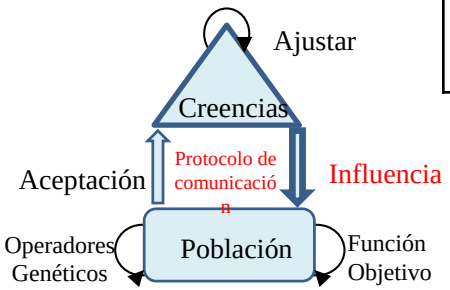
Conocimiento Normativo

$$Lac(P^u) = \left[\frac{\left(lv + \overline{P} \right)}{2} \right]$$

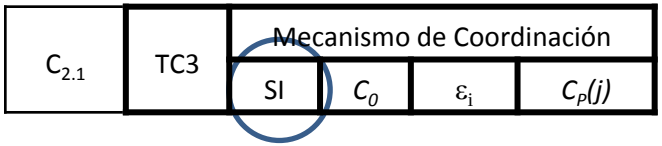
Lac, limite actual
Lv, limite anterior

Algoritmos culturales (AC):

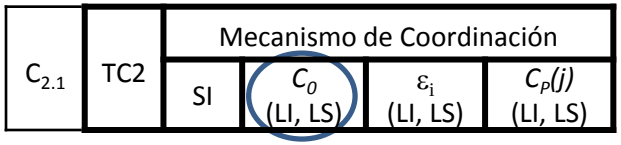
Protocolo de comunicación: función influencia



Conocimiento Circunstancial en el Individuo



Conocimiento Normativo en el Individuo



Mutación dirigida