

# Introducción a la Computación Evolutiva

**Dr. Carlos A. Coello Coello**

**Departamento de Computación**

**CINVESTAV-IPN**

**Av. IPN No. 2508**

**Col. San Pedro Zacatenco**

**México, D.F. 07300**

**email: [ccoello@cs.cinvestav.mx](mailto:ccoello@cs.cinvestav.mx)**

**[http: //delta.cs.cinvestav.mx/~ccoello](http://delta.cs.cinvestav.mx/~ccoello)**

## Mutación No Uniforme

Propuesta por Michalewicz (1992).

Dado:  $P = \langle V_1, \dots, V_m \rangle$

el individuo mutado será:  $P' = \langle V_1, \dots, V'_k, \dots, V_m \rangle$

donde:

$$V'_k = \begin{cases} V_k + \Delta(t, UB - V_k) & \text{si } R = \text{Cierto} \\ V_k - \Delta(t, V_k - LB) & \text{si } R = \text{Falso} \end{cases} \quad (1)$$

y la variable  $V_k$  está en el rango  $[LB, UB]$

## Mutación No Uniforme

$$R = \text{flip}(0.5)$$

$\Delta(t, y)$  regresa un valor en el rango  $[0, y]$  tal que la probabilidad de que  $\Delta(t, y)$  esté cerca de cero se incrementa conforme  $t$  (generación actual) crece. Esto hace que este operador explore de manera más global el espacio de búsqueda al inicio (cuando  $t$  es pequeña) y de manera más local en etapas posteriores.

## Mutación No Uniforme

Michalewicz sugiere usar:

$$\Delta(t, y) = y \cdot (1 - r^{(1 - \frac{t}{T})^b})$$

donde:

$r$  es un número aleatorio real entre 0 y 1,  $T$  es el número máximo de generaciones y  $b$  es un parámetro que define el grado de no uniformidad de la mutación (Michalewicz sugiere usar  $b = 5$ ).

## Mutación No Uniforme

Ejemplo:  $P = \langle 2.3, 4.5, -1.2, 0.8 \rangle$

$$V_k = 4.5, \quad l_k = -2.0, \quad u_k = 6.5, \quad T = 50, \quad t = 5,$$
$$R = \text{Falso}, \quad r = 0.24, \quad b = 5.$$

$$V'_k = V_k - \Delta(t, V_k - l_K) = 4.5 - \Delta(5, 4.5 + 2) = 4.5 - \Delta(5, 6.5)$$

$$\Delta(5, 6.5) = 6.5(1 - 0.24)^{(1 - \frac{5}{50})^5} = 6.489435$$

$$V'_k = 4.5 - 6.489435 = -1.989435$$

## Mutación de Límite

Dado:  $P = \langle V_1, \dots, V_m \rangle$

el individuo mutado será:  $P' = \langle V_1, \dots, V'_k, \dots, V_m \rangle$

donde:

$$V'_k = \begin{cases} LB & \text{si } \textit{flip}(0.5) = \text{TRUE} \\ UB & \text{de lo contrario} \end{cases} \quad (2)$$

y  $[LB, UB]$  definen los rangos mínimos y máximos permisibles de valores para la variable  $V'_k$ .

## Mutación de Límite

Ejemplo:

$$P = \langle 1.5, 2.6, -0.5, 3.8 \rangle$$

$$V'_k = -0.5, \quad LB = -3.0, \quad UB = 1.3$$

Supongamos que:  $\text{flip}(0.5) = \text{TRUE}$

$$V'_k = -3.0$$

## Mutación Uniforme

Dado:

$$P = \langle V_1, \dots, V_m \rangle$$

el individuo mutado será:

$$P' = \langle V_1, \dots, V'_k, \dots, V_m \rangle$$

donde:

$$V'_k = rnd(LB, UB)$$

se usa una distribución uniforme y  $[LB, UB]$  definen los rangos mínimos y máximos de la variable  $V'_k$ .

## Mutación Uniforme

Ejemplo:

$$P = \langle 5.3, -1.3, 7.8, 9.1 \rangle$$

$$V_k = 5.3, \quad \text{LB} = 0.0, \quad \text{UB} = 10.5$$

$$V'_k = \text{rnd}(0.0, 10.5) \quad V'_k = 4.3$$

## Parameter-Based Mutation

Utilizada en conjunción con SBX. Fue propuesta por Deb (1995,1997). El procedimiento es el siguiente:

- 1) Crear un número aleatorio  $u$  entre 0 y 1
- 2) Calcular:

$$\vec{\delta} = \begin{cases} (2u)^{\frac{1}{\eta_m+1}} - 1 & \text{si } u < 0.5 \\ 1 - [2(1-u)]^{\frac{1}{\eta_m+1}} & \text{de lo contrario} \end{cases} \quad (3)$$

Donde  $\eta_m$  es el índice de distribución para la mutación y toma cualquier valor no negativo. Deb sugiere usar:

$$\eta_m = 100 + t \quad (t = \text{generación actual})$$

## Parameter-Based Mutation

3) El valor de la posición mutada se determina usando:

$$V'_k = V_k + \delta \vec{\Delta}_{max}$$

donde  $\Delta_{max}$  es la máxima perturbación permitida. Si se conoce el rango de la variable  $V_K$ , suele usarse:

$$\Delta_{max} = UB - LB$$

considerando que:

$$V_k \in [LB, UB]$$

## Parameter-Based Mutation

Ejemplo:

$$P = \langle 2.3, 4.5, -1.2, 0.8 \rangle$$

$$V_k = -1.2, \quad u = 0.72, \quad t = 20$$

$$LB = -2.0, \quad UB = 6.0$$

$$n_m = 100 + t = 120$$

$$\vec{\delta} = 1 - [ 2 ( 1 - 0.72 ) ] \frac{1}{n_m + 1}$$

## Parameter-Based Mutation

$$\vec{\delta} = 0.00478043$$

$$\Delta_{max} = UB - LB = 6.0 + 2.0 = 8.0$$

$$V'_k = -1.2 + 0.00478043(8.0) = -1.1617566$$

## Cruza vs. Mutación

- La cruce uniforme es más “explorativa” que la cruce de un punto.

Por ejemplo, dados:

$$P1 = 1 * * * * 1$$

$$P2 = 0 * * * * 0$$

La cruce uniforme producirá individuos del esquema \* \* \* \* \*, mientras que la cruce de un punto producirá individuos de los esquemas 1 \* \* \* \* 0 y 0 \* \* \* \* 1.

## Cruza vs. Mutación

¿Cuál es el poder exploratorio de la mutación?

- Si el porcentaje de mutación es cero, no hay alteración alguna.
- Si es uno, la mutación crea siempre complementos del individuo original.

## Cruza vs. Mutación

- Si es 0.5, hay una alta probabilidad de alterar fuertemente el esquema de un individuo.

En otras palabras, podemos controlar el poder de alteración de la mutación y su capacidad de exploración puede hacerse equivalente a la de la cruza.

## Cruza vs. Mutación

- El tipo de exploración efectuada por la mutación es, sin embargo, diferente a la de la cruza.

Por ejemplo, dados:

$$P1 = 10 * * * *$$

$$P2 = 11 * * * *$$

La cruza producirá sólo individuos del esquema 1 \* \* \* \*\*.

## Cruza vs. Mutación

El primer “1” en el esquema está garantizado (sin importar qué tipo de cruce se use), porque es común en los esquemas de ambos padres. La mutación, sin embargo, no respetará necesariamente este valor.

## Cruza vs. Mutación

- La craza “preserva” los alelos que son comunes en los 2 padres. Esta preservación limita el tipo de exploración que la craza puede realizar. Esta limitación se agudiza conforme la población pierde diversidad, puesto que el número de alelos comunes se incrementará.

## Cruza vs. Mutación

- Cuando buscamos localizar el óptimo global de un problema, la mutación puede ser más útil que la cruza. Si lo que nos interesa es ganancia acumulada (el objetivo original del AG), la cruza es entonces preferible.
- La cruza parece trabajar bien con funciones que están altamente correlacionadas o tienen epístasis moderada.

## Ajuste de Parámetros de un AG

Los parámetros principales de un AG son:

- Tamaño de población
- Porcentaje de cruza
- Porcentaje de mutación

Estos parámetros normalmente interactúan entre sí de forma no lineal, por lo que no pueden optimizarse de manera independiente.

## Ajuste de Parámetros de un AG

De Jong [1975] efectuó una serie de experimentos para comparar AGs con técnicas de gradiente. En su estudio, De Jong propuso cinco funciones de prueba que exhibían una serie de características que las hacían difíciles para las técnicas de gradiente.

## Ajuste de Parámetros de un AG

Sin embargo, antes de proceder a realizar sus experimentos, De Jong decidió analizar la influencia de los parámetros de un AG en su desempeño.

## Ajuste de Parámetros de un AG

Para medir el impacto de los parámetros de un AG, De Jong propuso dos métricas:

1. **Desempeño en Línea (Online)**: Es la aptitud promedio de todos los individuos que han sido evaluados en las últimas  $t$  generaciones.
2. **Desempeño fuera de Línea (Offline)**: Es el promedio de las mejores aptitudes evaluadas en las últimas  $t$  generaciones.

## Ajuste de Parámetros de un AG

Para que un algoritmo de búsqueda tenga un buen desempeño **en línea**, debe decidir rápidamente dónde están las regiones más prometedoras de búsqueda y concentrar ahí sus esfuerzos.

El desempeño **fuera de línea** no penaliza al algoritmo de búsqueda por explorar regiones pobres del espacio de búsqueda, siempre y cuando ello contribuya a alcanzar las mejores soluciones posibles (en términos de aptitud).

## Ajuste de Parámetros de un AG

Los parámetros hallados por De Jong que proporcionaron el mejor desempeño tanto en línea como fuera de línea son:

Tamaño de la población	50 a 100 individuos
Porcentaje de cruza	0.60
Porcentaje de mutación	0.001

## Ajuste de Parámetros de un AG

Algunas conclusiones interesantes de De Jong [1975] fueron:

- Incrementar el tamaño de la población reduce los efectos estocásticos del muestreo aleatorio en una población finita, por lo que mejora el desempeño del algoritmo a largo plazo, aunque esto es a costa de una respuesta inicial más lenta.

## Ajuste de Parámetros de un AG

- Incrementar el porcentaje de mutación mejora el desempeño **fuera de línea** a costa de sacrificar el desempeño **en línea**.
- Reducir el porcentaje de cruza mejora la media de desempeño, lo que sugiere que producir una generación de individuos completamente nuevos no es bueno.

## Ajuste de Parámetros de un AG

- Observando el desempeño de diferentes operadores de cruza, De Jong concluyó que, aunque el incrementar el número de puntos de cruza afecta su disrupción de esquemas desde una perspectiva teórica, esto no parece tener un impacto significativo en la práctica.

## Ajuste de Parámetros de un AG

Grefenstette [1986] usó un AG para optimizar los parámetros de otro (un meta-AG).

El meta-AG fue usado para evolucionar unos 50 conjuntos de parámetros de un AG que se usó para resolver las funciones de De Jong.

## Ajuste de Parámetros de un AG

Cada individuo codificaba seis parámetros:

1. Tamaño de la población
2. Porcentaje de cruza
3. Porcentaje de mutación
4. Intervalo generacional (porcentaje de la población que se reemplaza en cada generación)
5. Ventana de escalamiento (sin escalamiento, escalamiento basado en  $f(x)$  mínima de la primera generación, escalamiento basado en la  $f(x)$  mínima de las últimas  $W$  generaciones).
6. Estrategia de selección (elitista o puramente seleccionista).

La aptitud de un individuo era una función de su desempeño en línea y fuera de línea.

## Ajuste de Parámetros de un AG

El meta-AG usaba los parámetros de De Jong, y con él, Grefenstette [1986] obtuvo los siguientes valores óptimos de los parámetros para el desempeño **en línea**:

Tamaño de la población:	30 individuos
Porcentaje de cruza:	0.95
Porcentaje de mutación:	0.01
Selección:	Elitista
Intervalo generacional:	1.0 (100 %)
Ventana de escalamiento:	1 (basado) en la $f(x)$ mínima de la primera generación)

## Ajuste de Parámetros de un AG

Estos parámetros mejoraron ligera, pero no significativamente el desempeño **en línea** del AG con respecto a los de De Jong. Sin embargo, Grefenstette no pudo mejorar el desempeño **fuera de línea**.

## Ajuste de Parámetros de un AG

Algunas observaciones realizadas por Grefenstette [1986] fueron las siguientes:

- Los porcentajes de mutación por encima de 0.05 tienden a ser perjudiciales con respecto al desempeño **en línea**, y el AG aproxima el comportamiento de la búsqueda aleatoria para porcentajes de mutación  $\geq 0.1$ , sin importar qué otros parámetros se usen.

## Ajuste de Parámetros de un AG

- La ausencia de mutación está también asociada con un desempeño pobre del AG, lo que sugiere que su papel es más importante de lo que normalmente se creía en aquel entonces, pues permite refrescar valores perdidos del espacio de búsqueda.

## Ajuste de Parámetros de un AG

- El tamaño óptimo de la población para lograr el mejor desempeño posible **fuera de línea** está entre 60 y 110 individuos. Un alto intervalo generacional y el uso de una estrategia elitista también mejoran el desempeño del AG.

## Ajuste de Parámetros de un AG

- Para poblaciones pequeñas (20 a 40 individuos), el buen desempeño **en línea** está asociado con un porcentaje alto de cruza combinado con un porcentaje bajo de mutación o viceversa (un porcentaje bajo de cruza combinado con un porcentaje alto de mutación).

## Ajuste de Parámetros de un AG

- Para poblaciones de tamaño medio (30 a 90 individuos), el porcentaje óptimo de cruce parece decrementarse conforme se aumenta el tamaño de la población.

## Ajuste de Parámetros de un AG

Goldberg [1985] realizó un estudio teórico del tamaño ideal de la población de un algoritmo genético en función del número esperado de nuevos esquemas por miembro de la población. Usando una población inicial generada aleatoriamente con igual probabilidad para el cero y el uno, Goldberg derivó la siguiente expresión:

$$\text{Tam\_Poblacion} = 1,65^{2^{0,21L}} \quad (4)$$

donde:  $L$  = longitud de la cadena (binaria).

## Ajuste de Parámetros de un AG

Esta expresión sugiere tamaños de población demasiado grandes para cadenas de longitud moderada.

Ejemplos:

$$L = 30, \text{ Tam\_Población} = 130$$

$$L = 40, \text{ Tam\_Población} = 557$$

$$L = 50, \text{ Tam\_Población} = 2389$$

$$L = 60, \text{ Tam\_Población} = 10244$$

## Ajuste de Parámetros de un AG

Han habido innumerables ataques al trabajo de Goldberg antes mencionado, ya que éste se basó en una interpretación errónea del teorema de los esquemas. Para entender la falacia del argumento de Goldberg, debemos comenzar por definir un concepto muy importante de computación evolutiva: el **paralelismo implícito**.

## Ajuste de Parámetros de un AG

El **paralelismo implícito** se define así:

Mientras un AG calcula explícitamente las aptitudes de los  $N$  miembros de una población, al mismo tiempo estima implícitamente las aptitudes promedio de una cantidad mucho mayor de esquemas, calculando implícitamente las aptitudes promedio observadas de los esquemas que tienen instancias en la población.

## Ajuste de Parámetros de un AG

Según el teorema de los esquemas (que veremos más adelante), un AG procesa  $O(N^3)$  esquemas. A partir de esta idea, Goldberg concluye entonces que a mayor valor de  $N$  (tamaño de la población), mejor desempeño tendrá el AG, y de ahí deriva su expresión para calcular el tamaño óptimo de una población.

## Ajuste de Parámetros de un AG

El problema de este argumento es que sólo hay  $3^L$  esquemas en una representación binaria, por lo que no se pueden procesar  $O(N^3)$  esquemas si  $N^3$  es mucho mayor que  $3^L$ .

## Ajuste de Parámetros de un AG

Robertson [1986] determinó que en los AGs paralelos, el desempeño se incrementaba monotónicamente con el tamaño de la población sin seguir la expresión exponencial de Goldberg.

## Ajuste de Parámetros de un AG

Otros investigadores han derivado expresiones según las cuales, un incremento lineal del tamaño de la población corresponde con un buen desempeño del AG.

La regla empírica más común es usar una población de al menos 2 veces  $L$ .

## Tamaño Óptimo de la Población

Algunas observaciones de Goldberg [1989] son las siguientes:

- Cuando puede demostrarse convergencia de un AG, ésta parece no ser peor que una función cuadrática o cúbica del número de bloques constructores del problema, independientemente del tipo de esquema de solución utilizado.
- La teoría sugiere que el tamaño óptimo de la población es  $N = 3$ , sin importar la longitud de la cadena cromosómica. Esta observación dio pie al micro-AG (Krishnakumar [1989]).

## Micro-Algoritmos Genéticos

El funcionamiento de un micro-AG es el siguiente:

1. Generar al azar una población muy pequeña.
2. Aplicar los operadores genéticos hasta lograr convergencia nominal (por ejemplo, todas las cadenas son iguales).
3. Generar una nueva población transfiriendo los mejores individuos de la población anterior a la nueva, y generando al azar los individuos restantes.
4. Continuar hasta llegar al óptimo.

## Los Experimentos de David Schaffer



Schaffer et al. [1989] efectuaron una serie de experimentos que consumieron aproximadamente 1.5 años de tiempo de CPU (en una Sun 3 y una VAX), en los cuales intentaron encontrar los parámetros óptimos de un AG con codificación de Gray y usando muestreo estocástico universal.

## Los Experimentos de David Schaffer

Los parámetros sugeridos por estos experimentos (para el desempeño “en línea”) fueron:

Tamaño de población:	20-30 individuos
Porcentaje de cruce (2 puntos):	0.75-0.95
Porcentaje de mutación:	0.005-0.01

## Los Experimentos de David Schaffer

Algunas de las observaciones de Schaffer et al. [1989] fueron:

- El uso de tamaños grandes de población ( $> 200$ ) con porcentajes altos de mutación ( $> 0,05$ ) no mejora el desempeño de un AG.
- El uso de poblaciones pequeñas ( $< 20$ ) con porcentajes bajos de mutación ( $< 0,002$ ) no mejora el desempeño de un AG.
- La mutación parece tener mayor importancia de lo que se cree en el desempeño de un AG.

## Los Experimentos de David Schaffer

- El AG resultó relativamente insensible al porcentaje de cruce. Un NE (*naive evolution*), o sea, un AG sin cruce, funciona como un *hill climber*, el cual puede resultar más poderoso de lo que se cree.
- Los operadores genéticos pueden muestrear eficientemente el espacio de búsqueda sin necesidad de usar tamaños de población excesivamente grandes.
- La cruce de 2 puntos es mejor que la de un punto, pero sólo marginalmente.
- Conforme se incrementa el tamaño de la población, el efecto de la cruce parece diluirse.

## Auto-Adaptación

En general, es poco probable poder determinar “a priori” un conjunto óptimo de parámetros para un AG cualquiera aplicado a un problema en particular. Algunos investigadores creen que la mejor opción es la **auto-adaptación**, o sea permitir que un algoritmo genético adapte por sí mismo sus parámetros.

## Adaptación en Línea

Srinivas y Patnaik [1994] propusieron un esquema para adaptar las probabilidades de cruce y mutación de un algoritmo genético. La propuesta se basa en la detección de que el algoritmo genético ha convergido. Para ello, verifican qué diferencia existe entre la aptitud máxima de la población y la aptitud promedio. De tal forma, se hacen variar los porcentajes de cruce y mutación en función de esta diferencia de valores (aptitud máxima y aptitud promedio de la población).

## Adaptación en Línea

Las expresiones propuestas por Srinivas y Patnaik [1994] son:

$$p_c = k_1 / (f_{max} - \bar{f})$$

$$p_m = k_2 / (f_{max} - \bar{f})$$

Sin embargo, con estas expresiones los porcentajes de cruce y mutación se incrementan conforme el algoritmo genético converge y produce un comportamiento altamente disruptivo en la vecindad del óptimo, de manera que el algoritmo puede no converger jamás.

## Adaptación en Línea

Para evitar este problema, estas expresiones deben modificarse de manera que se preserven las “buenas” soluciones.

La propuesta es ahora la siguiente:

$$p_c = k_1(f_{max} - f') / (f_{max} - \bar{f}), k_1 \leq 1,0$$

$$p_m = k_2(f_{max} - f) / (f_{max} - \bar{f}), k_2 \leq 1,0$$

donde  $k_1$  y  $k_2$  deben ser menores que 1.0 para hacer que los valores de  $p_c$  y  $p_m$  estén en el rango de 0.0 a 1.0. En estas fórmulas,  $f_{max}$  es la aptitud máxima de la población,  $f'$  es la aptitud más grande de los padres a recombinarse y  $f$  es la aptitud del individuo a mutarse. Así mismo,  $\bar{f}$  es la aptitud promedio de la población.

## Adaptación en Línea

Estas expresiones hacen que el porcentaje de cruce ( $p_c$ ) y de mutación ( $p_m$ ) disminuya cuando los individuos tienen una aptitud alta y que aumenten en caso contrario. Nótese sin embargo que  $p_c$  y  $p_m$  se harán cero al alcanzarse la aptitud máxima. También adviértase que  $p_c = k_1$  si  $f' = \bar{f}$  y  $p_m = k_2$  si  $f = \bar{f}$ . Para evitar valores mayores a 1.0 para  $p_c$  y  $p_m$ , se imponen las restricciones siguientes:

$$p_c = k_3, f' \leq \bar{f}$$

$$p_m = k_4, f \leq \bar{f}$$

donde  $k_3, k_4 \leq 1,0$ .

## Adaptación en Línea

Debido a que  $p_c$  y  $p_m$  se harán cero cuando el individuo sea el mejor en la población, su propagación puede llegar a ser exponencial, produciendo convergencia prematura. Para evitar eso, los autores proponen usar un porcentaje de mutación por omisión (0.005) en estos casos.

## Adaptación en Línea

Las expresiones finales son:

$$p_c = k_1(f_{max} - f') / (f_{max} - \bar{f}), f' \geq \bar{f}$$

$$p_c = k_3, f' < \bar{f}$$

$$p_m = k_2(f_{max} - f) / (f_{max} - \bar{f}), f \geq \bar{f}$$

$$p_m = k_4, f < \bar{f}$$

donde:  $k_1, k_2, k_3$  y  $k_4 \leq 1,0$ . Los autores sugieren:

$$k_2 = k_4 = 0,5, \quad k_1 = k_3 = 1,0$$

## Adaptación en Línea

Con estos valores, se usan soluciones con una aptitud inferior al promedio para buscar la región donde reside el óptimo global. Un valor de  $k_4 = 0,5$  hará que estas soluciones sean totalmente disruptivas. Lo mismo hará  $k_2 = 0,5$  con las soluciones cuya aptitud iguale el promedio de la población.

## Adaptación en Línea

Asignar  $k_1 = k_3 = 1,0$  hará que todas las soluciones cuya aptitud sea menor o igual a  $\bar{f}$  se sometan compulsivamente a cruza.

La probabilidad de cruza decrece conforme la aptitud del mejor de los padres a recombinarse tiende a  $f_{max}$  y se vuelve cero para los individuos con una aptitud igual a  $f_{max}$ .

## Auto-adaptación de la probabilidad de mutación

En este caso, el porcentaje de mutación se agrega como un parámetro más al genotipo, de tal forma que se vuelva una variable más tal que su valor oscile entre 0.0 y 1.0.

Bäck y Schütz [1996] proponen usar:

$$p'_m = \frac{1}{1 + \frac{1-p_m}{p_m} e^{-\gamma N(0,1)}}$$

## Auto-adaptación de la probabilidad de mutación

donde:

$p_m$  = porcentaje actual de mutación,  $p'_m$  = nuevo porcentaje de mutación.

$\gamma$  = tasa de aprendizaje (se sugiere  $\gamma = 0,2$ ).

## Auto-adaptación de la probabilidad de mutación

$N(0, 1)$  indica una variable aleatoria con una distribución normal tal que su esperanza es cero y su desviación estándar es uno.

Aplicando este operador, pasamos del genotipo:

$$c = (x, p_m)$$

al nuevo genotipo:

$$(x', p'_m)$$

## Auto-adaptación de la probabilidad de mutación

La mutación de la variable  $x$  está dada por:

$$x' = \begin{cases} x_j & \text{si } q \geq p'_m \\ 1 - x_j & \text{si } q < p'_m \end{cases}$$

donde:  $q$  es un valor aleatorio (distribución uniforme) muestreado de nuevo para cada posición  $j$ .

## Auto-adaptación de la probabilidad de mutación

Este esquema puede generalizarse incluyendo un vector  $p$  de porcentajes de mutación asociados a cada variable:

$$p = (p_1, \dots, p_L)$$

El genotipo  $c$  tiene la forma:

$$c = (x, p)$$

## Auto-adaptación de la probabilidad de mutación

Los porcentajes de mutación se actualizan usando:

$$p'_j = \frac{1}{1 + \frac{1-p_j}{p_j} e^{-\gamma N(0,1)}}, j = 1, \dots, L.$$

## La propuesta de Davis

Davis [1989,1991] realizó un estudio muy interesante sobre un mecanismo de auto-adaptación aplicado a algoritmos genéticos. En su estudio, Davis asignó a cada operador genético una “aptitud”, la cual era función de cuántos individuos con aptitud elevada habían contribuido a crear dicho operador en un cierto número de generaciones. Un operador era recompensado por crear buenos individuos directamente, o por “dejar la mesa puesta” para ello (es decir, por crear ancestros para los buenos individuos).

## La propuesta de Davis

La técnica de Davis se usó en un AG de estado uniforme. Cada operador (cruza y mutación) empezaba con la misma aptitud y cada uno de estos operadores se seleccionaba con una probabilidad basada en su aptitud para crear un nuevo individuo, el cual reemplazaba al individuo menos apto de la población. Cada individuo llevaba un registro de quién lo creó. Si un individuo tenía una aptitud mayor que la mejor aptitud actual, entonces el individuo recibía una recompensa para el operador que lo creó y ésta se propagaba a su padre, su abuelo, y tantos ancestros como se deseara.

## La propuesta de Davis

La aptitud de cada operador sobre un cierto intervalo de tiempo estaba en función de su aptitud previa y de la suma de recompensas recibidas por todos los individuos que ese operador hubiese ayudado a crear en ese tiempo.

## La propuesta de Davis

Para implementar la auto-adaptación, suelen codificarse los porcentajes de cruce y mutación (y a veces incluso el tamaño de la población) como variables adicionales del problema. Los valores de los parámetros del AG se evolucionan de acuerdo a su efecto en el desempeño del algoritmo.

## Críticas a la Auto-Adaptación

La auto-adaptación no ha sido tan exitosa en los AGs, como lo es en otras técnicas evolutivas (p.ej., las estrategias evolutivas) ¿Por qué?

El problema fundamental es que nadie ha podido contestar satisfactoriamente la siguiente pregunta [Mitchell, 1996]:

¿qué tan bien corresponde la velocidad de adaptación de la población con la adaptación de sus parámetros?

## Críticas a la Auto-Adaptación

Dado que la información necesaria para auto-adaptar los parámetros proviene de la población misma, esta información podría no poder viajar suficientemente rápido como para reflejar con fidelidad el estado actual de la población. De ahí que el uso de auto-adaptación en un AG siga siendo objeto de controversia.

## Mecanismos de Adaptación

### Mutaciones Variables

Varios investigadores han abordado el problema del ajuste del porcentaje de mutación de un algoritmo genético. La idea de usar porcentajes de mutación dependientes del tiempo fue sugerida por Holland [1975], aunque no proporcionó una expresión en particular que describiera la variabilidad requerida. Fogarty [1989] usó varias expresiones para variar  $p_m$  en las que se incluye el tiempo, logrando mejoras notables de desempeño. En ambos casos, la propuesta fue decrementar de manera determinística los porcentajes de mutación, de manera que tiendan a cero.

## Mecanismos de Adaptación

Otra propuesta es la de Hesser y Männer [1991], en la cual se usa:

$$p_m(t) = \sqrt{\frac{\alpha}{\beta}} \frac{e^{-\gamma t/2}}{\lambda \sqrt{l}}$$

donde:  $\lambda$  = tamaño de la población,  $l$  = longitud cromosómica,  $t$  = generación actual,  $\alpha, \beta, \gamma$  son constantes definidas por el usuario (dependientes del problema).

Nótese que en todas estas propuestas se usa el mismo porcentaje de mutación para todos los individuos de la población en la generación  $t$ .

## Mecanismos de Adaptación

Bäck y Schütz [1996] propusieron un porcentaje de mutación que se decrementa usando:

$$p_m(t) = \frac{L}{2 + \frac{L-2}{T}t}$$

donde:  $0 \leq t \leq T$ ,  $L$  = longitud cromosómica,  $t$  = generación actual y  $T$  es el número máximo de generaciones.

## Mecanismos de Adaptación

La variabilidad es:

$$p_m(0) = 0,5$$

$$p_m(T) = \frac{1}{L}$$

## Mutación dependiente de la aptitud

Bäck [1992] sugirió el uso de un porcentaje de mutación que fuera función de la aptitud de cada individuo:

$$p_m(x) = \frac{1}{2(f(x) + 1) - L}$$

## AGs adaptativos

Los objetivos principales de los AGs adaptativos son los siguientes:

- Mantener diversidad en la población.
- Mejorar la convergencia del AG, evitando a la vez la convergencia prematura.
- Evitar la disrupción de esquemas ocasionada por la cruza.

## AGs adaptativos

De acuerdo a como lo plantean Herrera y Lozano [1996], un AGA incluye:

- Ajuste adaptativo de parámetros (probabilidad de cruce y mutación, longitud del genotipo y tamaño de la población).
- Función de aptitud adaptativa.
- Operador de selección adaptativo.
- Operadores de búsqueda (variación) adaptativos.
- Representación adaptativa.

## AGs adaptativos

El mecanismo de adaptación puede estar completamente separado del mecanismo de búsqueda del AG. Este tipo de esquema “no acoplado” no es muy atractivo, porque implica un control centralizado, superimpuesto al mecanismo de búsqueda del AG.

Otra posibilidad es que el mecanismo de búsqueda del AG sea usado parcialmente por el mecanismo adaptativo. En este caso, se dice que el AG y el mecanismo adaptativo están “ligeramente acoplados” (*loosely coupled*).

## AGs adaptativos

Si la adaptación es conducida por las fuerzas internas de la búsqueda evolutiva, podemos hablar de un “acoplamiento fuerte”. En este caso, se origina un acoplamiento de los 2 espacios de búsqueda sobre los que opera el AG (el espacio de las soluciones y el de las variables de decisión).

## Técnicas Adaptativas Basadas en Lógica Difusa

Los controladores difusos suelen usarse con frecuencia como técnica adaptativa con los AGs [Herrera y Lozano, 1996].

La integración de AGs y controladores difusos suelen orientarse hacia los temas siguientes:

1. Elegir los parámetros del AG antes de efectuar las corridas.
2. Ajustar los parámetros en línea, adaptándose a nuevas situaciones.
3. Asistir al usuario en detectar las soluciones emergentes útiles, en monitorear el proceso evolutivo con la finalidad de evitar convergencia prematura, y en diseñar AGs para una cierta tarea en particular.

## Representaciones Adaptativas

Se han propuesto varios esquemas en los que lo que se adapta es la representación usada con un AG. A continuación veremos 2 propuestas muy interesante de este tipo.

### Sistema ARGOT

Propuesto por Schaefer [1987], el método ARGOT es un algoritmo de búsqueda diseñado de tal manera que puede “aprender” la estrategia de búsqueda más adecuada.

ARGOT consiste de un AG convencional al que se agregan varios operadores que modifican el mapeo intermedio que traduce los cromosomas en parámetros (o variables) de decisión.

## Representaciones Adaptativas

Estos operadores se controlan por medio de 3 tipos de medidas internas de la población:

- (a) Medidas de convergencia (p.ej., la uniformidad de los cromosomas en un cierto lugar en particular).
- (b) Medidas de posicionamiento (posición promedio relativa de las soluciones actuales con respecto a sus rangos extremos).
- (c) Medidas de varianza (p.ej., el “ancho” de la distribución de las soluciones con respecto a los rangos permisibles).

## Representaciones Adaptativas

Estas medidas se aplican sobre cada gene del cromosoma y se usan para activar un cierto operador cuando resulta adecuado.

Los operadores incluyen uno que altera la resolución de un gene (número de bits empleados para representar una variable) y otros que mueven (*shift*), expanden y contraen el mapeo intermedio entre cromosomas y variables de decisión. Estos cambios (expansión y contracción) hacen que los rangos de cada variable se modifiquen con la finalidad de focalizar la búsqueda y permiten también aplicar restricciones de desigualdad a las variables de decisión.

## Representaciones Adaptativas

Además de los operadores primarios antes mencionados, se usaron otros secundarios tales como un operador de mutación de Metropolis que acepta un cambio en un bit sólo si mejora la solución actual con la mutación. Si el cambio no mejora, se decide si se acepta o no el cambio usando una distribución de Boltzmann. También se propuso un operador de homotopía (búsqueda local) para evitar convergencia a un óptimo local.

## Representaciones Adaptativas

### Codificación Delta

La idea de esta propuesta [Matthias & Whitley 1994] es cambiar dinámicamente la representación de un problema. Nótese, sin embargo, que no intenta “aprender” cuál es la mejor representación del espacio de búsqueda, sino más bien se cambia la representación de manera periódica para evitar los sesgos asociados con una representación determinada del problema.

## Representaciones Adaptativas

El algoritmo de la codificación delta empieza con la ejecución inicial de un algoritmo genético usando cadenas binarias. Una vez que la diversidad de la población ha sido explotada adecuadamente, se almacena la mejor solución bajo el nombre de “solución temporal”. Se reinicializa entonces el AG con una nueva población aleatoria. En esta ocasión, sin embargo, las variables se decodifican de tal forma que representen una distancia o *valor delta* ( $\pm\delta$ ) con respecto a la solución temporal.

## Representaciones Adaptativas

El valor de  $\delta$  se combina con la solución temporal de manera que los parámetros resultantes se evalúen usando la misma función de aptitud.

De esta manera, la codificación delta produce un nuevo mapeo del espacio de búsqueda a cada iteración. Esto permite explorar otras representaciones del problema que podrían “facilitar” la búsqueda.

## Representaciones Adaptativas

**Ejemplo de codificación binaria usando códigos delta.**

Parámetros numéricos	0	1	2	3	4	5	6	7
Codificación binaria	000	001	010	011	100	101	110	111
Cambios numéricos	0	1	2	3	-3	-2	-1	-0
Codificación delta	000	001	010	011	111	110	101	100