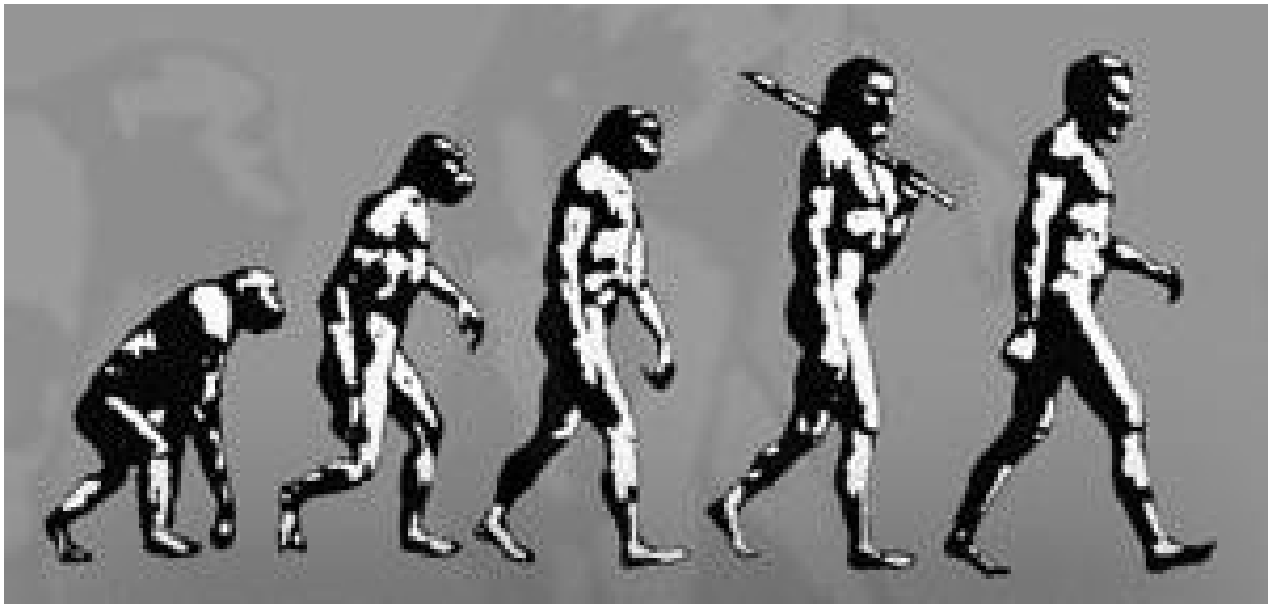
Introducción a la Computación Evolutiva

Jose Aguilar

Evolución

Es el cambio en las trazas heredadas en una poblacion de una generación a la otra



Selección natural

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.

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

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

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 probabilisticas, 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

Procedimiento general de los algoritmos evolutivos

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

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

Operadores Genéticos

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

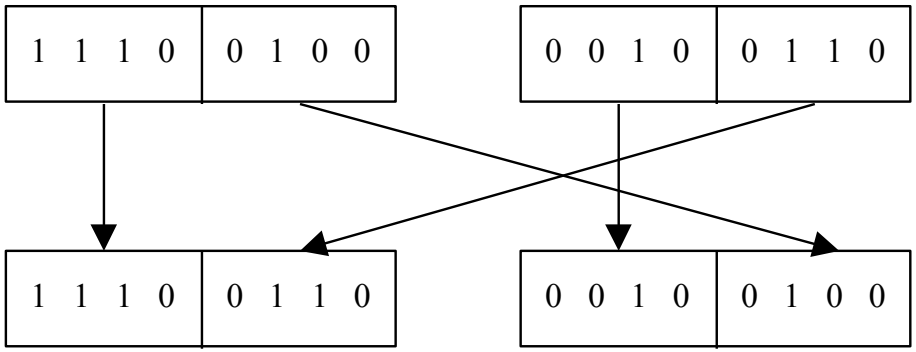
  realizan los cambios de la población durante la ejecución de un algoritmo evolutivo.

- 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 informacion entre diversos individuos para producir dos nuevos individuos.
- El punto de corte es escogido aleatoriamente.

Operador Cruce

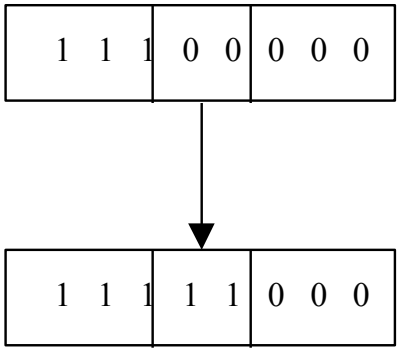


- Tipos:
  - Cruce multipunto
  - Cruce segmentado
  - Cruce uniforme

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 Reemplazo

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

PARAMETROS QUE CONTROLAN EL PROCESO

- Probabilidades de uso de los operadores

- Criterios de parada
  - Por ejemplo, Número de generaciones, similitud de los individuos, etc.

- Tamaño y diversidad de la población

Applicaciones

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

Programación Genética

PROGRAMACIÓN GENÉTICA

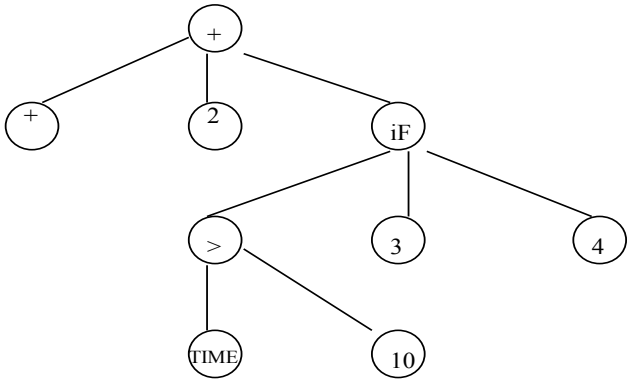
Consiste en utilizar la metodología de CE ,no para encontrar soluciones a problemas, sino para encontrar el mejor procedimiento para resolverlos.

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:



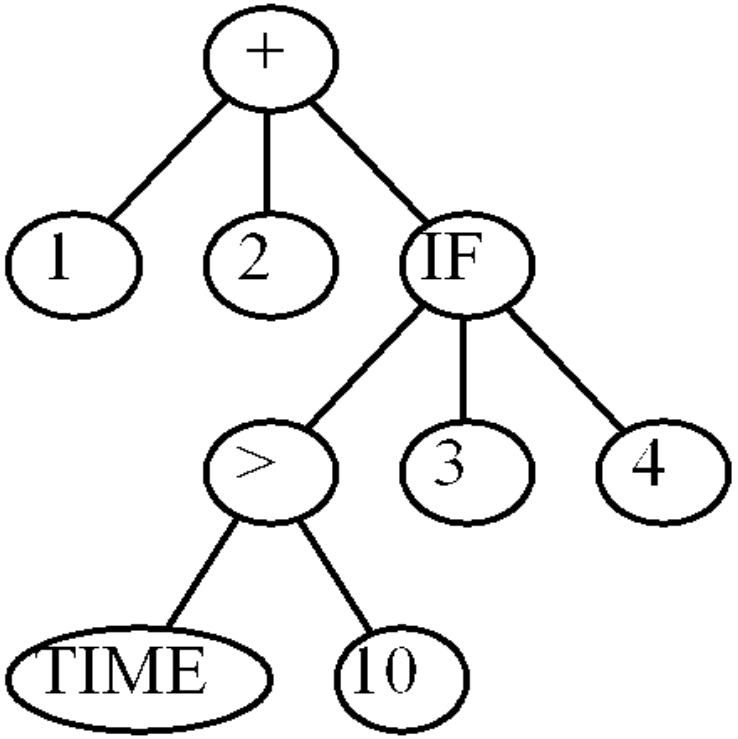
PROGRAMACIÓN GENÉTICA

```
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

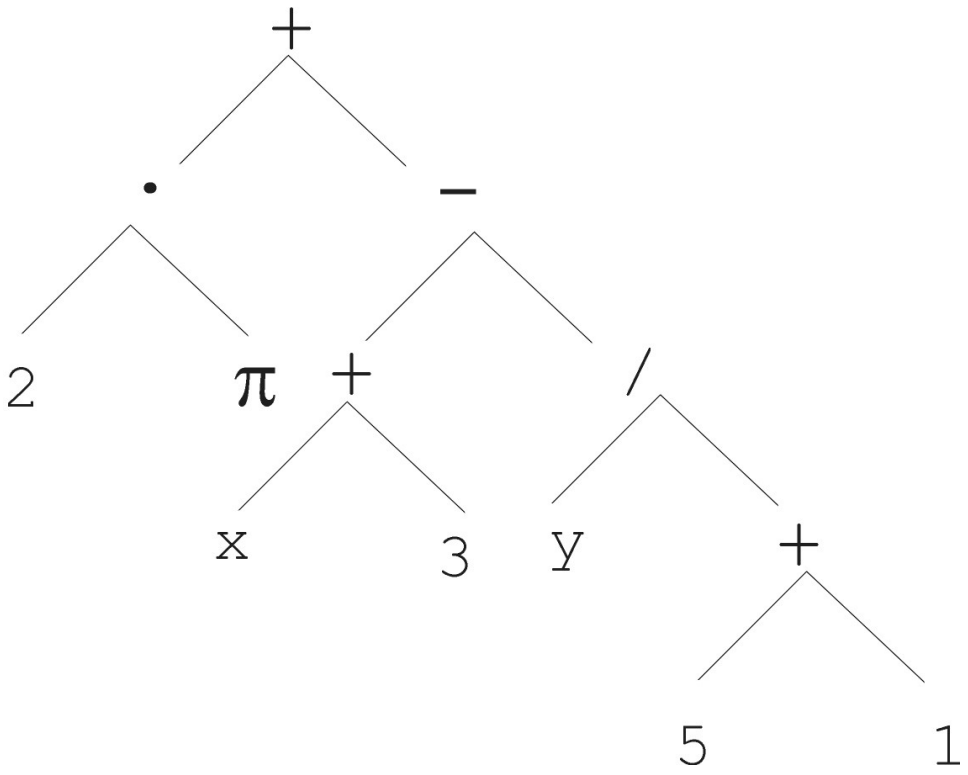
Representación de un programa




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

PROGRAMACIÓN GENÉTICA

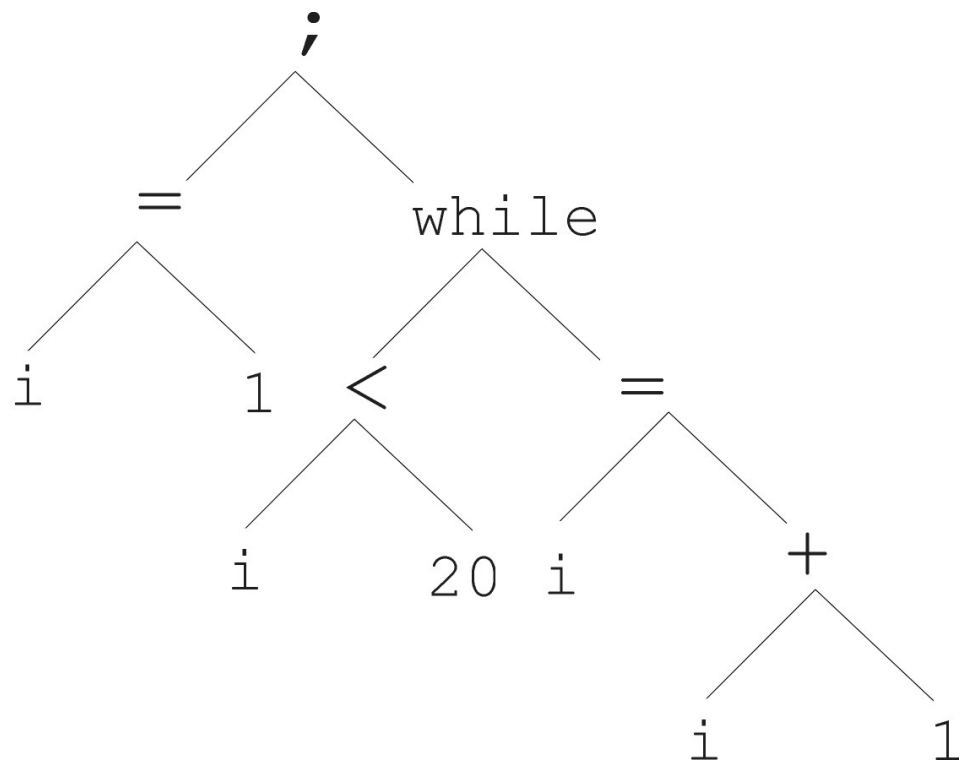
Representaciones de una fórmula



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

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.

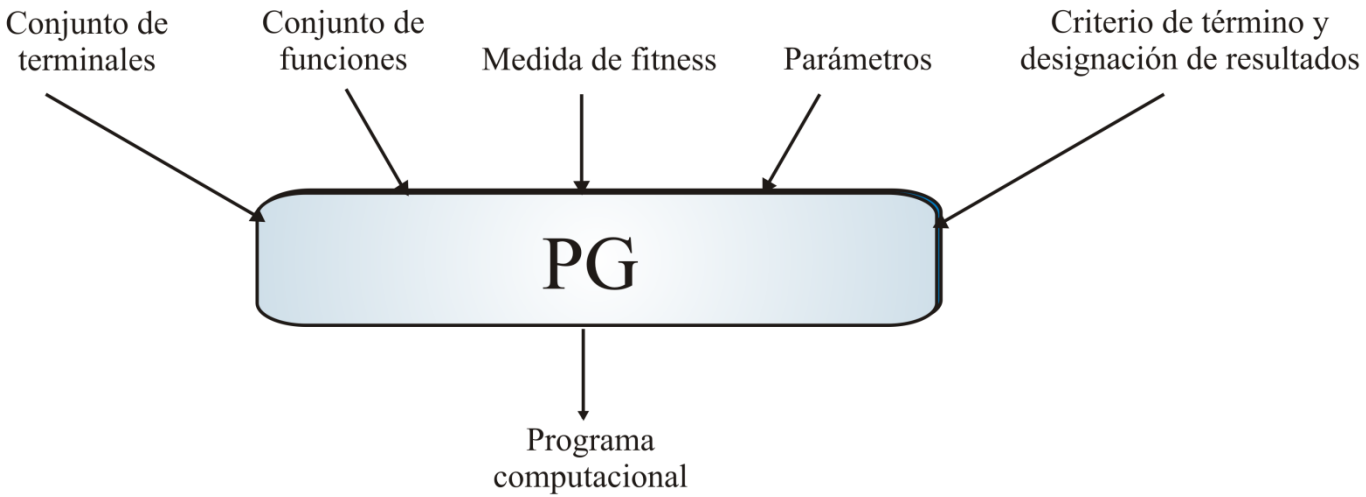
Las funciones pueden ser operaciones aritméticas, funciones matemáticas, funciones lógicas, o funciones especificas del dominio.

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
- Determinar el conjunto de funciones
- Determinar la medición del fitness
- Determinar los parámetros para la ejecución
- Determinar 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

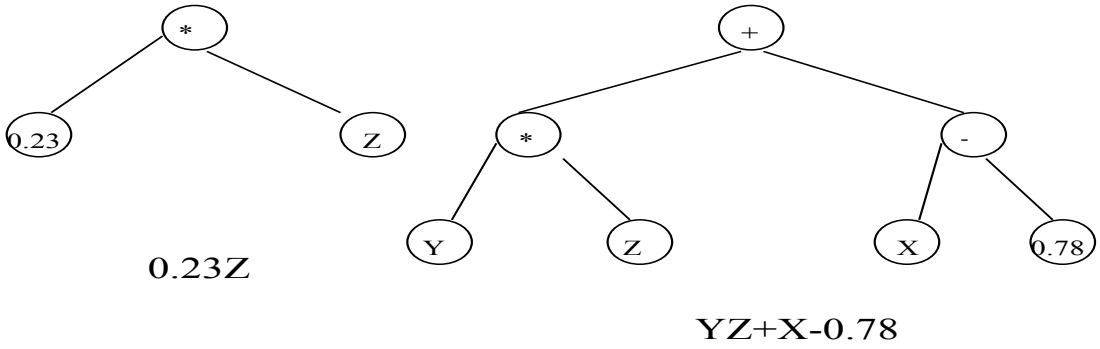
El espacio de búsqueda del problema son todas las posibles combinaciones de funciones y terminales

PROGRAMACIÓN GENÉTICA

EJEMPLO:

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

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



Parámetros de control

- Los parámetros de control más importantes son:
  - **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

La función de Evaluación

- Normalmente la Función de Fitness es un “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

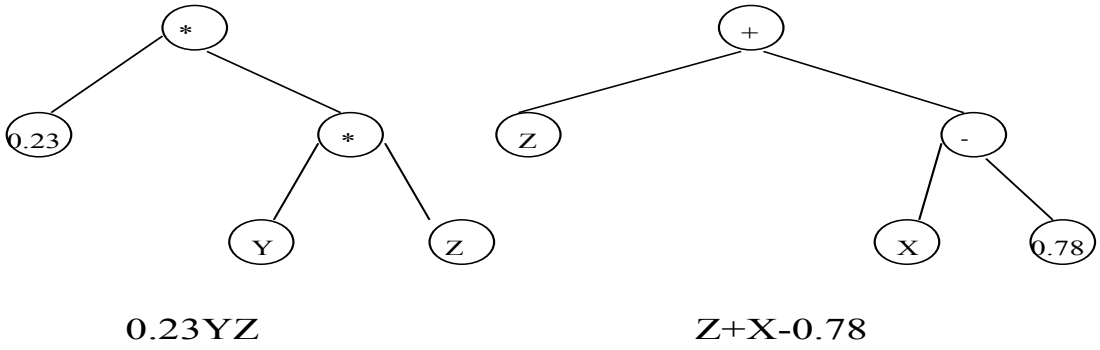
– Función de evaluación de error cuadrático $$f_p = \sum_{i=1}^{n} (p_i - o_i)^2$$

PROGRAMACIÓN GENÉTICA

OPERADOR CRUCE

0.23Z YZ+X-0.78

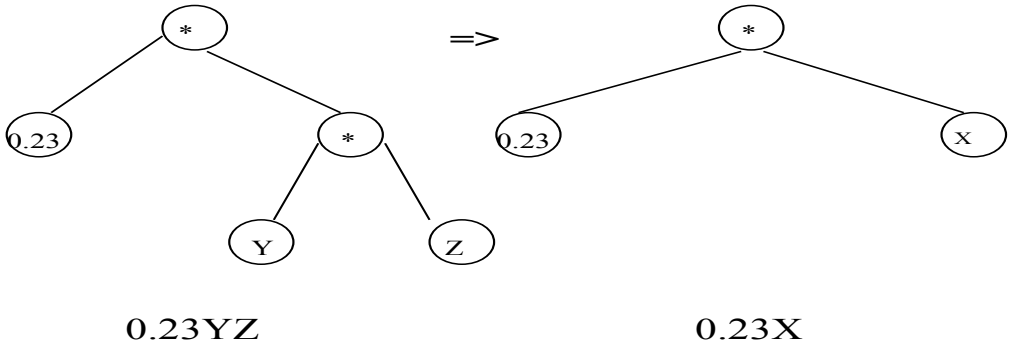
=> 0.23YZ Z+X-0.78



PROGRAMACIÓN GENÉTICA

OPERADOR MUTACION

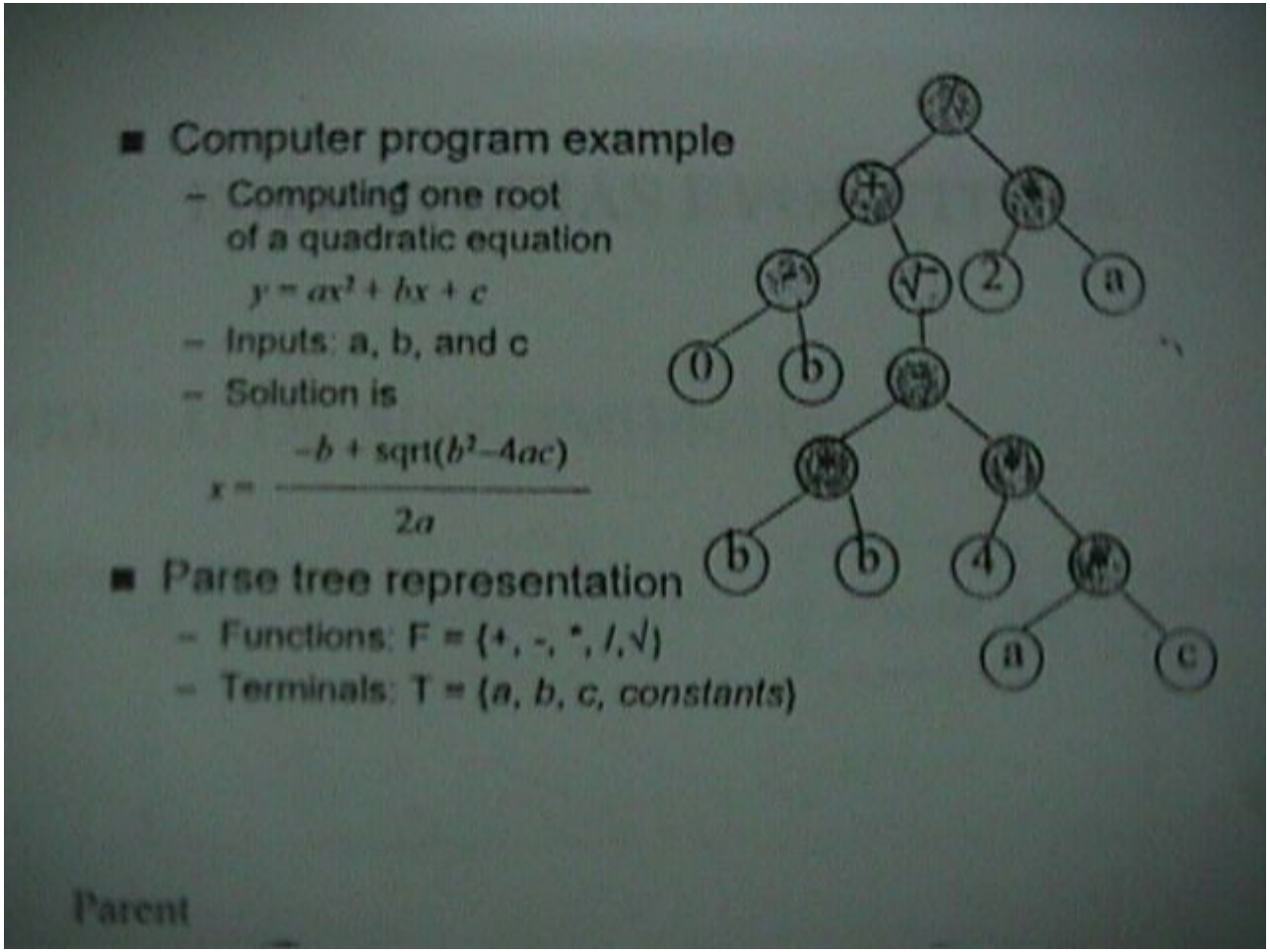
0.23YZ



PROGRAMACIÓN GENÉTICA

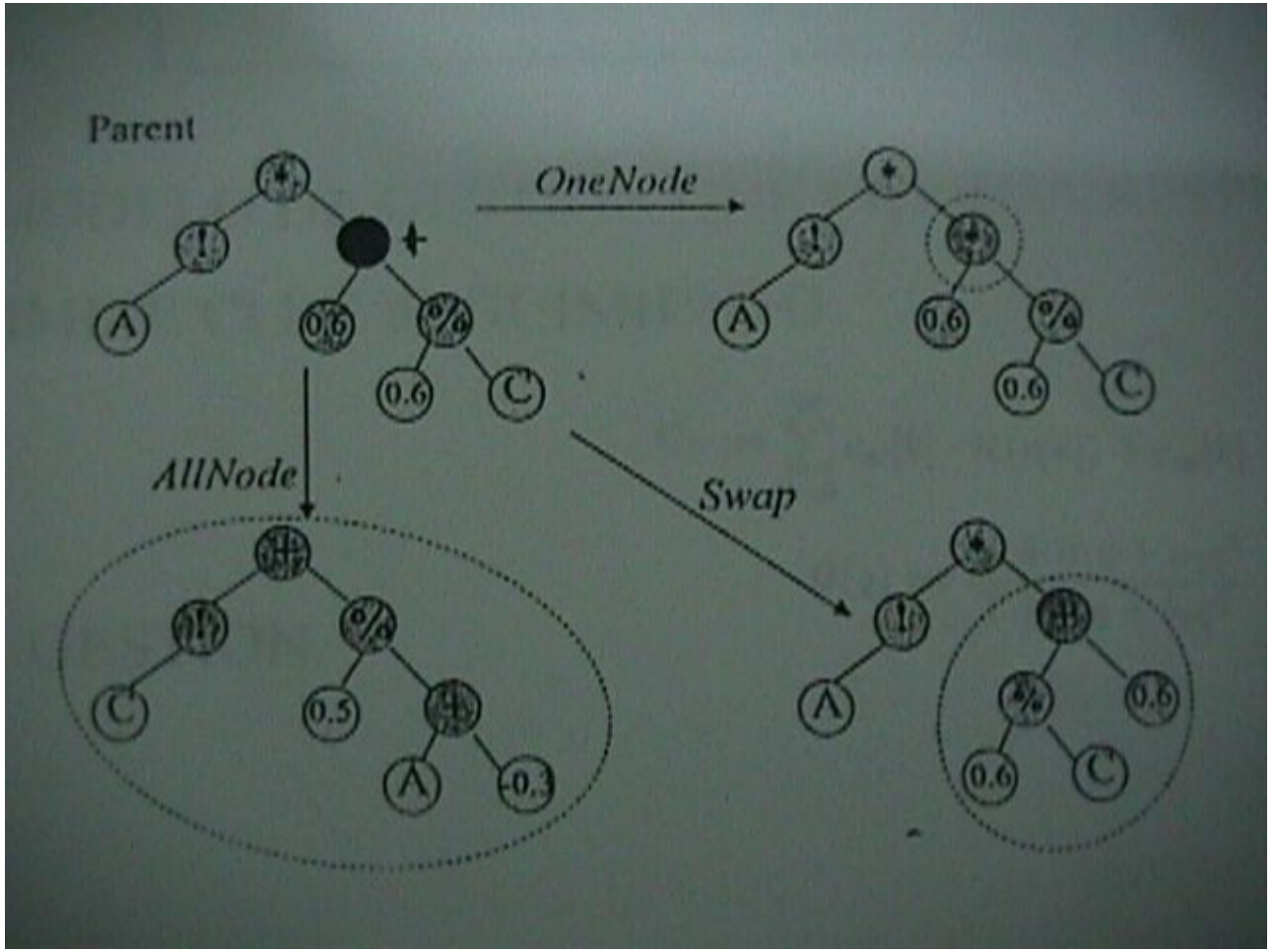
EJEMPLO PARA PRESENTAR OTROS OPERADORES

SUPONGA



PROGRAMACIÓN GENÉTICA

OTROS OPERADORES



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

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

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

Código 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.**

Problema de la mochila

Definición de la función de evaluación

Función de evaluación

$$q_P = \frac{1}{n_f} \sum_{j=1}^{n_f} \frac{u_j - z_j}{u_j}$$

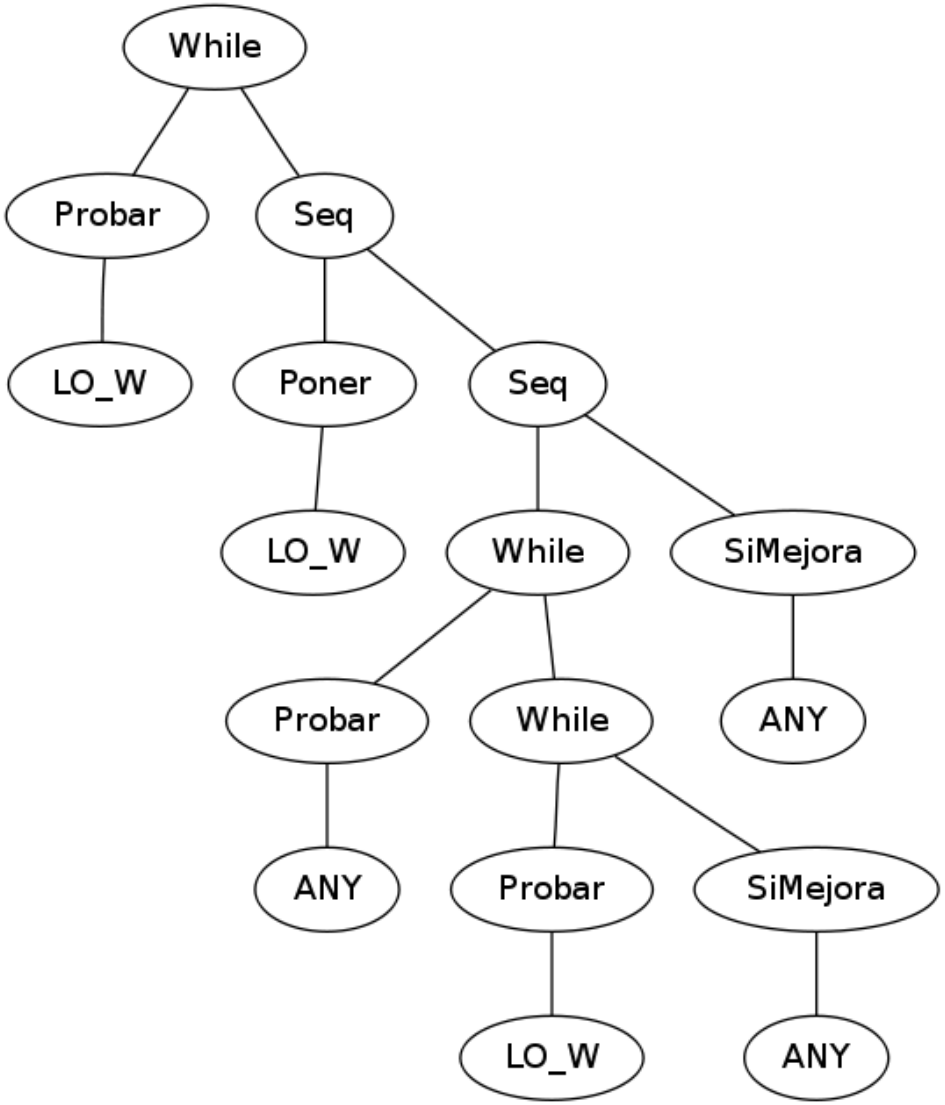
$S_p =$ m / n_p

$$f_p = \alpha q_p + (1 - \alpha) S_p, \ \alpha \in [0,1]$$

Problema de la mochila

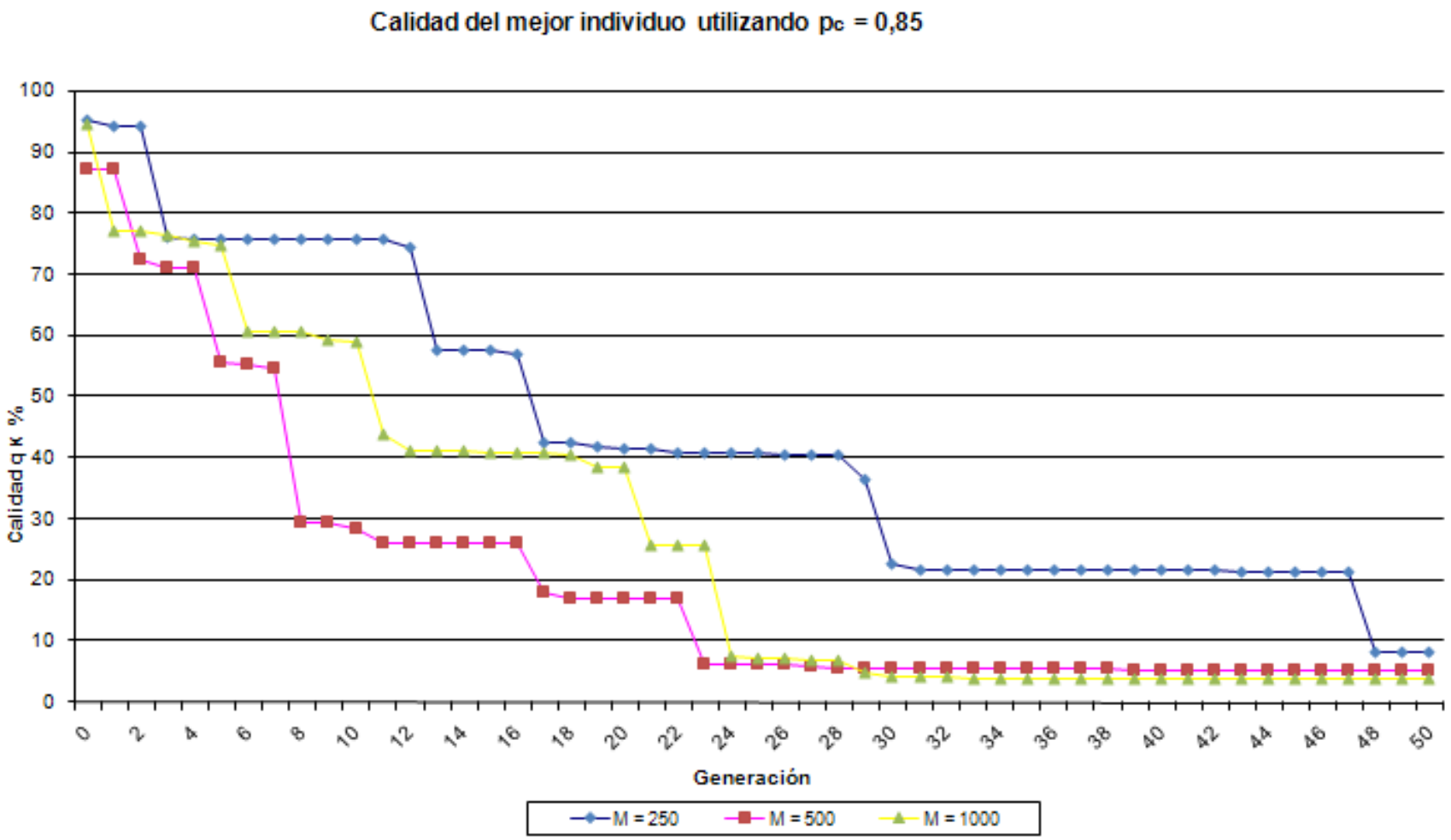
Tamaño de la población (M)	500
Numero máximo de generaciones (G)	50
Probabilidad de cruzamiento (Pc)	85%
Probabilidad de mutación (Pm)	0%

Problema de la mochila



Problema de la mochila

Convergencia



Problema de Coloración de Vértices

Dado un grafo no dirigido $G = (V, E)$, donde V es el conjunto de vértices y E el conjunto de aristas, existe una función $F: V \rightarrow N$ donde cada uno de los vértices adyacentes reciben un color distinto en N; esto es, si $u,v \in V$, entonces $F(u) != F(v)$. El problema consiste en encontrar el menor número de colores necesarios para realizar una coloración en G.

Problema Coloración de Vértices: Formulación matemática

$$\min \sum_{k=1}^{n} y_k$$

$$\sum_{k=1}^{n} x_{ik} = 1 \quad \forall\, i \in V$$

$$x_{ik} + x_{jk} \leq y_k \quad \forall\, (i,j) \in V, k = 1,\ldots,n$$

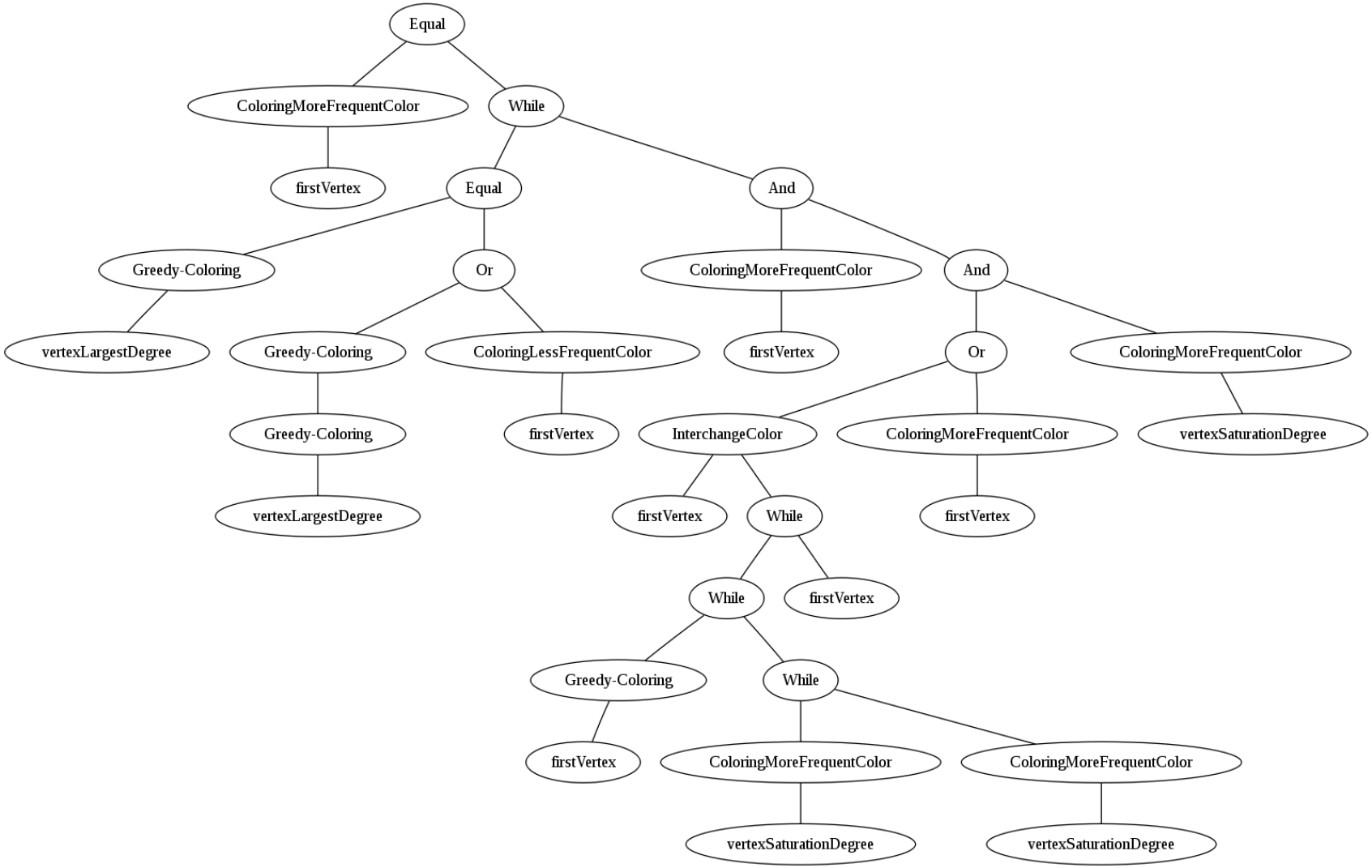
$$x_{ik} \in \{0,1\} \quad \forall\, i \in V, k = 1,\ldots,n$$

$$y_k \in \{0,1\} \quad k = 1,\ldots,n$$

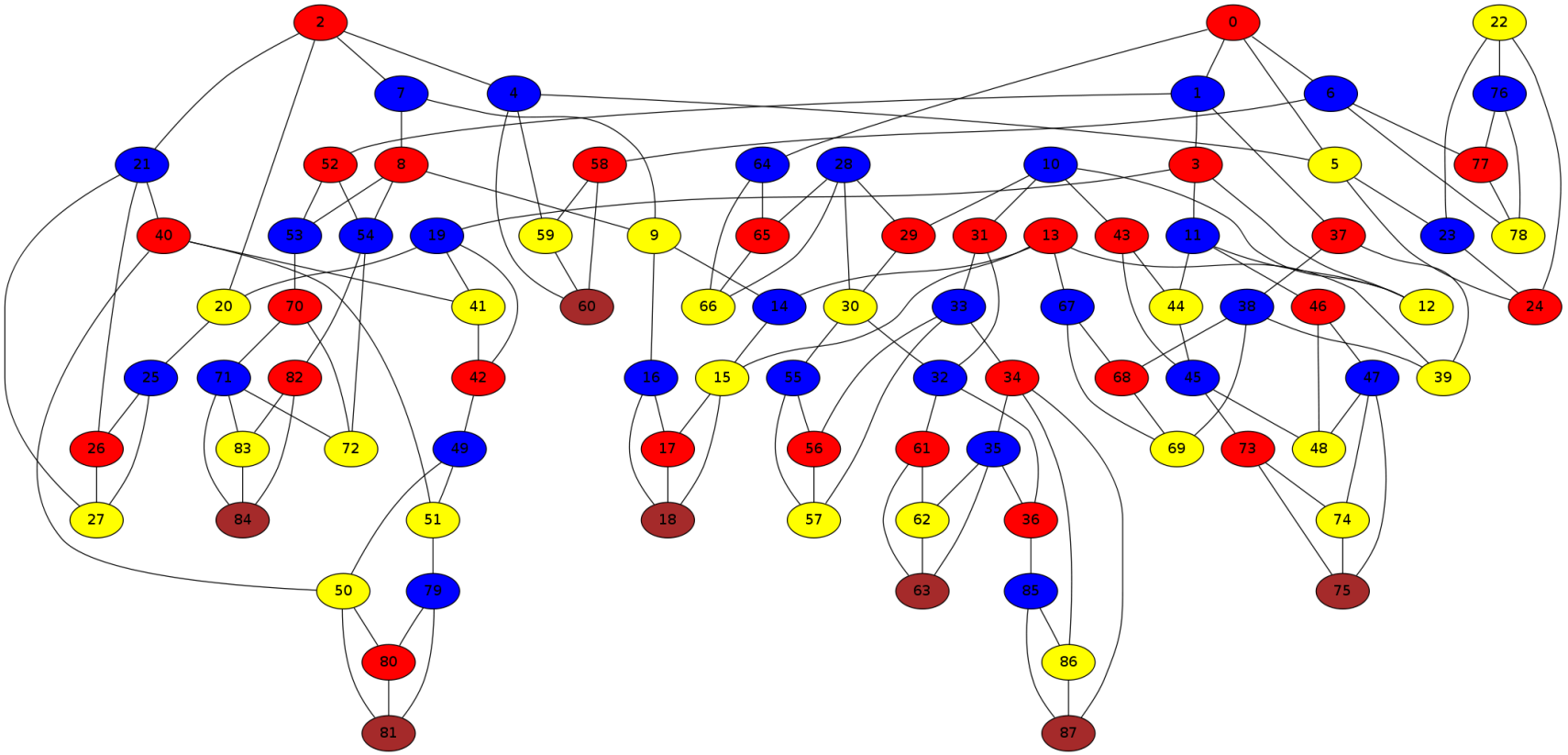
Problema Coloración de Vértices: Parámetros

Tamaño de la población (M)	1000
Numero máximo de generaciones (G)	500
Probabilidad de cruzamiento (Pc)	80%
Probabilidad de mutación Shrink (Pm)	3%

Problema Coloración de Vértices: Estructura algorítmica

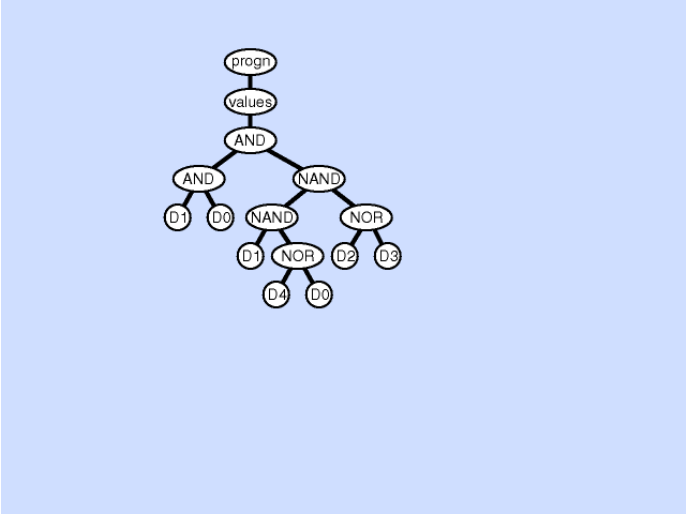


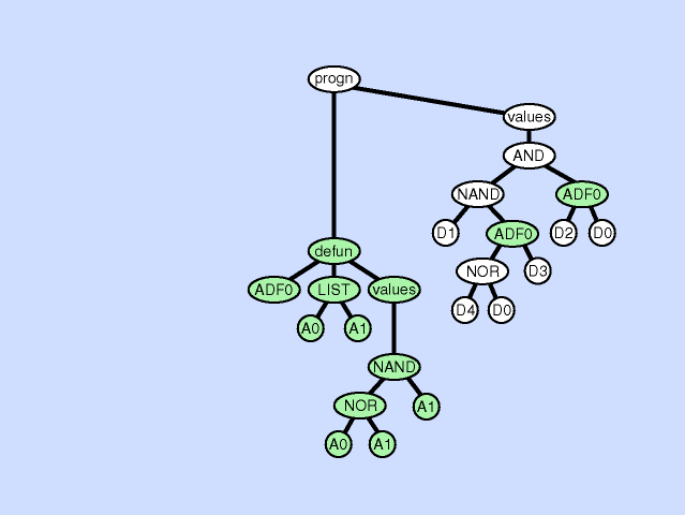
Problema Coloración de Vértices: Algunas coloraciones

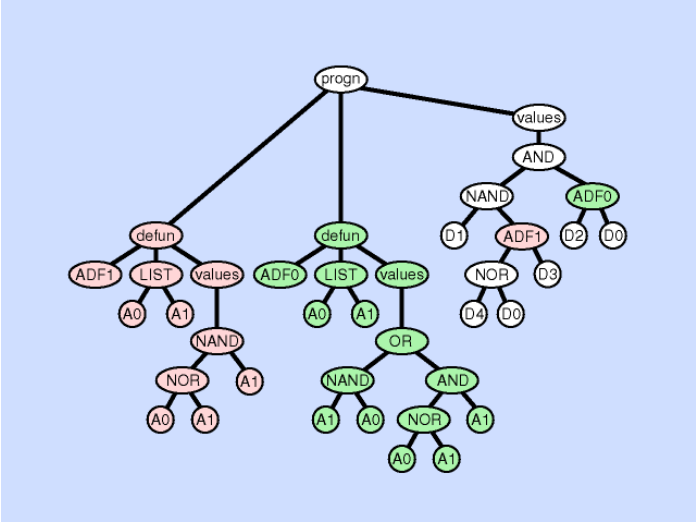


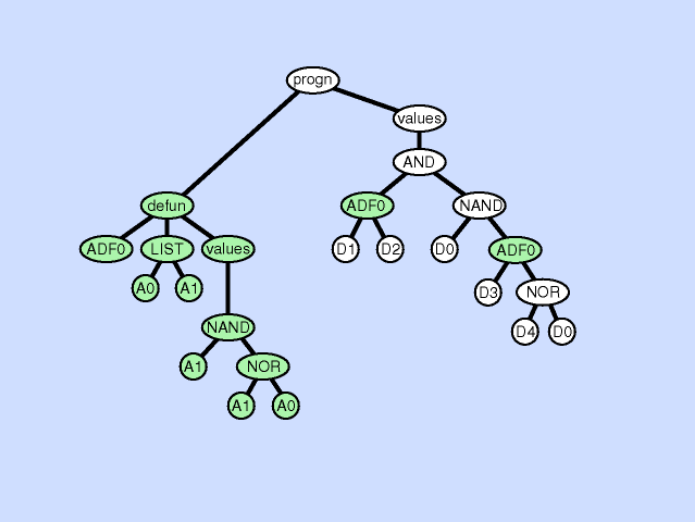
Nombre: 'mug88_1.col', X(G): 4, Cantidad de colores utilizados: 4

Operaciones que alteran la Arquitectura









PROGRAMACIÓN GENÉTICA

ADF (Automatically Defined Functions)

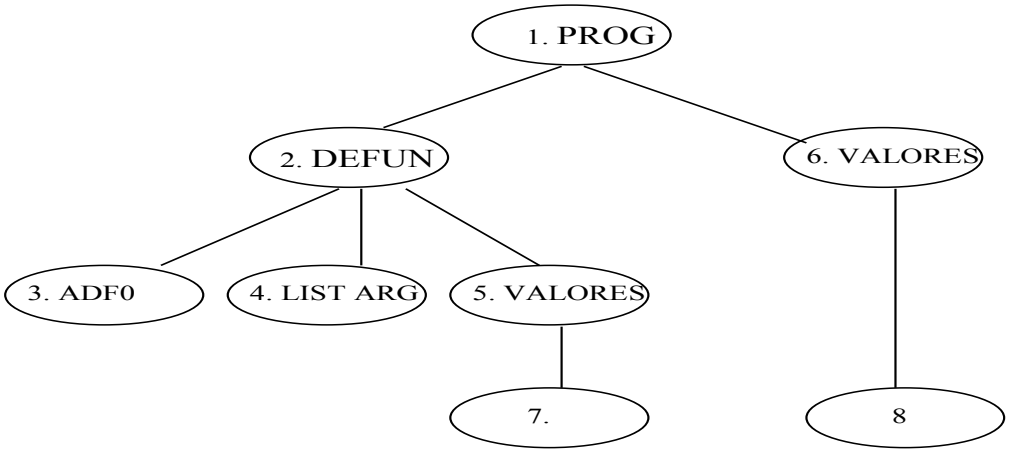
- DESCUBRIMIENTO AUTOMATICO DE UNIDADES FUNCIONALES

- 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

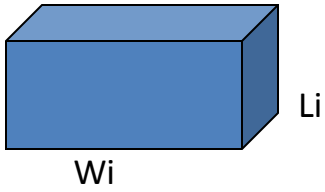


1. RAIZ
2. INICIO RAMA DEF. FUNCION
3. NOMBRE ADF
4. LISTA ARGUMENTOS
5. VALORES A REGRESAR
6. VARIAB. QUE GUARDAN RES.
7. CUERPO ADF
8. CUERPO PROGRAMA

PROGRAMACIÓN GENÉTICA

EJEMPLO ADF

- PROGRAMA CALCULA DIF ENTRE 2 CUBOS



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

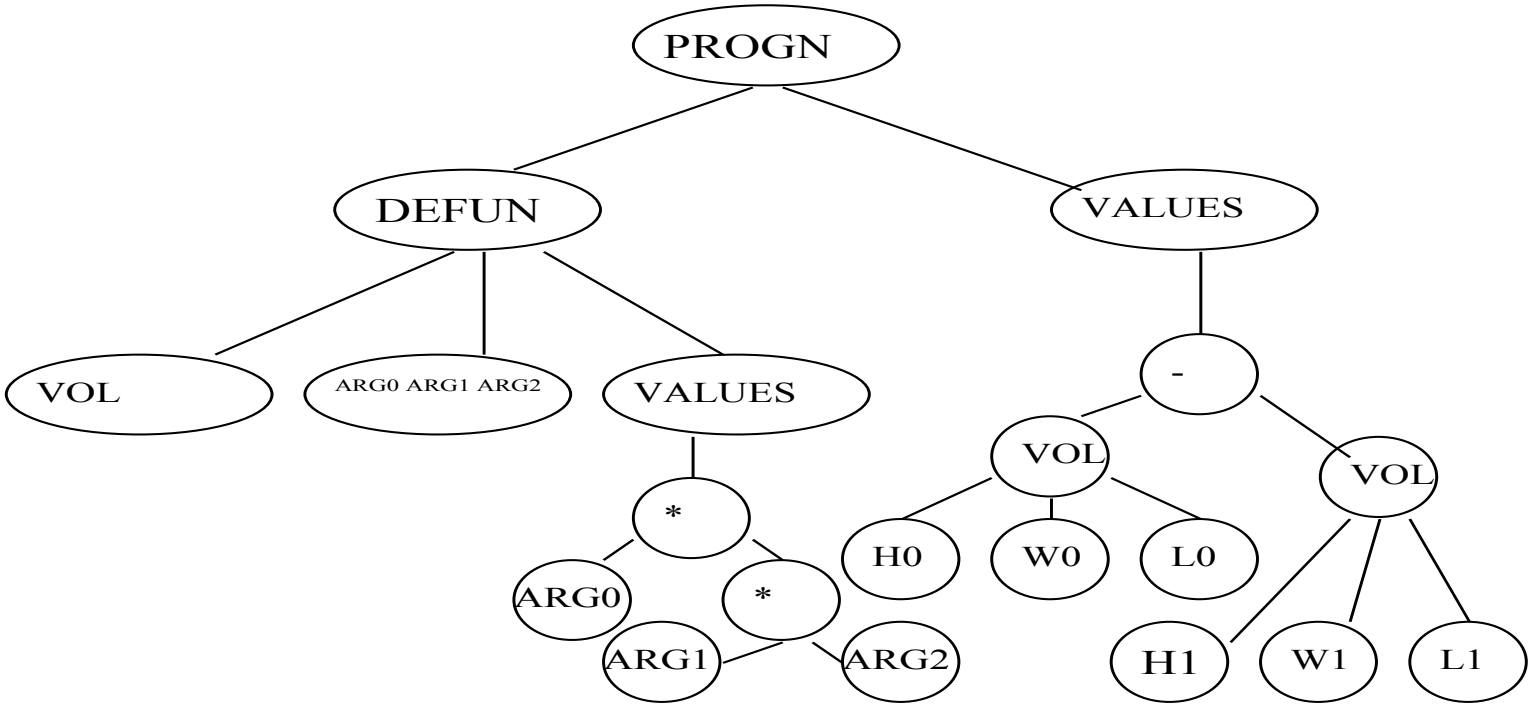
ENFOQUE CON ADF: (PROG (DEF (VOL ARG0 AR1 ARG2)

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

PROGRAMACIÓN GENÉTICA

ADL (AUTOMATICALLY DEFINED LOOP)

REPETICION DE OPERACIONES PREESTABLECIDAS

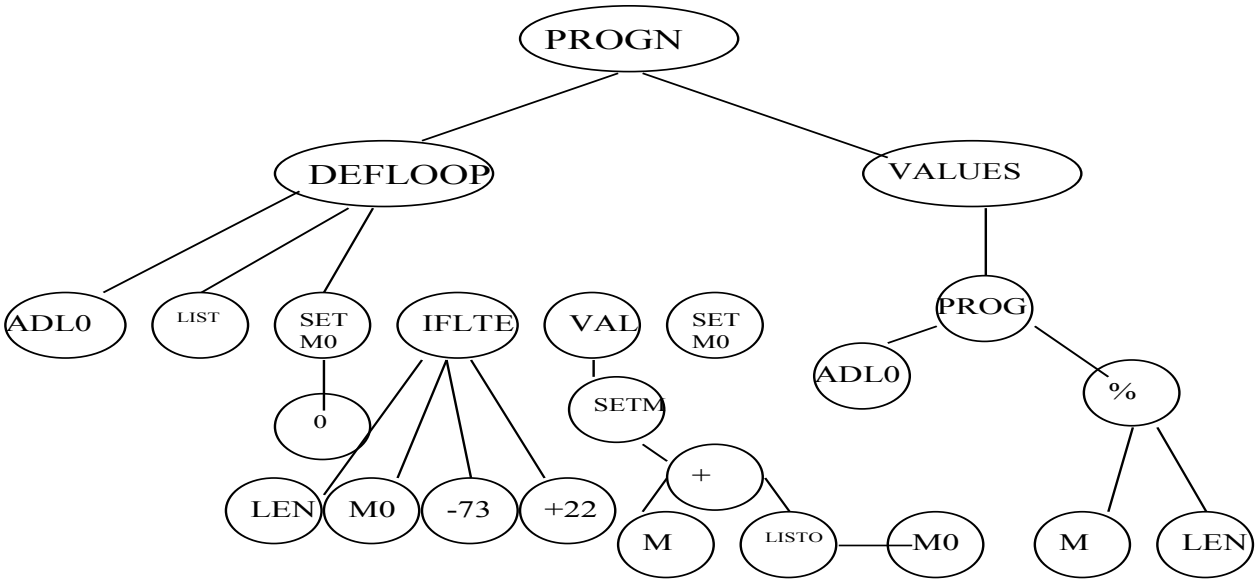
```
for (i=0; i<len; i++)
    m=m+v[i];
```

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

PROGRAMACIÓN GENÉTICA

ADL

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

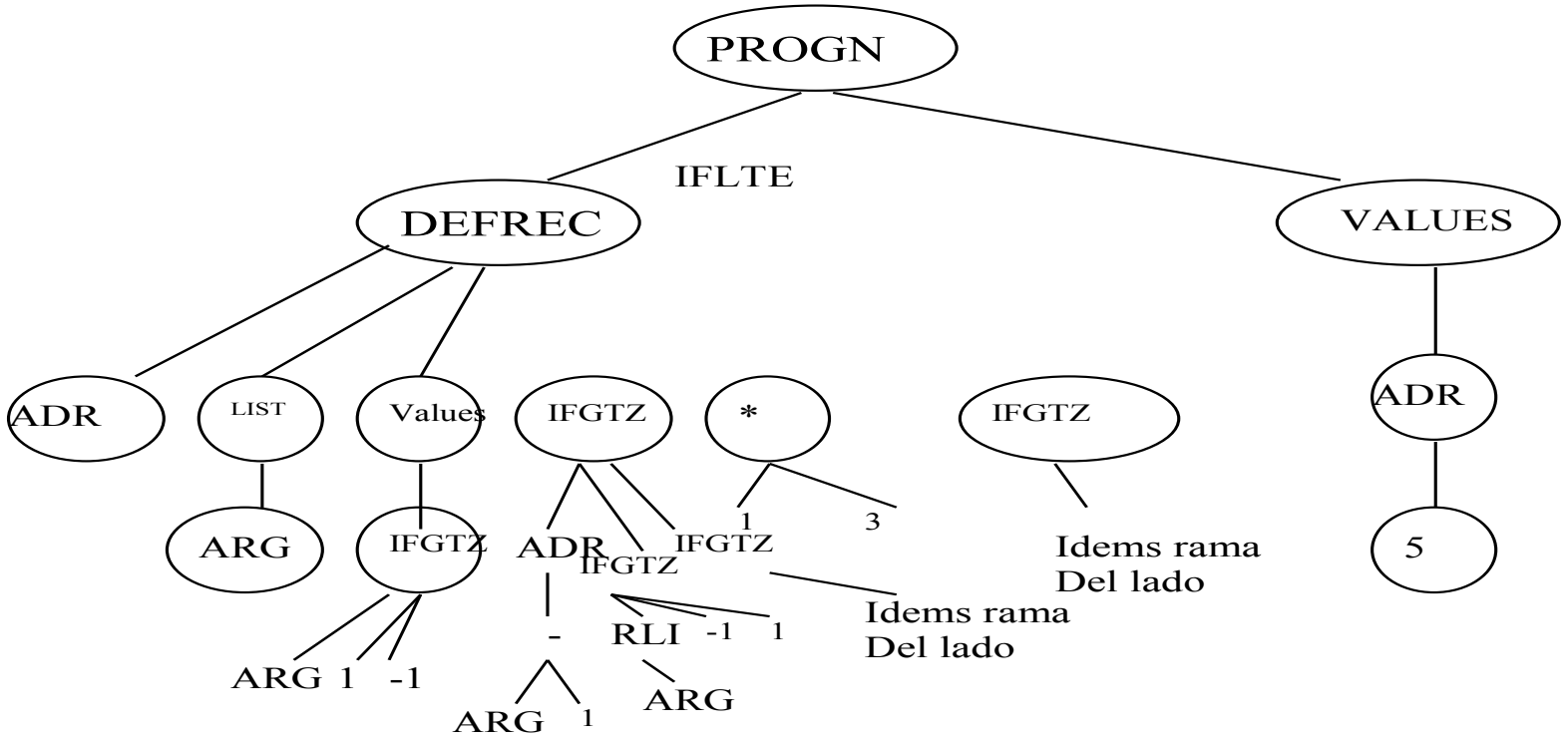
ADR

Partes

- Rama Condicional (RCB)
- Rama Actualizacion (RUB)
- cuerpo (RBB)
- Rama de desarrollo (RGB)

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

PROGRAMACIÓN GENÉTICA



Gramática de una regla (ADR)

IF { ANTECEDENTE } THEN

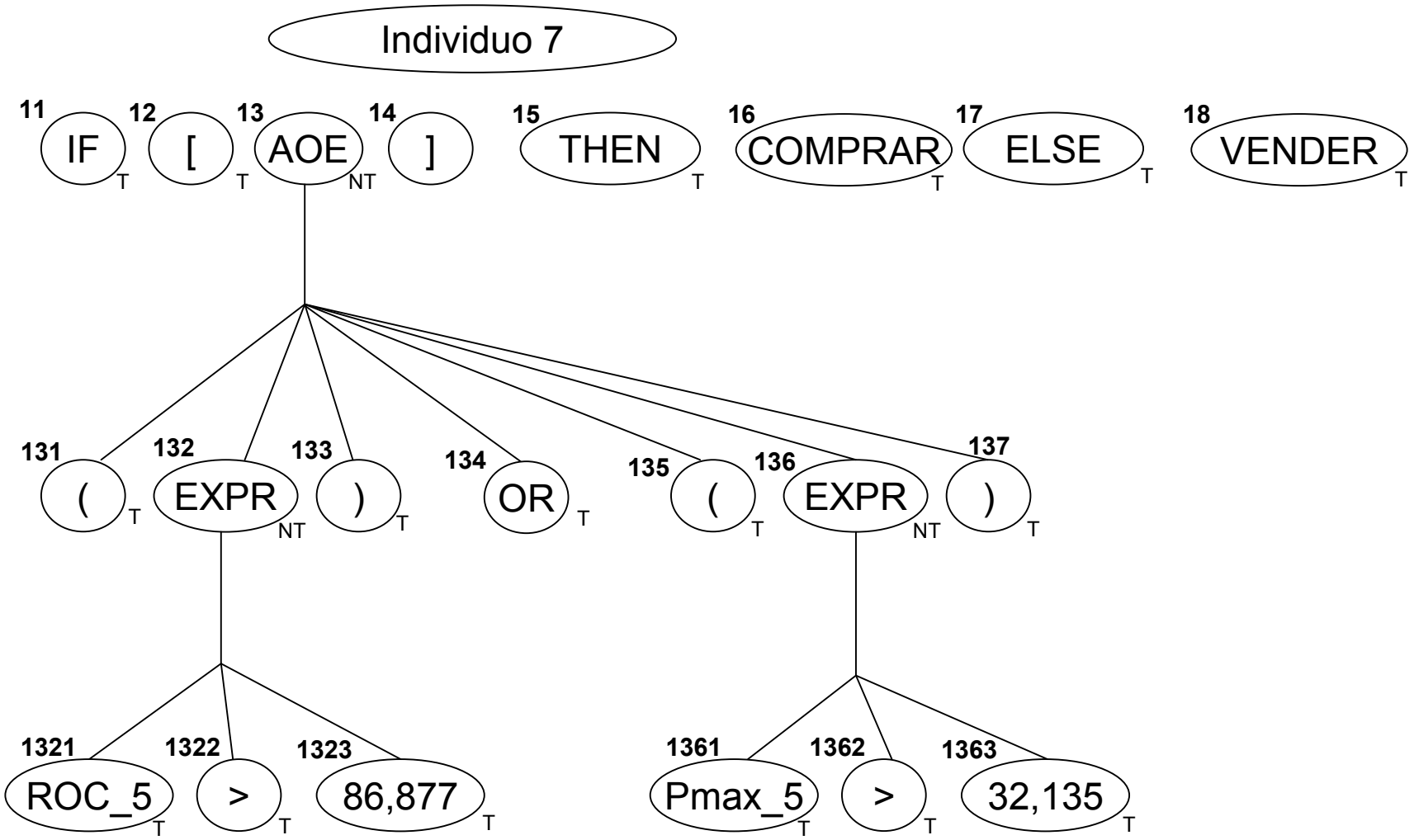
{ CONSECUENTE }

ELSE

{not- CONSECUENTE }

ADR

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


Individuo 7
11
IF
T
12
[
T
13
AOE
NT
14
]
15
THEN
T
16
COMPRAR
T
17
ELSE
T
18
VENDER
T
131
(
T
132
EXPR
NT
133
)
T
134
OR
T
135
(
T
136
EXPR
NT
137
)
T
1321
ROC_5
T
1322
>
T
1323
86,877
T
1361
Pmax_5
T
1362
>
T
1363
32,135
T

PROGRAMACIÓN GENÉTICA

ADR

Operadores

Creación

=> Tomar un pedazo del código y crear ADR

Eliminación

=> regeneración aleatoria del pedazo

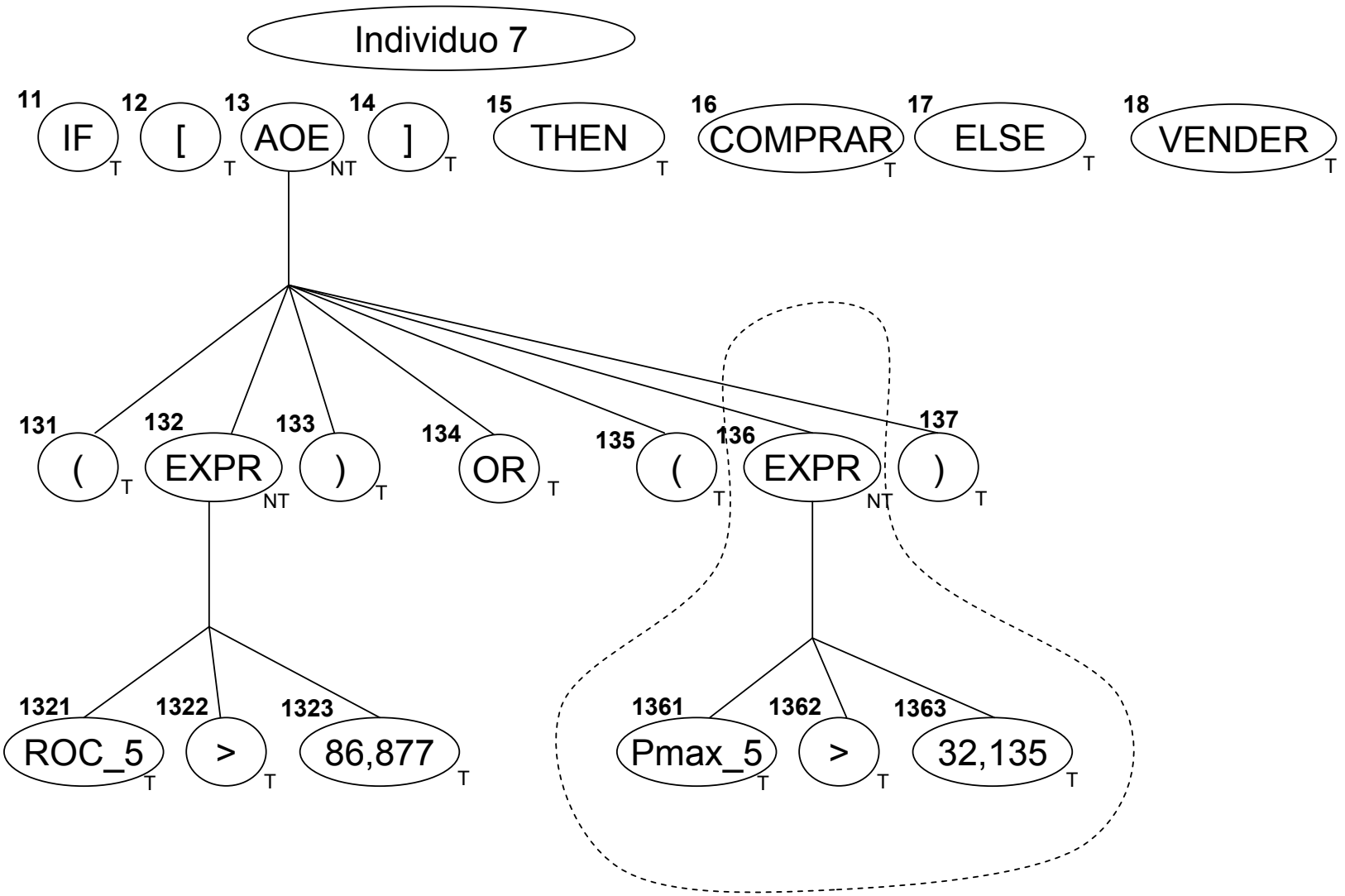
=> modificación conjunto terminal y funciones

Duplicación

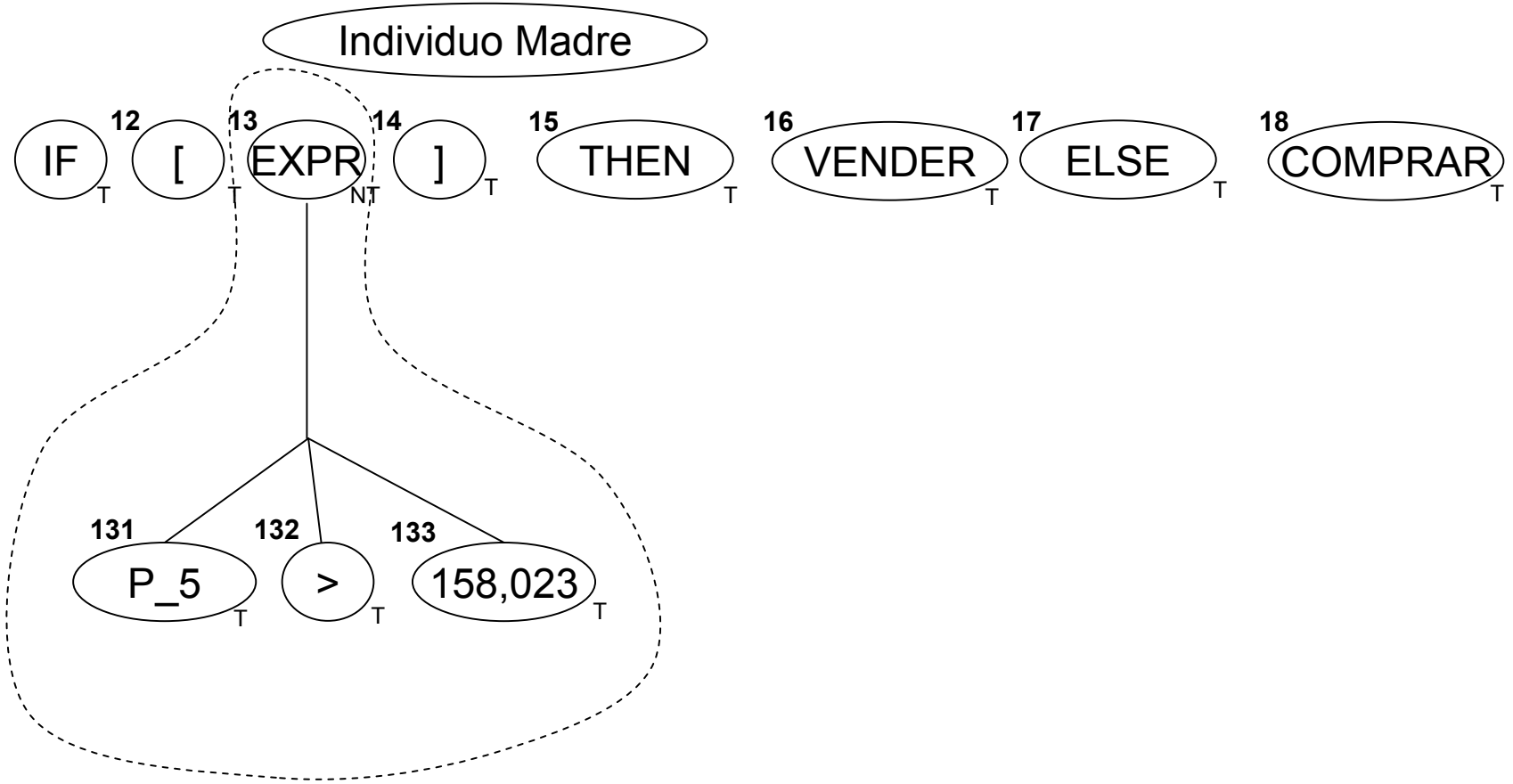
=> Escoger un ADL existente y modificar nombre, etc.

=> Llamados al ADL viejo son aleatoriamente modificados

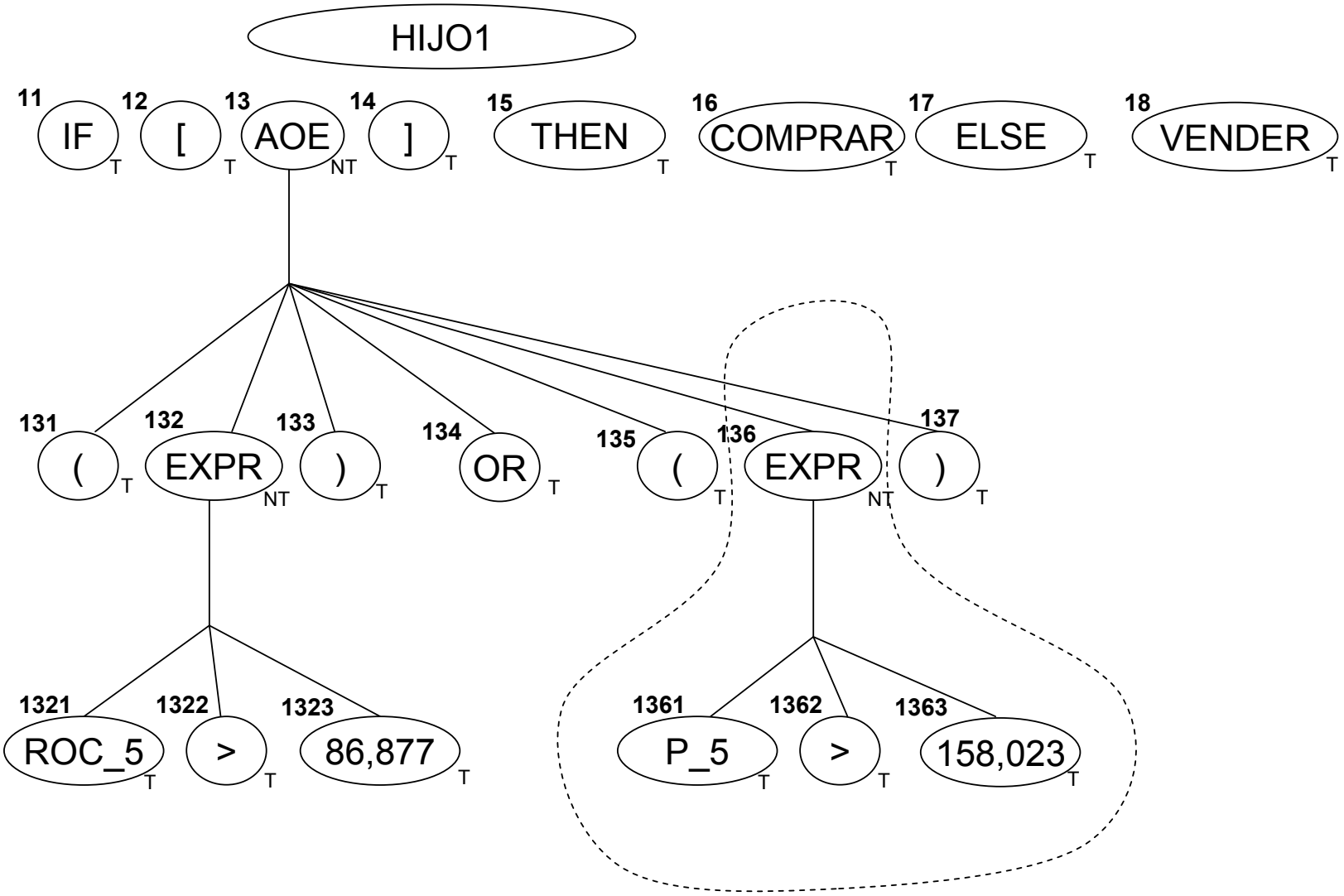
Individuo Padre: IF [(ROC_5 > 86,877) OR (Pmax_5 > 32,135)]
THEN COMPRAR ELSE VENDER



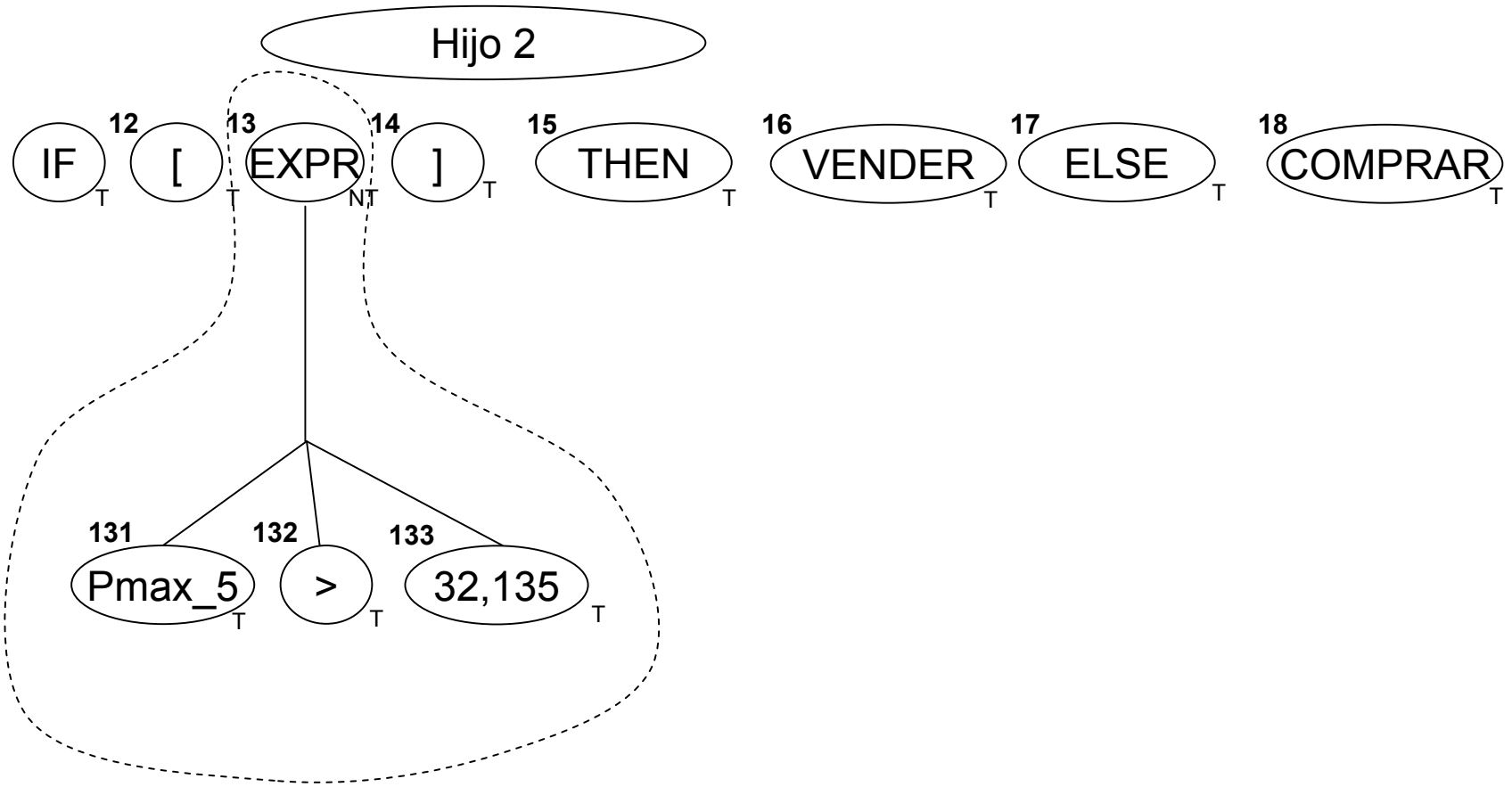
Individuo Madre: IF [P_5 > 158,023] THEN VENDER ELSE COMPRAR



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

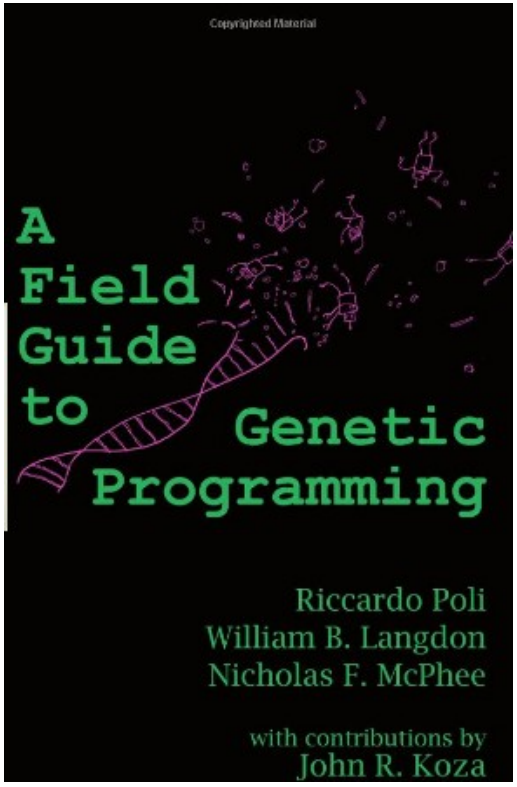
- CONTROL: IDENTIFICACION, ETC.
- MINERIA DE DATOS
- SISTEMAS MULTIAGENTES
- HARDWARE EVOLUTIVO

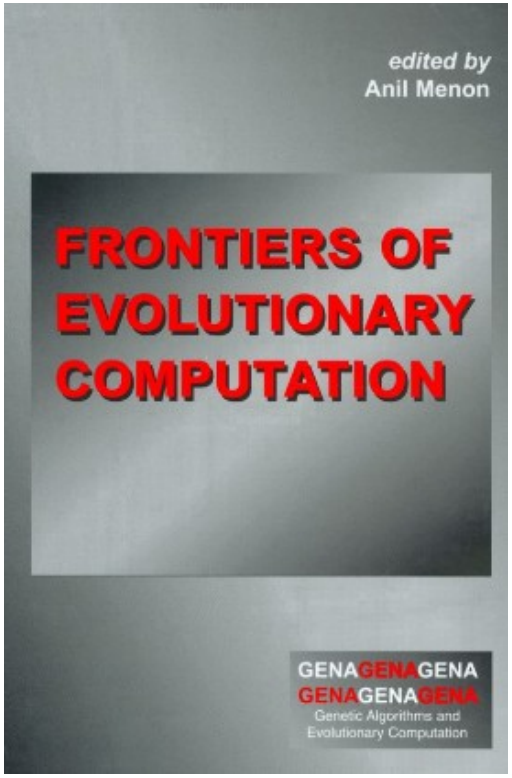
www.aic.nrl.navy.mil/galist

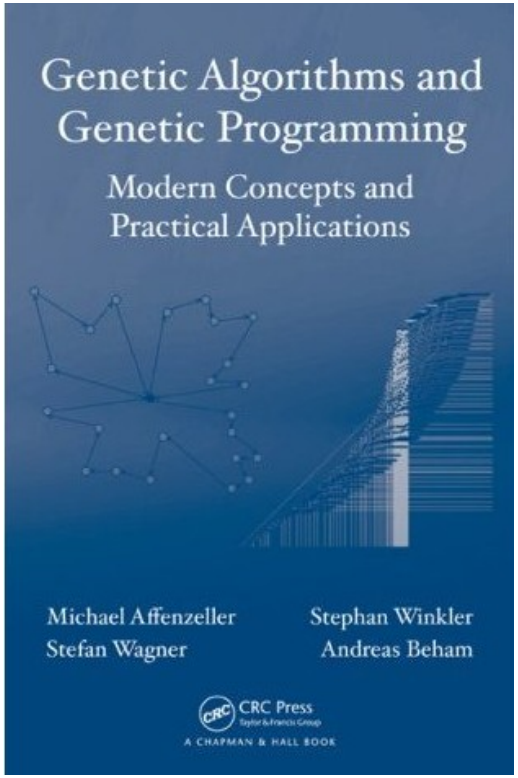
Kernel GPC++

Es una biblioteca de clases escrita en 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







Algoritmos Culturales

El surgimiento de la Inteligencia Social y resolución de problemas socialmente motivados

- Tradicionalmente la Inteligencia Artificial se ha centrado en la naturaleza de la "inteligencia individual".

- El objetivo era crear programas que compiten con los mejores "individual".

Deep Blue en el ajedrez

- Con el advenimiento del alto rendimiento derivado de las redes de computadoras, computación grid, etc., hay mucho incentivo para investigar la resolución de problemas socialmente motivados.

Ejemplo han venido de los sistemas naturales

El surgimiento de la Inteligencia Social y resolución de problemas socialmente motivados.

- Enjambres de partículas en los sistemas físicos.
- Colonia de Hormigas en los Sistemas Biológicos.
- Algoritmos Culturales en los sistemas culturales.

Nos permite comenzar a entender la naturaleza de la inteligencia social en términos de la resolución de problemas en grupo.

Enfoques motivados socialmente

- Particle Swarm Optimization (Kennedy y Eberhart 1995)
  - Sobre la base de los comportamientos sociales de las aves y los peces
  - Partículas (soluciones potenciales) se mueven siguiendo su mejor propio y lo mejor de su vecino

- Ant Colony Optimization (Dorigo et. al. 1996)
  - En base a lo observado en las colonias de hormigas
  - la feromona y de la densidad del feromona por el sendero es el conocimiento de que ACO comparte entre sus hormigas individuales

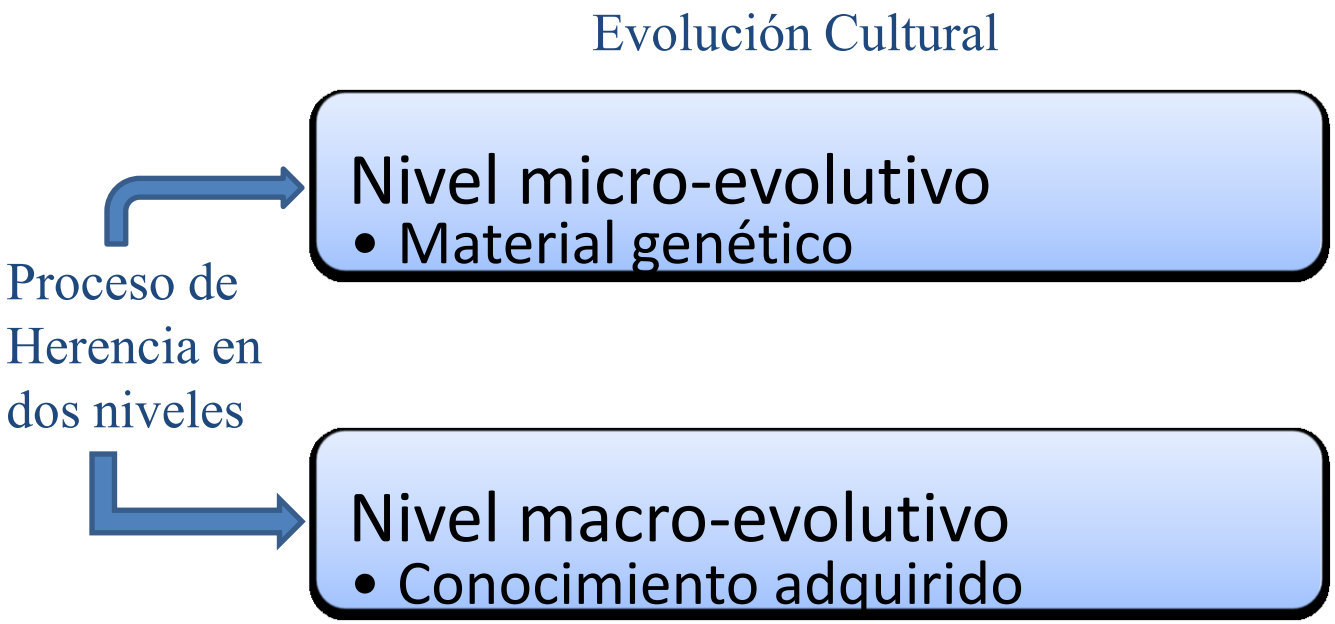
Algoritmos Culturales

- Consisten en una población social y un espacio de creencias [Reynolds, 1979]
- Derivado de modelos de evolución cultural en Antropología [Flannery, 2000, 1986, 1969].
- Proporciona un marco para acumular y comunicar el conocimiento a fin de permitir la auto-adaptación en un modelo de evolución.
- Abarca las escalas de los otros enfoques por lo que debe ser capaz de resolver sus tipos de problemas.

Algoritmos Culturales

Los Algoritmos Culturales (CA), fueron propuestos por Robert Reynolds como un tipo de algoritmo evolutivo en donde el conocimiento del dominio no se integra "*a priori*", sino que se extrae durante el mismo proceso de búsqueda.

Algoritmos Culturales



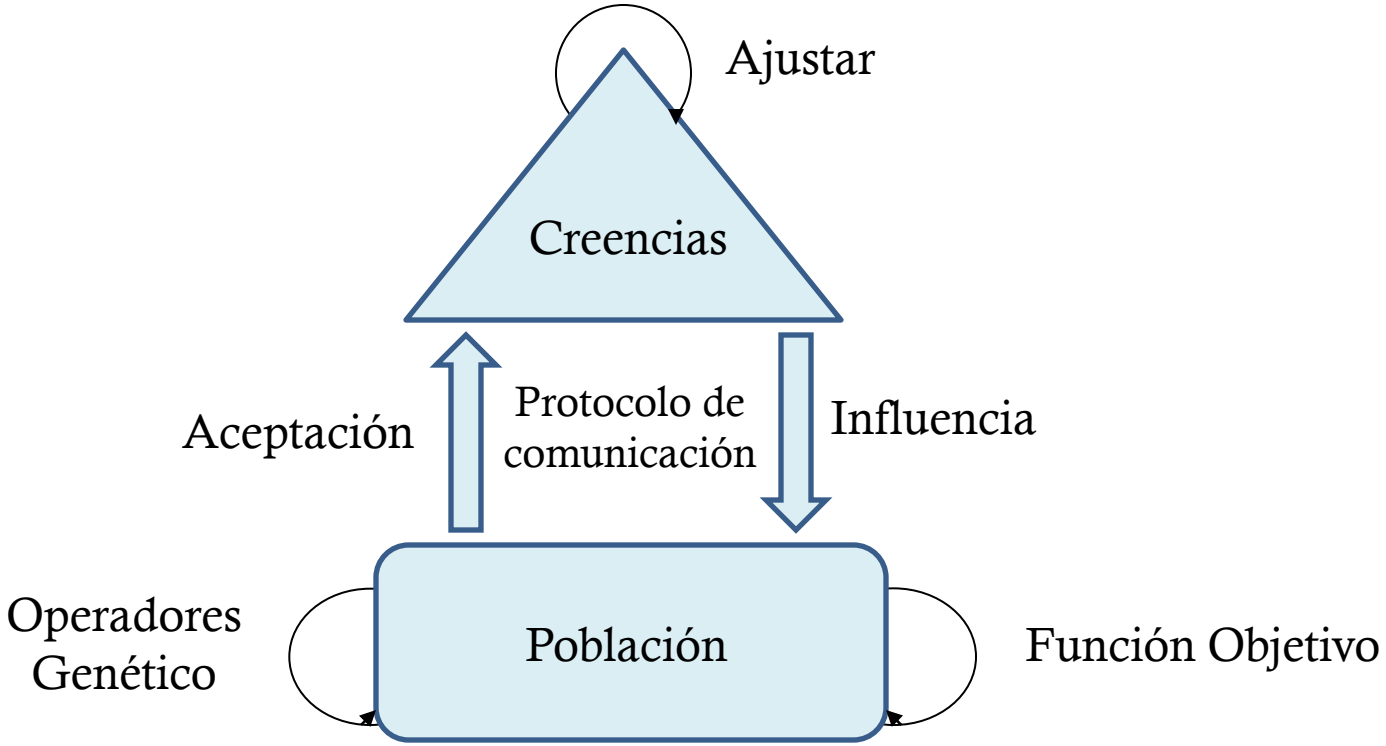
Características Generales

- Herencia Dual (en los niveles de población y de conocimiento)
- Conocimiento son "faros" que guían la evolución de la población
- Soporta estructuración jerárquica en los espacios. de la población y de creencias
- El conocimiento de dominio separado de los individuos (por ejemplo, ontologías)

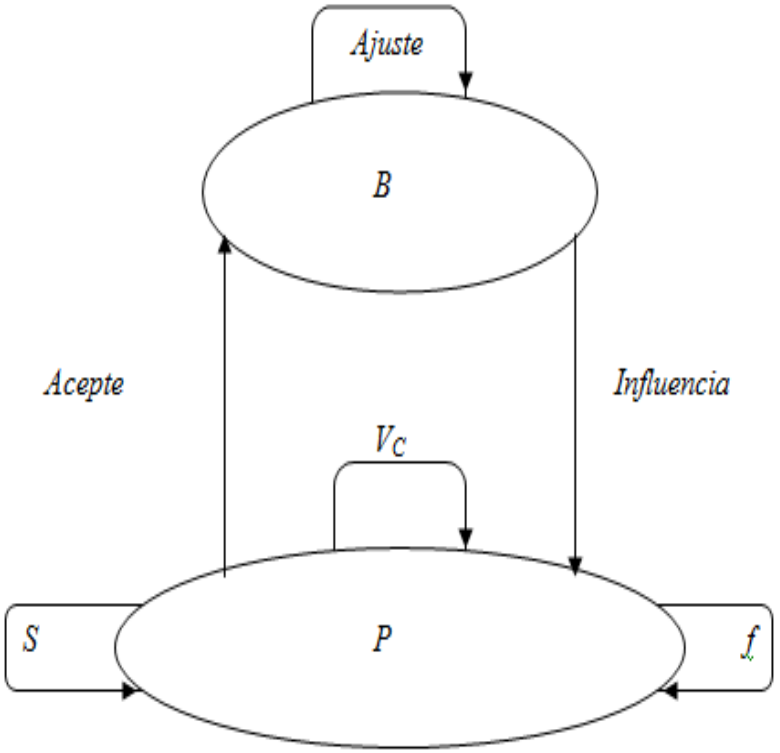
Características Generales

- Soporta auto adaptación a distintos niveles
- La evolución puede tener lugar a diferentes velocidades en diferentes niveles ("La cultura evoluciona 10 veces más rápido que el componente biológico").
- Soporta enfoques híbridos a la resolución de problemas. Un marco computacional en el que todos los modelos de cambio cultural se puede expresar.

Algoritmos Culturales



Algoritmo Cultural



Donde:

P: Representa la población

S: Es el operador de selección

V_C: Es un operador de variación

f: Es la función aptitud

B: Es el espacio de creencias

Acepte: es la función de aceptación

Ajuste: Es un operador del espacio de creencias (B) que actualiza el conocimiento

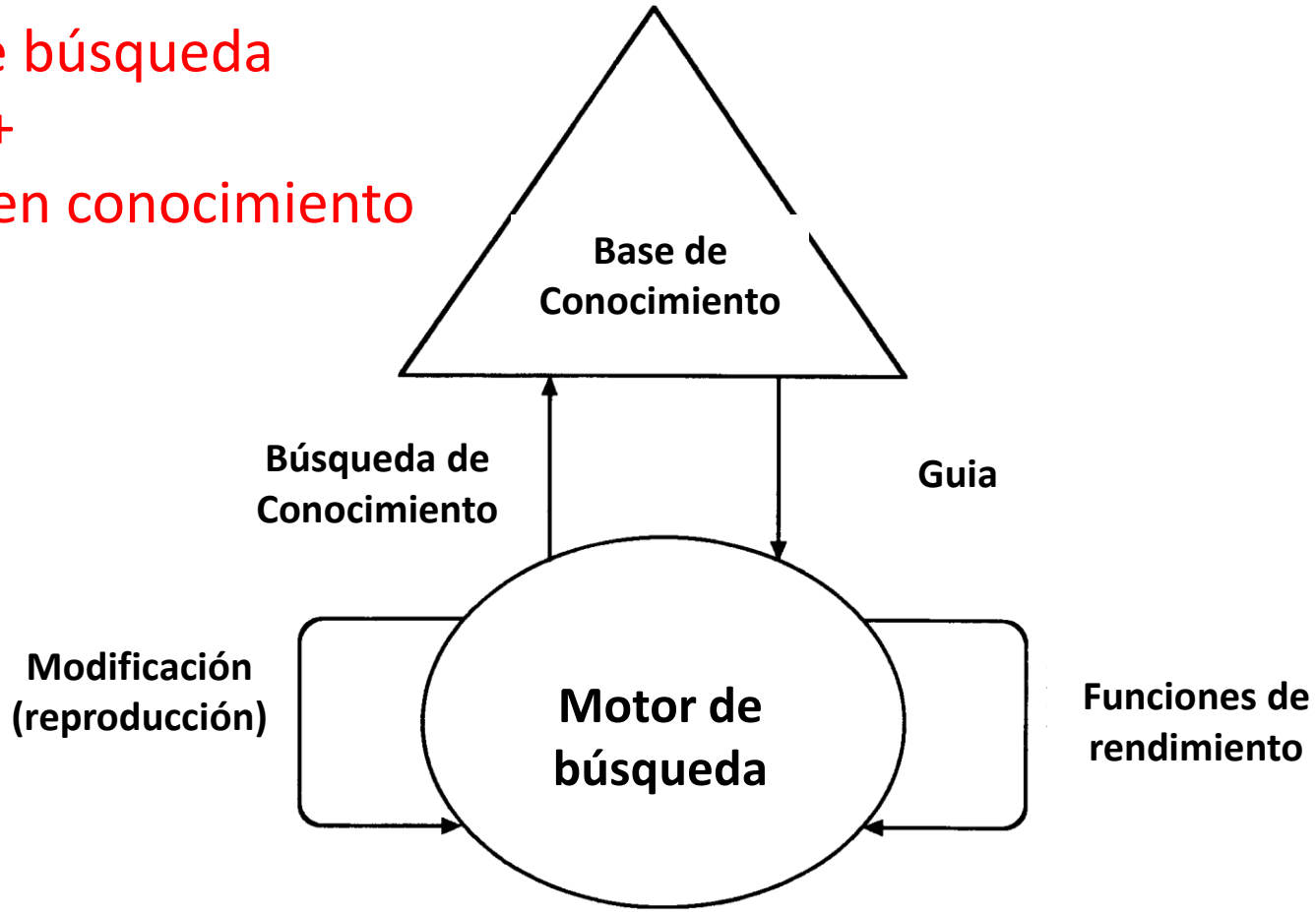
Influencia: es un conjunto de funciones de influencias utilizadas para incidir en el operador de variación V_C.

Sistema Hibrido

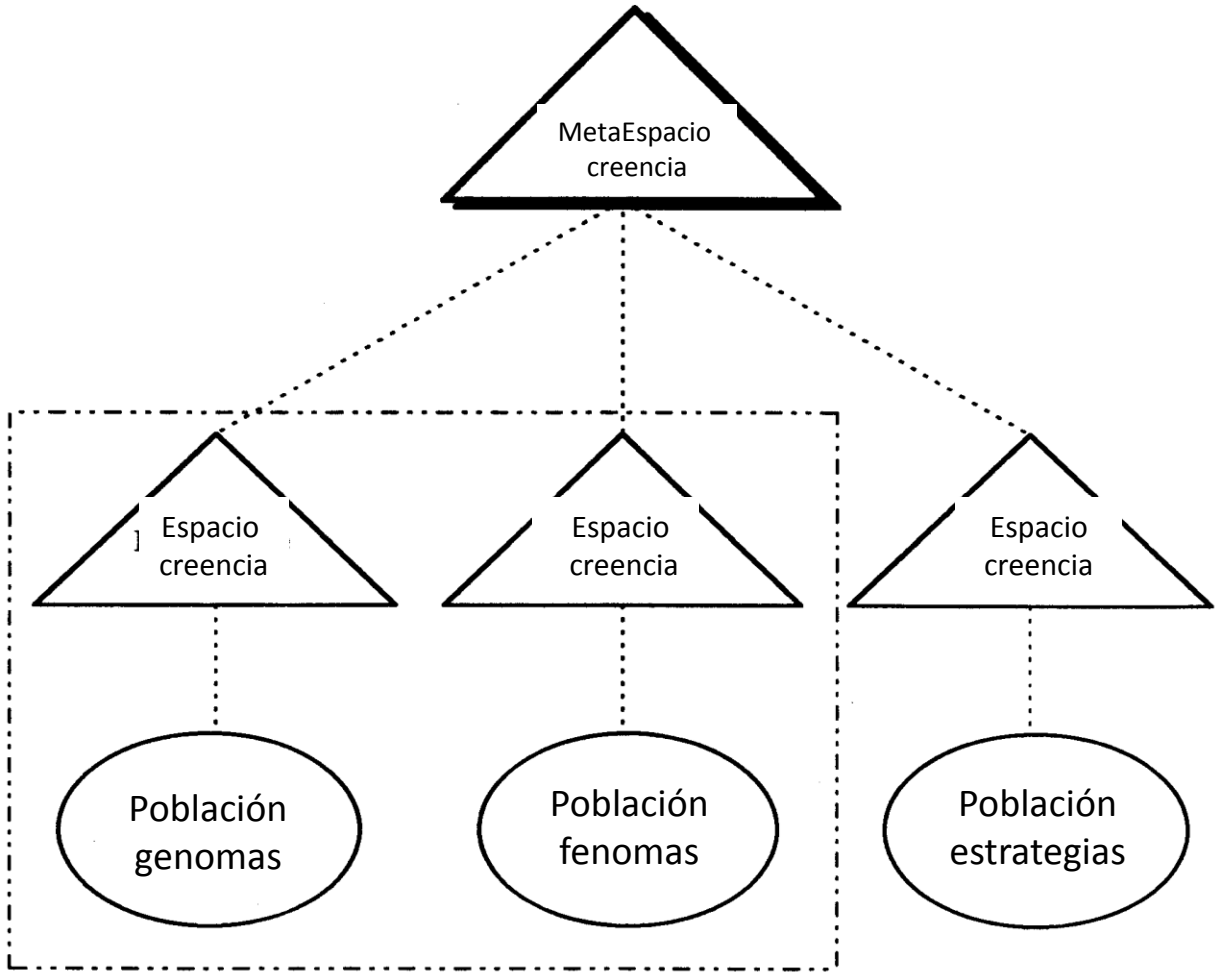
Método de búsqueda

+

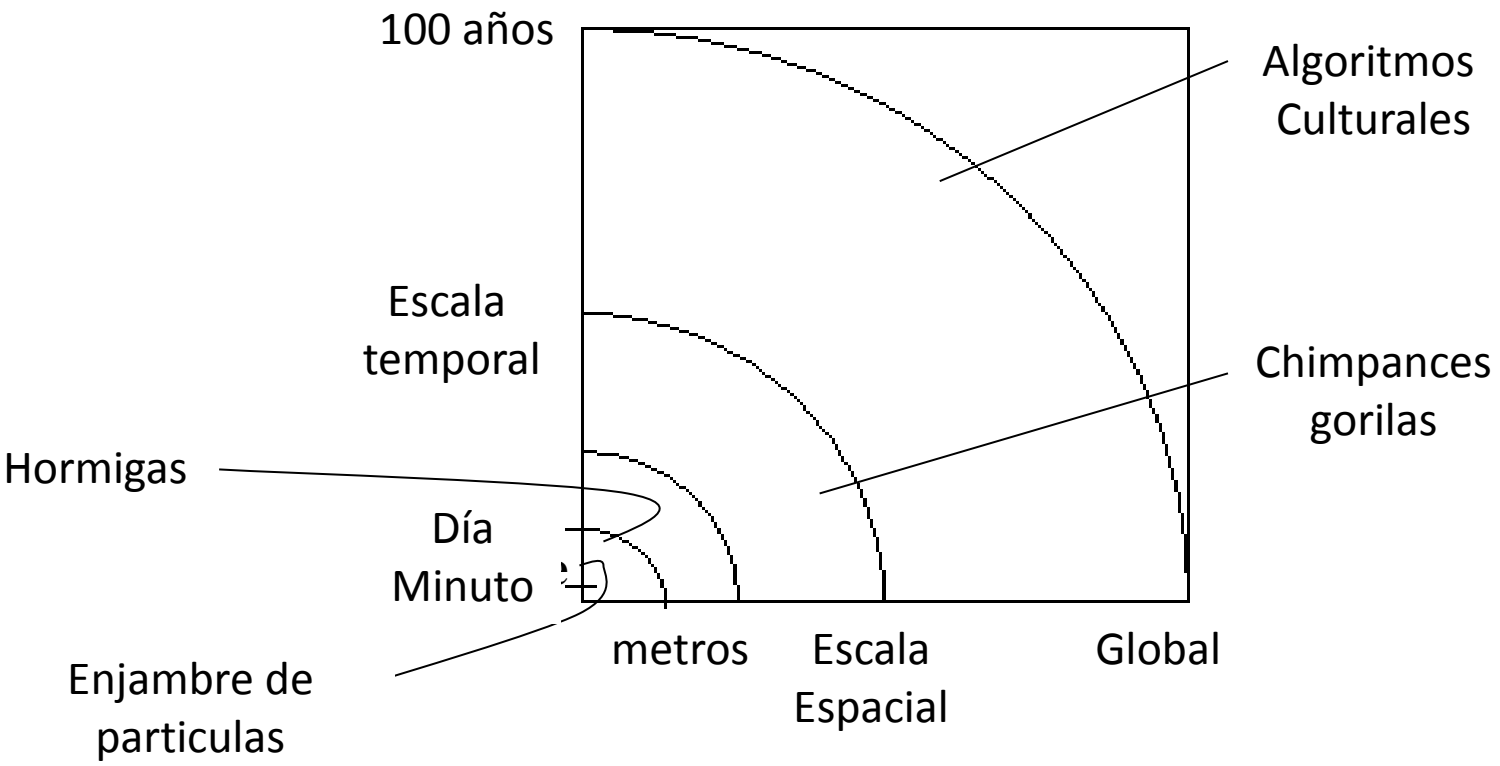
Método basado en conocimiento



Puede apoyar el surgimiento de estructuras jerárquicas, tanto en el espacio de creencia y de la población



Escala de Interacción Social



Algoritmos Culturales

Pseudo-código:

1) Generar población inicial
2) Iniciar el espacio de creencias
3) Evaluar población inicial
4) Repetir
   41) Actualizar el espacio de creencias (con los individuos aceptados)
   42) Aplicar operadores de variación (bajo la influencia del espacio de creencias)
   43) Evaluar cada hijo
   44) Realizar la selección
5) Hasta que no se cumpla la condición de finalización

Espacio de la Población

- Proporciona una base fundamental en la que los individuos residen e interactúan

- Podría adaptarse a cualquier modelo de computación evolutiva basada en la población

Resolución de problemas de optimización

- Una población de individuos se mueven a través de un paisaje caracterizado por una distribución de los recursos en busca de las regiones de alta productividad.

- Movimiento de un individuo es controlado por un modelo de conocimiento.

Función de Aceptación

- Espacio de Población -> Espacio de Creencia

- Se elige un subconjunto de la población de para afectar el espacio de creencias.

- En un problema de optimización en general es un porcentaje de los de mejor desempeño.

Categorías de conocimiento básicos

Clasificación por Saleem & Reynolds

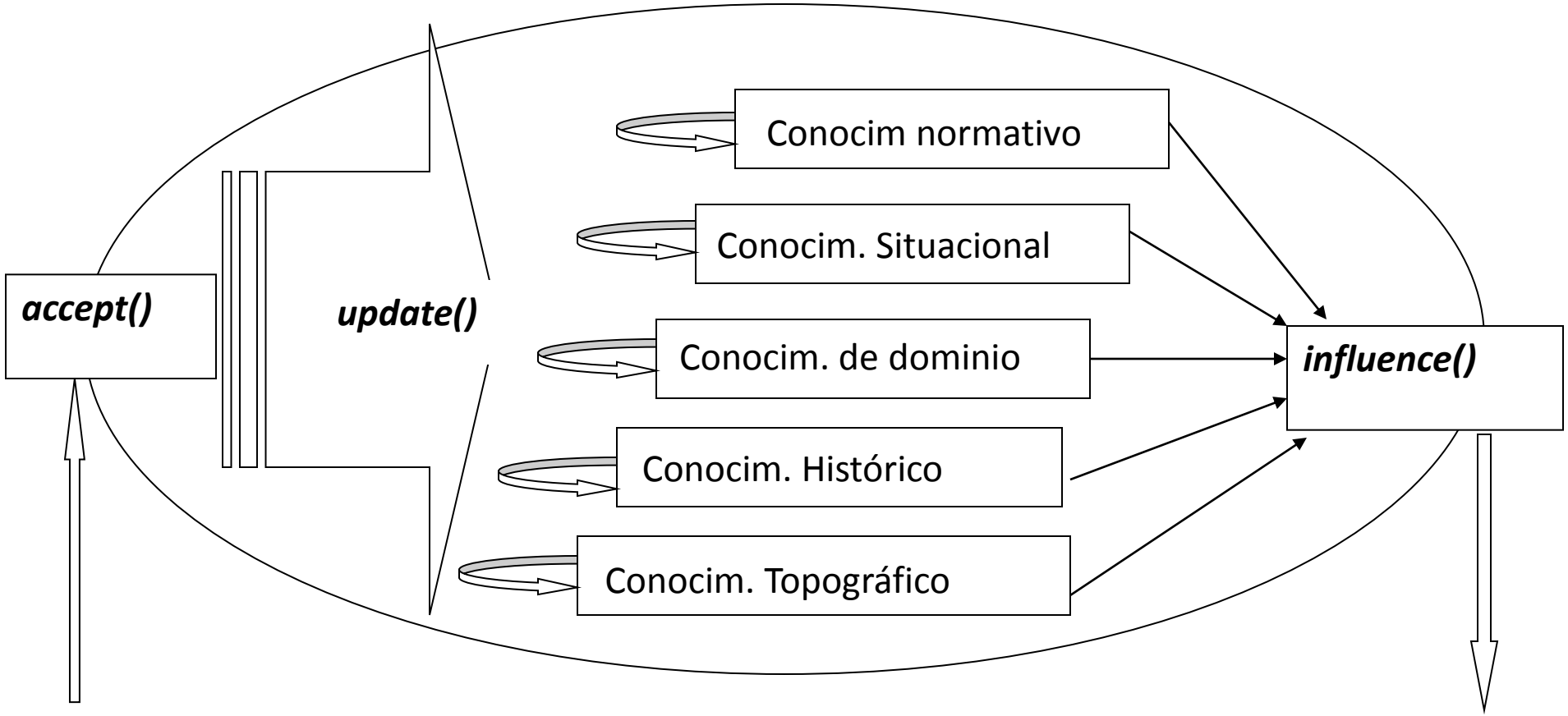
- Normativo
- Situacional
- De dominio
- Histórica
- Topográfica

La clasificación debe ser completa para un dominio dado

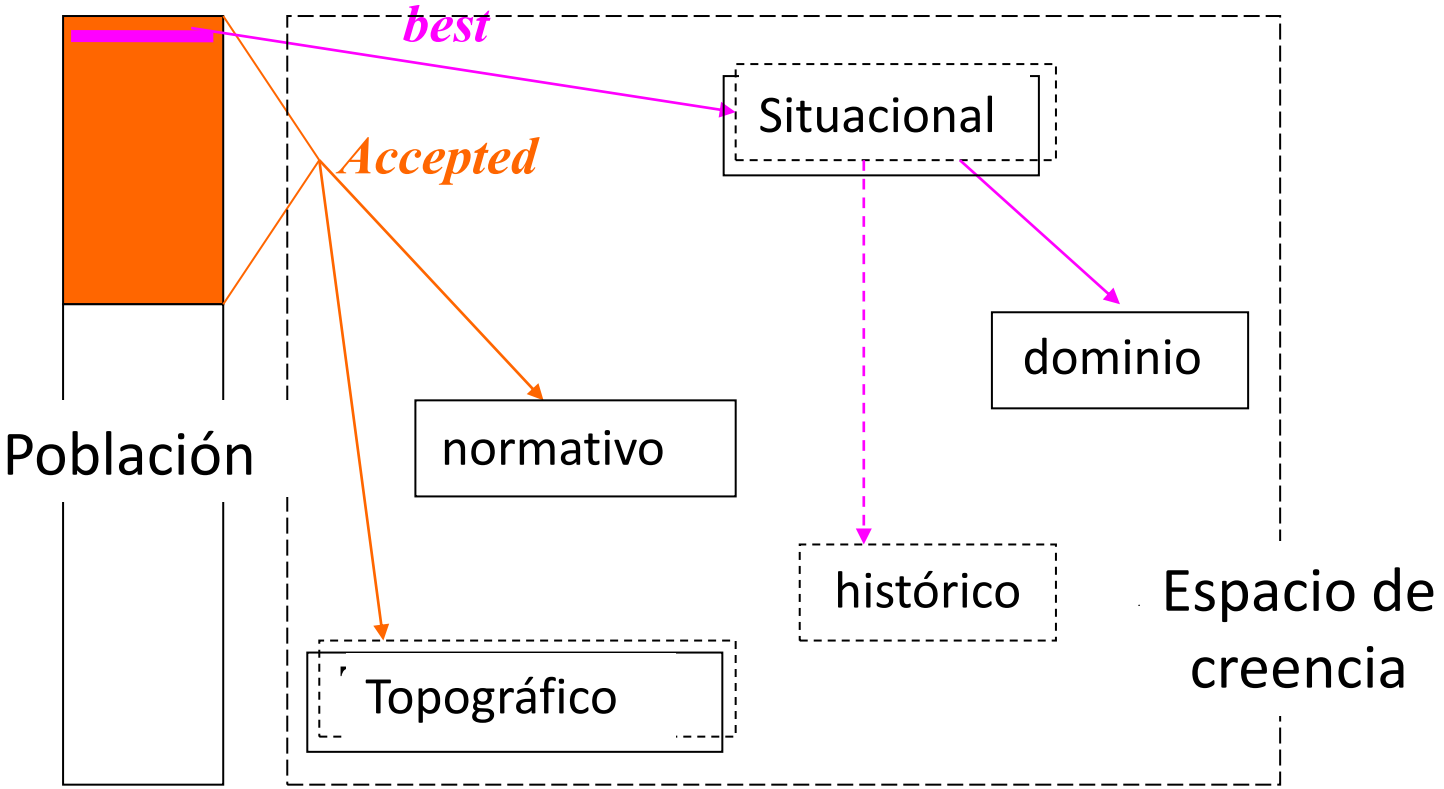
Base cognitiva para cada tipo de conocimiento

- Procedimiento (codificado)
- Declarativa (Bechtel, 1999)
  - Episódica (secuencia recordada)
  - Semántica (hechos conocidos)
- En cuanto a los conocimientos sociales, naturales y técnicos para un dominio dado.
- Predisposición para la adquisición de tipos de conocimiento espaciales, numéricos y comunicacionales. (Landau, 1999)

La integración de las fuentes de conocimiento en la función de influencia



La interacción entre las fuentes de conocimiento



Diseño de Algoritmos Culturales

Diseño del componente de conocimiento

- representación del conocimiento ontológicamente (conceptos comunes compartidos de un dominio)
- Representación del conocimiento sobre las restricciones
- Representación de la Solución
- ¿Qué será modificado? La función de actualización de cada componente modificable.
- Conocimiento del Mantenimiento

Diseño del componente de población

- Variables de estado que determinan el comportamiento de la solución
- ¿Cómo se utilizan esas variables para producir una estrategia de solución a problemas o el comportamiento.
- ¿Cómo este tipo de comportamiento se evalúa?

Diseño de Algoritmos Culturales

- Usar Algoritmos Genéticos como ejemplo de modelo de población.

- Comenzar con el nivel de creencia o el nivel de la población depende del problema. ¿cuál de los dos es más limitado para el problema?

- Problemas de Clasificación vs. Problemas de Construcción. Para el primero comenzar con el espacio de creencias, mientras que para segundo es mejor el espacio de la población.

Rol de la Emergencia

Dos roles:

- "Exploradores" utilizan una combinación de fuentes de conocimiento a través del tiempo, y "explotadores“ explotan una.

- Los biólogos han observado esos dos roles emergentes en búsqueda de alimento, las especies "productoras" y los "vividoras” (Barnard y Sybley, 1981)

- Estos roles son producidos por la interacción de las fuentes de conocimiento y nos permite explicar el surgimiento de los roles en términos de la experiencia o conocimiento compartido.

Problemas Adecuados

- Cantidad importante de conocimientos de dominio (por ejemplo, problemas de optimización con restricciones).
- Sistemas Complejos donde la adaptación puede ser a diferentes niveles y a diferentes tasas en la población y en el espacio de creencias.
- El conocimiento es en diferentes formas y necesita ser razonado de diferentes maneras.
- Sistemas híbridos que requieren una combinación de búsqueda y conocimiento.
  Solución del problema requiere de múltiples poblaciones y múltiples espacios de creencias y su interacción.
- Problemas jerárquicamente estructurados donde elementos de la población y del conocimiento pueden surgir.

Ejemplo de Aplicación

Proponer un modelo de optimización de aprendizaje de esquemas de coordinación para sociedades de agentes (SMA) usando algoritmos culturales, y aplicar dicho modelo en el problema de integración en automatización

J. Terán, J. Aguilar, M. Cerrada“Collective Learning inMulti-Agent Systems Based on Cultural Algorithms”, *CLEI Electronic Journal*, Vol. 17, No 2, Agosto 2014

Tipos de Conversación

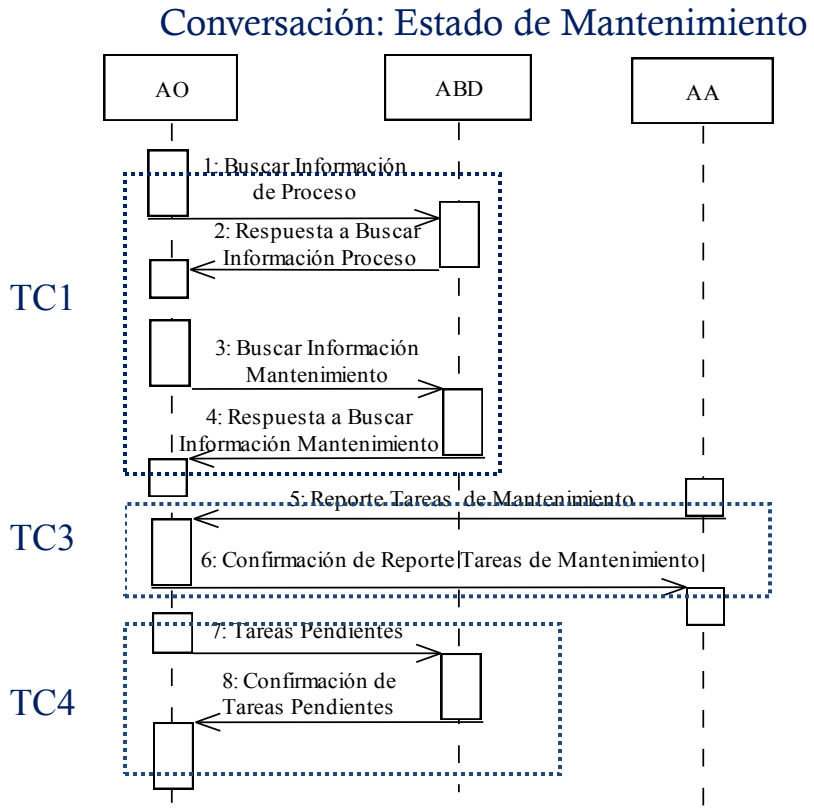
De las interacciones entre los agentes de una comunidad (Sistemas Multiagentes) nacen conversaciones

TC1: Consulta

TC2: Asignación

TC3: Información

TC4: Solicitud

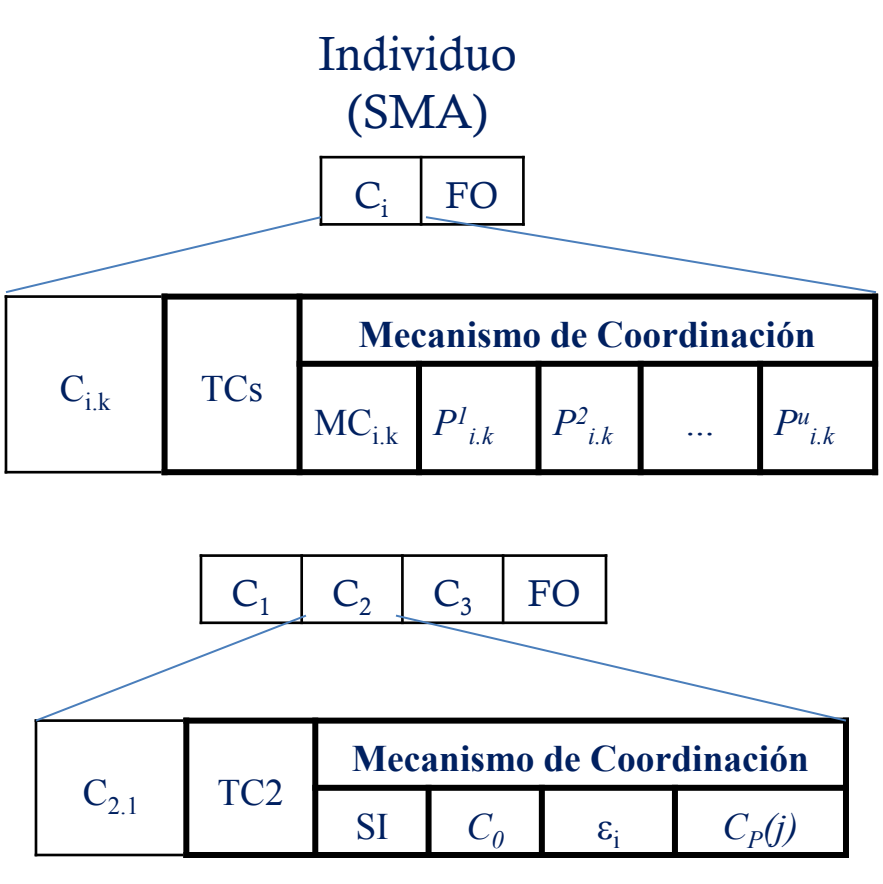


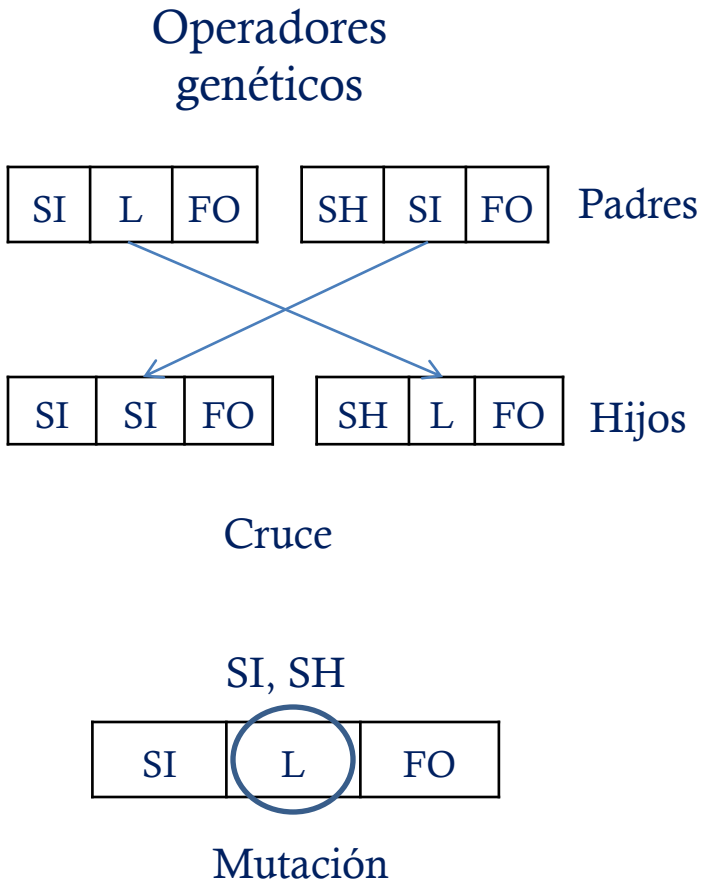
Parámetros de los Mecanismos de Coordinación

Mecanismos de Coordinación (MC)	Parámetros (P)		
Subasta (SI/SH)	C_0	ε_i	$C_P(j, t, x_r)$
Licitación (L)	$M(f)$	$f(T_h)$	-

- C_0 es el precio inicial.
- ε_i es la máxima cantidad que un agente puede ofertar.
- $C_P(j, t, x_r)$ es la condición de parada y puede ser por número de rondas, tiempo, o un valor umbral respectivamente.
- $M(f)$ es el mensaje inicial que contiene la fecha de expiración para el recibimiento de ofertas.
- $f(T_h)$ es una función que permite a los potenciales contratistas, evaluar sus capacidades para responder a la notificación de la solicitud de realización de una tarea T_h

Población


Individuo
(SMA)
C_i
FO
$C_{i.k}$
TCs
Mecanismo de Coordinación
$MC_{i.k}$
$P^1_{i.k}$
$P^2_{i.k}$
...
$P^u_{i.k}$
C_1
C_2
C_3
FO
$C_{2.1}$
TC2
Mecanismo de Coordinación
SI
C_0
ε_i
$C_P(j)$


Operadores
genéticos
SI
L
FO
SH
SI
FO
Padres
SI
SI
FO
SH
L
FO
Hijos
Cruce
SI, SH
SI
L
FO
Mutación

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

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

- Donde a y b son constantes definidas por el usuario y permiten normalizar las unidades de la función
- Conversaciones i = {1 … n}
- Subconversaciones 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 (subasta); especificación de condiciones en las que se requiere un servicio (Licitación)
- PE proceso de selección del agente ganador
- $A_{l,q}$ tiempo para preparar propuestas

Mec. De Coord.	PI	PE	$A_{l,q}$
Licitación (L)	Medio	Medio	Medio
Subasta Inglesa u Holandesa (SI,SH)	Bajo	Medio	Bajo

Costo de Comunicación

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

- CEP costo de envío de propuesta
- CEO costo de envío de ofertas
- CS costo de informar al ganador

Mec. De Coord.	CEP	CEO	CS
Licitación (L)	Bajo	Medio	Bajo
Subasta Inglesa u Holandesa (SI,SH)	Bajo	Medio	Bajo

Alto = 1; Medio = 0.6; Bajo = 0.2

Espacio de creencias

Conocimiento Circunstancial/situacional

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

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

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

Espacio de creencias

Conocimiento Normativo

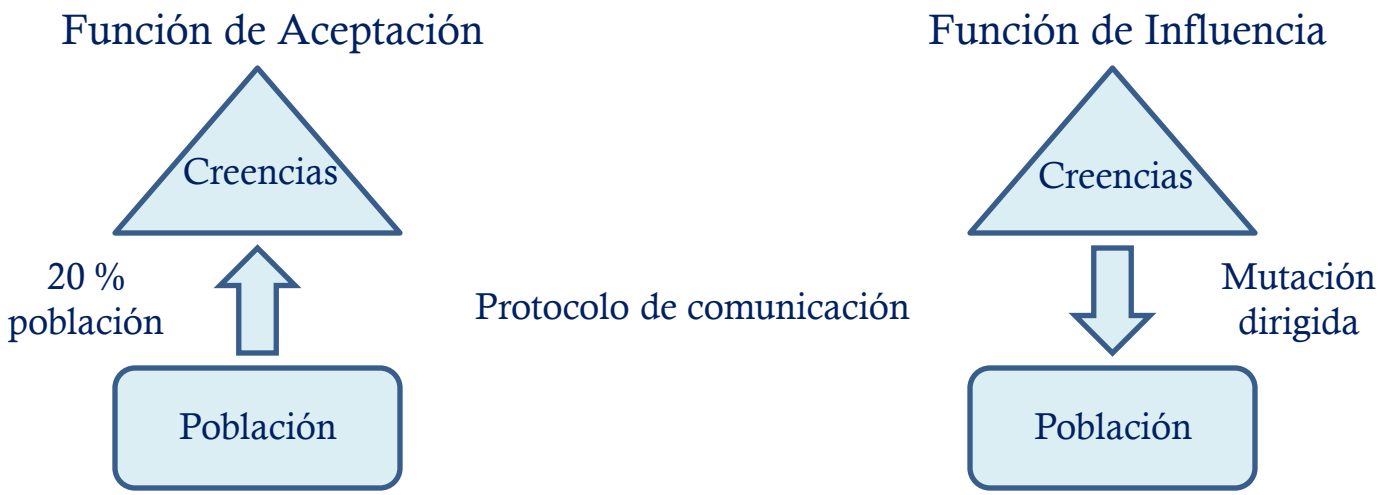
Son rangos de valores idóneos de cada una de las variables que integran al mecanismo de coordinación usado por los individuos

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

Protocolo de comunicación

Permite la interacción entre los dos espacios (creencias y población), y esta compuesto por dos funciones: de aceptación y de influencia



Protocolo de comunicación

Función de Aceptación

Conocimiento Circunstancial	Conocimiento Normativo
$IO_{(TC, MC, t)} = IO_{(TC, MC, t-1)} + (NO_{(TC, MC, t)} / TO_{(TC)})$	$Lac(P^{u}) = [(lv * \overline{m} + \overline{P} * m) / 2]$
$TO_{(TC)} = TO_{(TC)} + \Sigma^{k}{}_{i=1}\, NO_{(TC, MC, t)i}$	$\overline{m} = (1 - m)$
	$m = \mu / t$

Protocolo de comunicación

Función de Influencia

Conocimiento Circunstancial en el Individuo

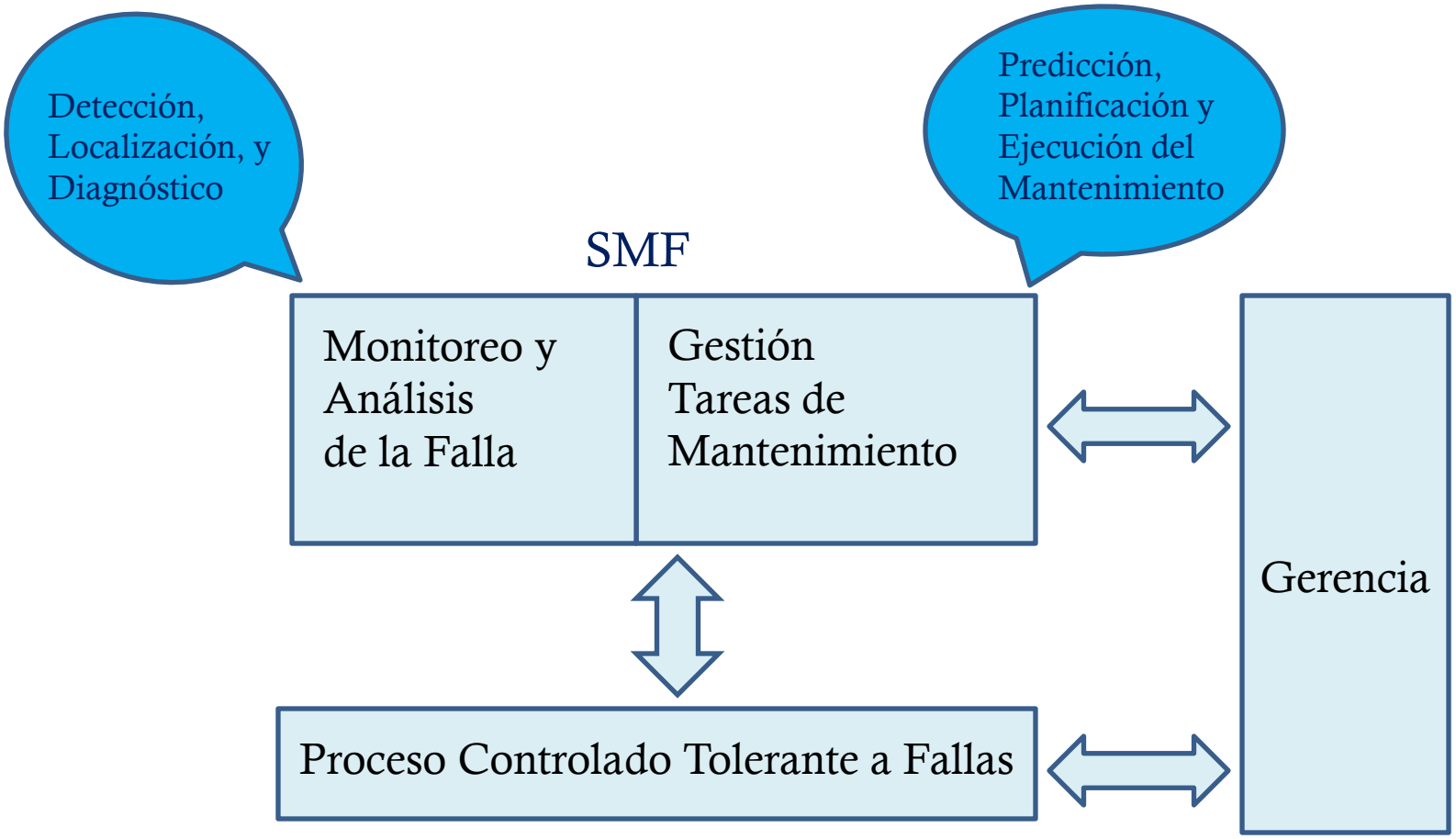
$C_{2.1}$	TC2	Mecanismo de Coordinación			
		SI	C_0	ε_i	$C_P(j)$

Conocimiento Normativo en el Individuo

$C_{2.1}$	TC2	Mecanismo de Coordinación			
		SI	C_0 (LI, LS)	ε_i (LI, LS)	$C_P(j)$ (LI, LS)

Caso de Estudio

Sistema Manejador de Fallas (SMF) basado en un Sistema Multi-Agente

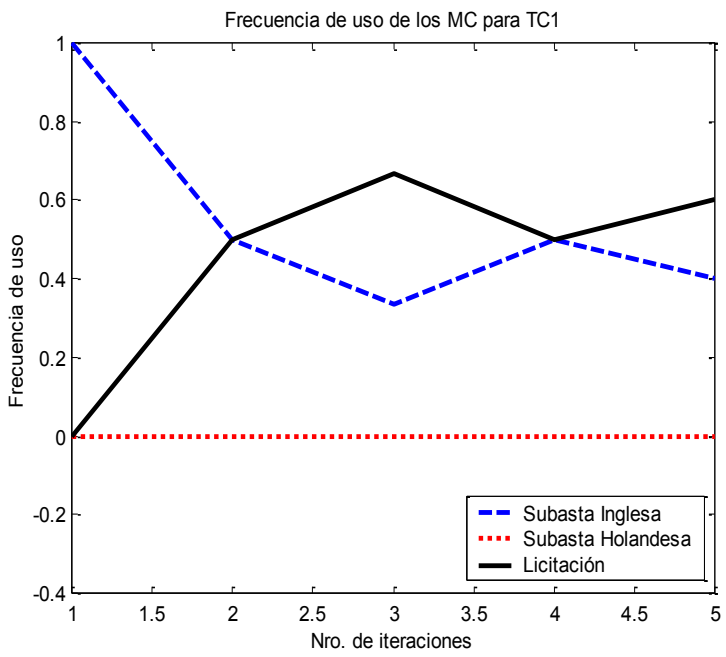


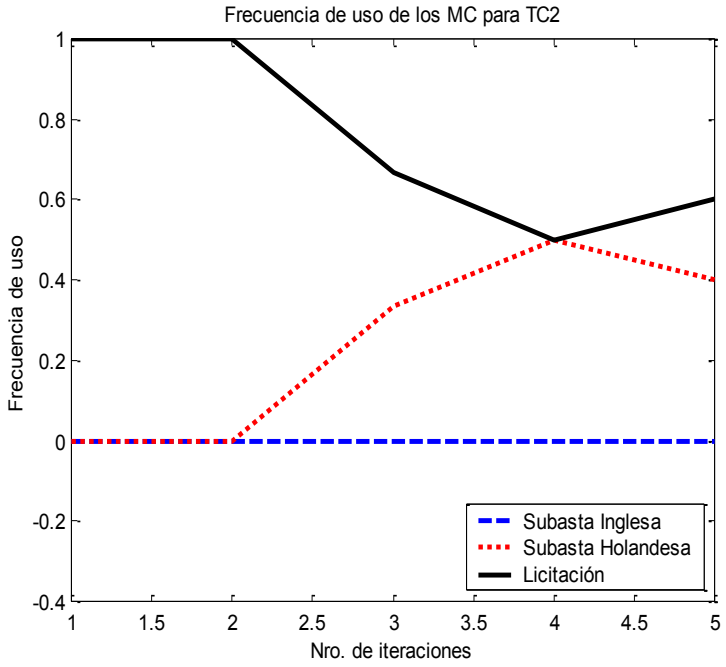
Resultados

		C_1 (TC1)	C_2 (TC2)	-	FO	TC	MC	*IO*	*TO*				
2da	1	L	L	-	12,0	TC1	L	1	1	L	*M(f)*	*f(Th)*	-
	2	SI	SI	-	*FO*						[0,5…1,5]	[2,5…35]	
	3	L	SH	-	*FO*	TC2	L	1	2	L	*M(f)*	*f(Th)*	-
	4	C_1	C_2	-	*FO*						[0,5…1,5]	[2,5…35]	
	5	C_1	C_2	-	*FO*								
3ra	1	L	SH	-	25,2	TC1	L	1	2	L	*M(f)*	*f(Th)*	-
	2	L	L	-	*FO*						[0,25…0,75]	[1,25…17,5]	
	3	C_1	C_2	-	*FO*	TC2	SH	1	1	SH	C_0	ε_i	$C_P(j)$
	4	C_1	C_2	-	*FO*						[2,5…7,5]	[2,5…35]	[0,5…2]
	5	C_1	C_2	-	*FO*								

Segunda y tercera iteración

Resultados





Gráficas para TC1 y TC2

Algoritmos Basados en Evolución Diferencial

Evolución diferencial

los operadores de cruce aritméticos se usan en lugar de los operadores geométricos como corte e intercambio.

Evolución diferencial

- Propuesto por Storn y otros en 1995

- Considerado como uno de los más poderosos algoritmos evolutivos para optimización de problemas con funciones de número reales de hoy en día

"Differential Evolution -- a simple and efficient adaptive scheme for global optimization over continuous spaces, 1995”

Evolución diferencial

- Premisas
  - Individuos en punto flotante
  - Elegantemente sustituye a las dos operadores clásicos:
    cruce
    mutación
  - Menos parámetros a ser entonados
  - Capacidad de auto-organización

Evolución diferencial

- Idea operador

  Si aleatorio[0,1]<CR o j=jrand entonces

  - $x'_{ij}(t)= x_{i3j}(t)+\beta(x_{i1j}(t)-x_{i2j}(t))$

  De lo contrario

  - $x'_{ij}(t)= x_{ij}(t)$

- CR y β variables de control, jrnad=int(aleatorio[0,1]+D)+1,
- D es dimensión indiv.
- $x_i \neq x_{i3} \neq x_{i1} \neq x_{i2}$
- Eso se hace para cada parametro j del individuo $x_i(t)$ cuya dimensión es D

Evolución diferencial

- Definir tamaño población, D, CR, Gmax
- mientras t<Gmax

  para cada individuo de la poblaciónSi aleatorio[0,1]<CR o j=jrand entonces
  - Ejecutar operador genetico para cada dimensión D
  - Seleccionar

    Si $f(x'_{ij}(t)(\leq f(x_{ij}(t))$

    $x_{ij(}(t+1)= x'_{ij}(t)$

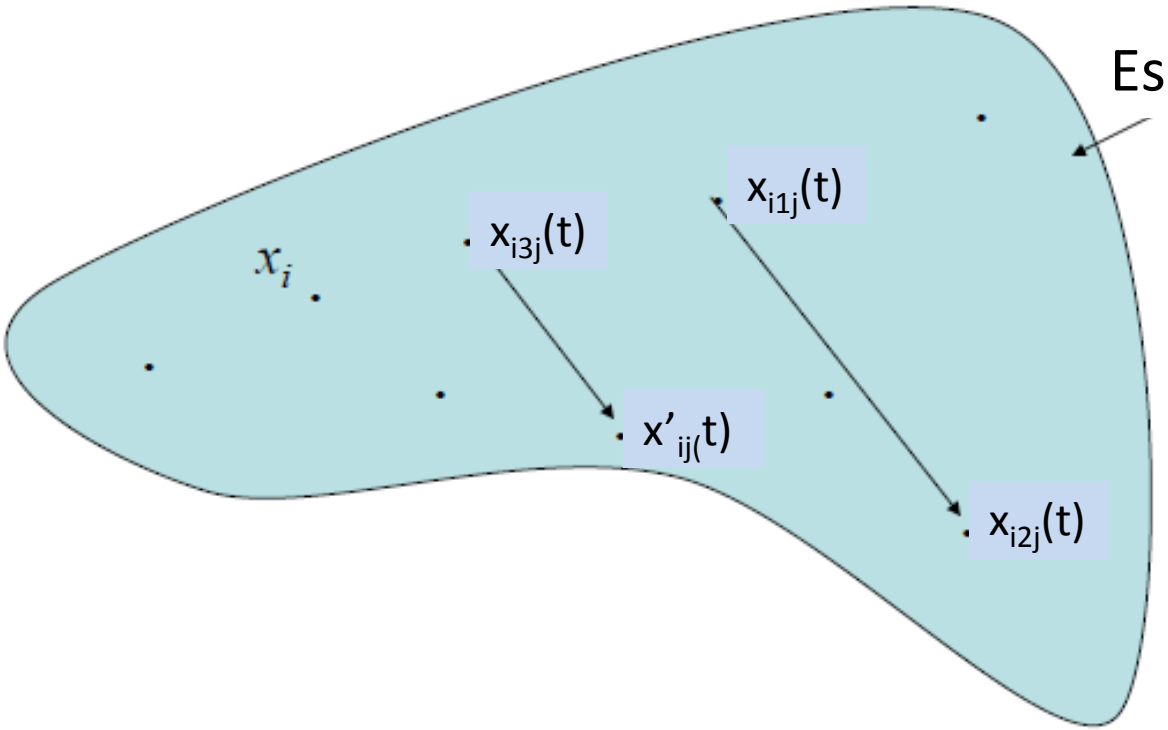
    De lo contrario

    $x_{ij(}(t+1)= x_{ij}(t)$

  t=t+1

Evolución diferencial

$x'_{ij(}t) = x_{i3j}(t) + \beta(x_{i1j}(t) - x_{i2j}(t))$



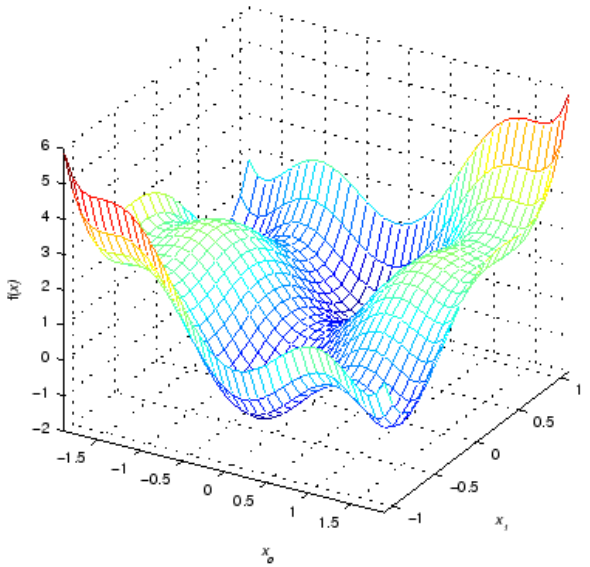
Evolución diferencial

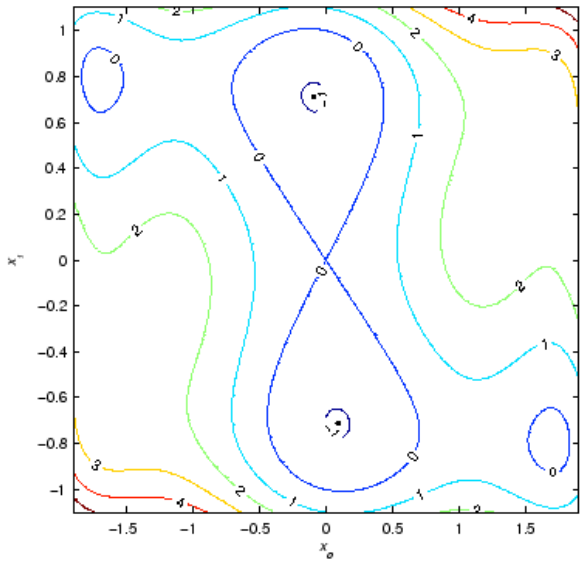
- Dado un problema de optimización, los algoritmos de optimización tradicionales se pueden aplicar para obtener una óptima.

- Sin embargo, en el mundo real, a menudo nos interesa no sólo un único óptimo, sino también otros posibles óptimos globales y locales.

Evolución diferencial

- Dada una función, un algoritmo debería determinar todos los puntos óptimos en una única corrida.





ESTRATEGIAS EVOLUTIVAS

Son métodos estocásticos con paso adaptativo, que permiten resolver problemas de optimización paramétrica.

A estos métodos se le han incorporado procedimientos propios de la CE que los han convertido en un paradigma mas de dicha metodología.

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 mutacion:

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

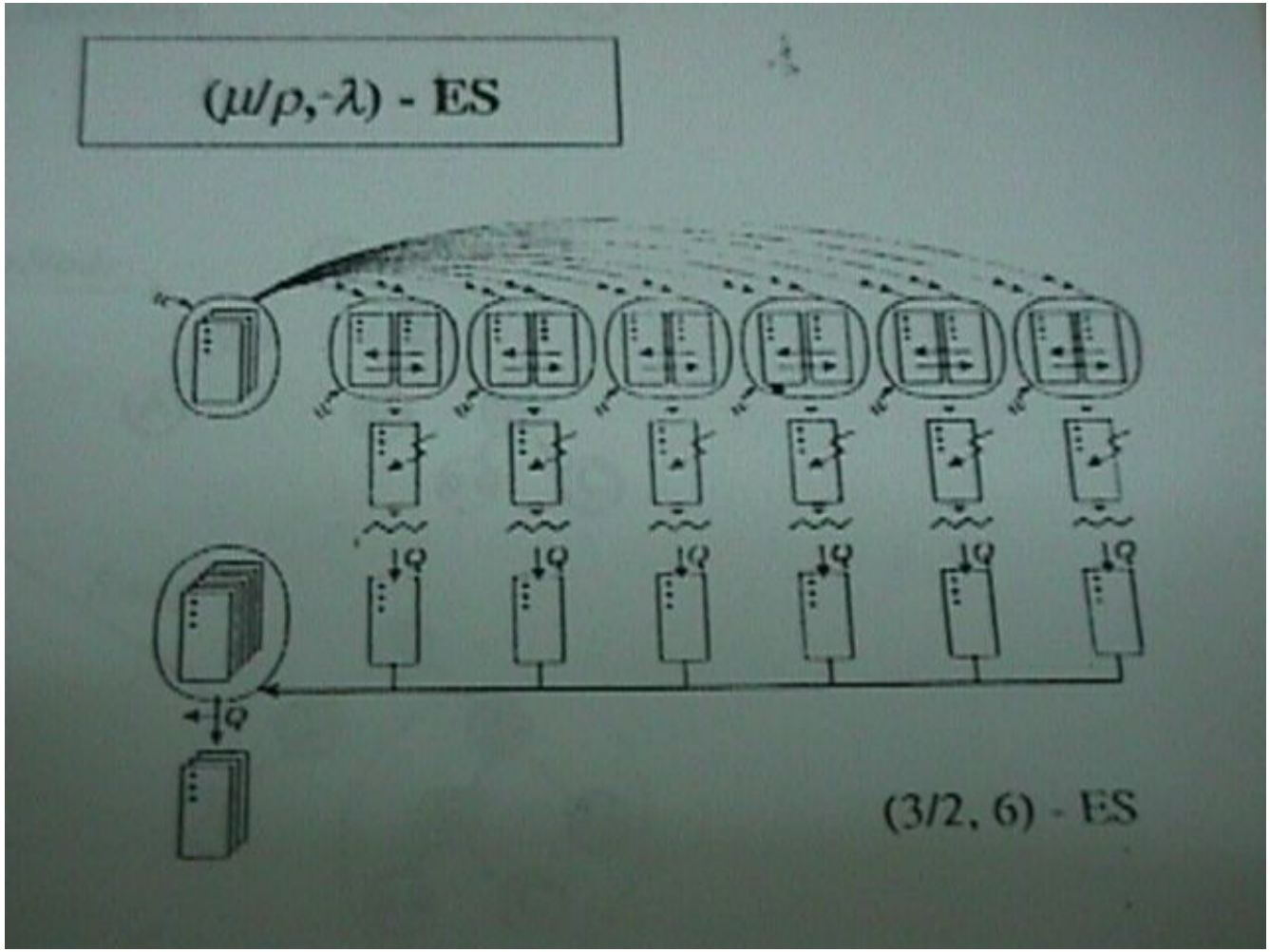
$\mathrm{sqrt}(S1^2+S'1^2), \ldots, \mathrm{sqrt}(S1^2+S'1^2)\rangle$

ESTRATEGIAS EVOLUTIVAS
Anidamiento

$[u'+, \lambda'(u+, \lambda)^{y}]^{y'}ES$

Hay *u'* poblaciones de *u* padres. Ellas son usadas para generar λ' poblaciones iniciales de *u* individuos. Por cada λ' población un $(u+, \lambda)^{y}ES$ es ejecutado durante *y* generaciones. Este esquema es respetido *y'* veces

ESTRATEGIAS EVOLUTIVAS



ESTRATEGIAS EVOLUTIVAS

- **RESOLUCIÓN DEL PROBLEMA DL CUBO.**

  - OBJETIVO: TODOS LOS LADOS CON EL MISMO COLOR

  - FUNCION: Fext= Fbas + Fborde + Fesquina

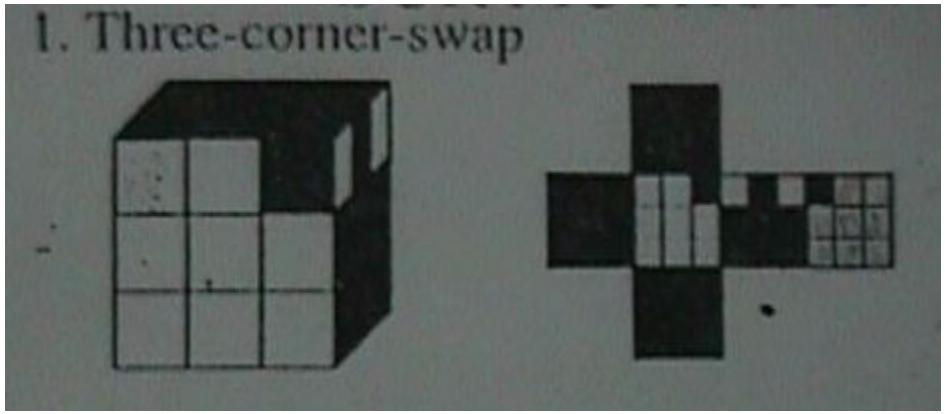
  Fbas: COLOR ERRADO DE LADO => +1

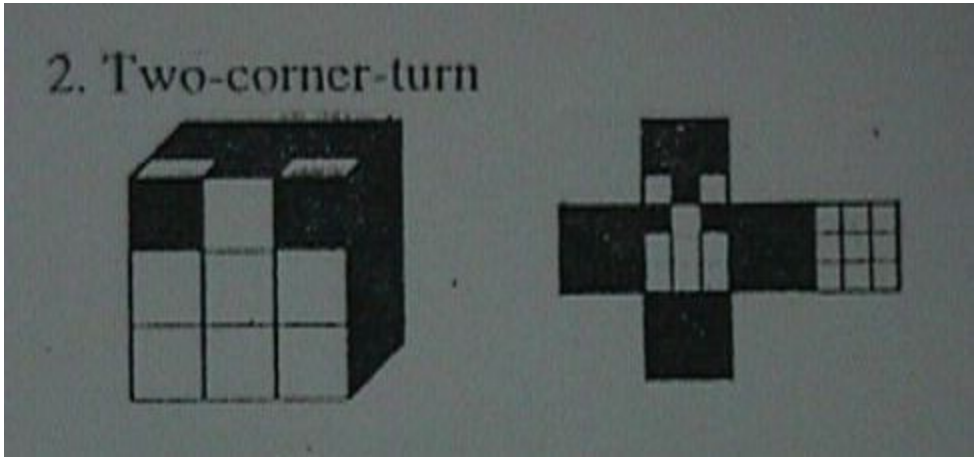
  Fesquina: COLOR ESQUINA ERRADO =>+6

  Fborde: COLOR BORDE ERRADO=>+4

ESTRATEGIAS EVOLUTIVAS

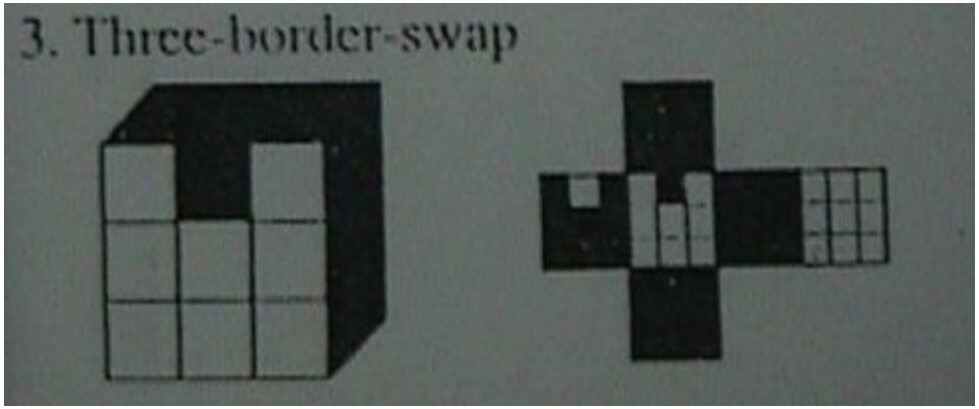
- **OPERADOR MUTACIÓN:**

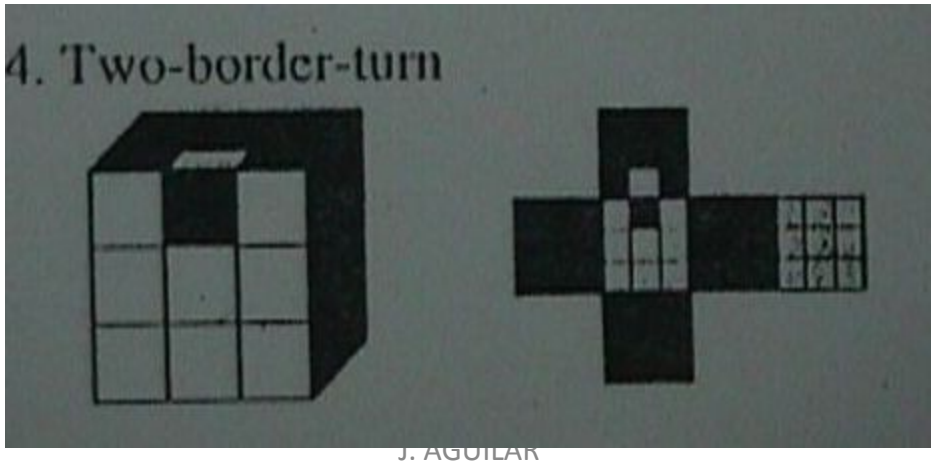


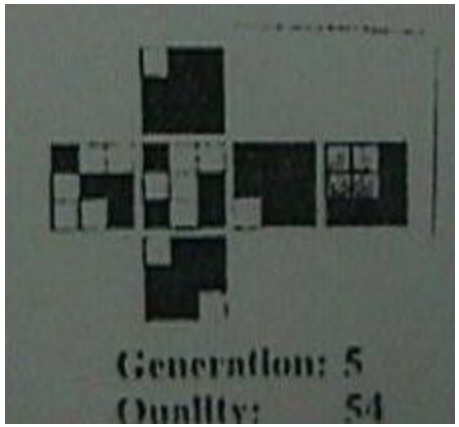


ESTRATEGIAS EVOLUTIVAS

- **OPERADOR MUTACIÓN:**





ESTRATEGIAS EVOLUTIVAS

- **EVOLUCIÓN:**