# **1 Parte**

## Introducción a la computación evolutiva.

La computación evolutiva está compuesta por modelos de evolución basados en poblaciones cuyos elementos representan soluciones a problemas, simular esto resulte ser una técnica de optimización probabilística que normalmente mejora a otros métodos clásicos..

La evolución es un proceso que opera sobre los cromosomas, en lo conocido como selección natural. La metáfora de evolución con la resolución de problemas seria la equiparación de:

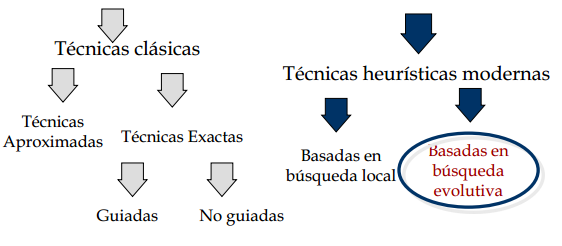
* Individuo: Solución candidata
* Adaptación: Calidad de la solución
* Entorno: Problema
* Reproducción: Selección de soluciones
* Mutación: Modificación de una solución candidata.
* Recombinación: Combinación de varias soluciones candidatas

En una búsqueda basada en trayectorias, podemos llegar fácilmente a un máximo local, en una basada en poblaciones, siempre se tiene información de la mejor solución lo que hace que sea más fácil salir de estos óptimos y encontrar mejores soluciones.

**Evolución Artificial.**

Existen cuatro paradigmas básicos en la evolución artificial:

* Algoritmos Genéticos.- Que utilizan operadores genéticos sobre cromosomas
* Estrategias de evolución.- Enfatizan cambios de comportamiento a nivel de los individuos
* Programación Evolutiva.- Enfatiza cambios de comportamiento a nivel de las especies
* Programación genética.- Que evoluciona expresiones representadas como árboles.



**CONTEXTUALIZACIÓN**

**Conclusiones**

Ventajas:

* Paralelismo intrínseco y superioridad con respecto a otras técnicas
* Buena actuación a un costo aceptable en amplia variedad de problemas
* Bajo coste de desarrollo, amplia escalabilidad
* Proporcionan un conjunto de soluciones que son interpretables.
* Fácil de hibridar con otras técnicas.

Desventajas:

* No garantizan la solución óptima y tienen una base teórica débil
* Muchos parámetros y computacionalmente lentos.

## Algoritmos Genéticos

Los algoritmos genéticos son algoritmos de optimización búsqueda y aprendizaje basados en los procesos de Evolución Natural y Evolución Genética. El ciclo evolutivo al igual que la estructura de un algoritmo genético se basa en:

* Población inicial (Padres) de la cual se seleccionan ciertos individuos.
* Se cruzan y mutan (en unos porcentajes variables), dando lugar a descendientes
* Los descendientes reemplazan a la población (de forma completa o parcial)
* Se itera el proceso.

Siendo este el proceso general, existen dos variantes:

**Modelo Generacional.**

Durante cada iteración, se crea una población completa con individuos nuevos que reemplaza la antigua. (La variante elitista mantiene el mejor de la población antigua)

**Modelo Estacionario.**

Durante cada iteración, se escogen dos padres de la población y se les aplican los operadores genéticos. Los descendientes reemplazan a los peores cromosomas de la población anterior, se trata obviamente de un modelo elitista, que tiene una convergencia rápida cuando se reemplazan los peores cromosomas de la población.

**Como se construye un algoritmo genético.**

Pasos dependientes del problema:

* Diseñar una representación: Hay que tener un mecanismo para codificar individuos como genotipos, representación binaria, real, de orden, etc.
* Como inicializar una población: Depende de la representación, por ejemplo si es binaria o real a base de números aleatorios entre intervalos, o también usando una población a partir de los resultados de una heurística previa.
* Diseñar una correspondencia entre genotipo y fenotipo: Es decir poder pasar de una representación interna del problema con datos a una solución.
* Diseñar una forma de evaluar un individuo: Depende del problema y suele ser costoso, como funciones de evaluación, simulador, subrutinas, etc.

Componentes del algoritmo:

* Diseñar operador de mutación adecuado: Pueden haber uno o varios:
  + Deben permitir alcanzar cualquier parte del espacio de búsqueda
  + Tamaño de mutación debe ser controlado
  + Debe producir cromosomas válidos.
  + Probabilidad muy baja de actuación.
* Diseñar operador de cruce adecuado: Se pueden tener uno o más operadores que deben tener ciertos aspectos en cuenta:
  + Los hijos deberían heredar características de cada padre, si no sería mutar.
  + La combinación debe dar cromosomas validos
* Decidir la selección de padres: Se debe garantizar que los mejores individuos tienen mayor posibilidad de ser padres, pero no obviando los menos buenos por que pueden tener material genético útil. Esto define la presión selectiva, que determina en qué grado la reproducción está dirigida por los mejores individuos.
  + Selección por torneo- K aleatorios que compiten.
  + Selección aleatoria, orden lineal, ruleta, etc.
* Decidir reemplazo de individuos de poblaciones: Esta parte cobra sentido cuando nos encontrarnos ante el modelo estacionario que no reemplaza toda la población, tenemos varias propuestas:
  + Reemplazar al peor de la población
  + Crowding determinístico. El hijo reemplaza a su padre más parecido, mantiene diversidad
  + Peor entre semejantes o torneo restringido, etc.

A parte de esto necesitamos una condición de parada.

* La obvia, cuando alcanzamos el óptimo (aunque igual no se sabe a priori)
* Máximo numero de evaluaciones o tiempo
* Después de x iteraciones sin mejora.

**Sobre su utilización:**

* No se deben sacar conclusiones con una sola ejecución, medias estadísticas o varias ejecuciones
* No se debe ajustar/chequear la actuación sobre ejemplos simples si va a trabajar sobre casos reales

## Algoritmos Genéticos. Exploración VS Explotación.

Todo algoritmo de búsqueda necesita un equilibrio entre estos dos factores, los algoritmos genéticos son un tipo de algoritmo de búsqueda cuyas componentes pueden establecer un equilibrio entre las dos componentes. Existen dos factores que influyen en la efectividad de un Algoritmo Genético:

* Convergencia.- Centrando la búsqueda en regiones prometedoras mediante la presión selectiva (los mejores se reproducen), procesos de competición entre padres e hijos
* Diversidad.- Si añadimos diversidad en la población podemos evitar la convergencia prematura. Algunas propuestas:
  + Diversidad con la mutación o con el cruce
  + Separación espacial, reemplazamiento de población.etc

**Algoritmo de evolución CHC.**

Una de las primeras propuestas de AG que introduce un equilibrio entre los dos factores, combina una selección elitista para conservar los mejores hasta el momento y un operador de cruce que produce hijos muy diferentes a los padres. Cuatro componentes novedosas:

CONVERGENCIA:

* Selección elitista: Selecciona los N mejores cromosomas entre padres e hijos. Los N mejores elementos encontrados hasta el momento permanecerán en la población

DIVERSIDAD:

* Cruce uniforme HUX: Intercambia la mitad de los alelos que son distintos en los padres, garantiza que los hijos estén “alejados” lo máximo de sus padres.
* Prevención de incesto: Se forman parejas y solo se cruzan aquellas cuyos miembros difieren lo suficiente (umbral de cruce, puede disminuir si no se cruza)
* Reinicialización: Si el umbral de cruce se reduce tanto como para ser inferior a 0, se reinicia la población de dos formas posibles:
  + Usando el mejor elemento como plantilla e introduciendo una copia
  + Manteniendo el mejor o parte de los mejores y el resto aleatorios

CHC no aplica operador de mutación

## Múltiples soluciones en problemas multimodales.

Muchos problemas tienen más de una solución localmente óptima, en la resolución del problema se desea obtener varios de estos.

* Se comienza con una población inicial con un muestreo aleatorio
* El proceso evolutivo suele provocar la convergencia de toda la población a una zona del espacio de búsqueda abandonando la exploración del resto de óptimos locales, conocido como deriva genética.

Con esto, las búsquedas se focalizan en un punto, perdiendo diversidad y convergiendo de forma prematura, haciendo que no se obtengan todos los óptimos.

El objetivo de tener múltiples soluciones en problemas multimodales es preservar la diversidad de la población, para conseguir esto se deben limitar las áreas de competencia en las que se actúa. La propuesta es utilizar algoritmos de nichos o multimodales, en la cual los AGs evolucionan una población que permita tener soluciones en diferentes zonas del espacio de búsqueda (nichos)

**Modelos de AGs Multimodales**

* Espaciales: Formación de diferentes nichos en las poblaciones de una misma ejecución del AG
* Temporales: Formación de diferentes nichos a lo largo de diferentes ejecuciones del AG.

El uso de AGs multimodales permite obtener múltiples soluciones en una única ejecución, manteniendo soluciones de diferentes partes del dominio.

El uso de técnicas de nichos es una alternativa importante para evitar la convergencia en óptimos locales y una herramienta importante para el desarrollo de algoritmos genéticos para problemas multiobjetivo.

## Programación genética.

La programación genética es un caso particular de AGs, usados para inducir programas de ordenador de un modo automático. La diferencia principal entre AGs y PG es la representación de las soluciones. Los cromosomas codifican de algún modo, programas de ordenador.

* Aplicación Típica: Regresión simbólica
* Características atribuidas:
  + Cromosomas no lineales: arboles, grafos
  + Cruce muy destructivo que genera mucha diversidad
  + Mutación posible pero no necesaria.

**Fundamentos de la programación genética:**

* Representación de programas: Existen diversas formas de representar programas
  + Arboles de expresiones (la más usada)
  + Secuencias de instrucciones de máquinas de estados finitos
  + Secuencias de instrucciones en código máquina.
* La representación de árbol y el manejo de gramáticas: La representación de programas en forma de árboles de expresión se basa en la existencia de una gramática libre de contexto que define las sentencias validas del lenguaje
  + Sin embargo al trabajar directamente con los árboles y las reglas necesarias para evaluarlos, no es necesario hacer referencia al lenguaje libre de contexto
  + El cromosoma como tal codifica la expresión del árbol utilizando cualquier recorrido sobre él
  + Para emplear esta representación es necesario definir e conjunto de símbolos terminales de la gramática
  + El conjunto de símbolos y funciones debe ser suficiente para resolver el problema.
  + Características principales:
    - AG: Representación lineal de estructuras
    - Los cromosomas en forma de árboles son estructuras no lineales
    - Los AG tienen cromosomas de tamaño fijo
    - Los árboles en PG pueden variar en profundidad y anchura.
* La función de adaptación: Debe especificar una medida de la calidad del programa codificado en la resolución del problema planteado. En principio para evaluar árboles se necesita una evaluación recursiva, subiendo desde las hojas asignando valores hasta la raíz.
* Generación de la población inicial: Se basa en una generación aleatoria de distintos programas que puede ser o no dirigida
  + El tamaño de los arboles ha de ser menor que un máximo permitido
  + Para introducir diversidad se puede:
    - Generar un porcentaje de árboles de cada tamaño valido
    - Generar programas puramente aleatorios o que hagan uso de todas las entradas
  + Para generar un árbol aleatorio se aplican progresiva y aleatoriamente las reglas de producción de la gramática.
* Operadores en programación genética: Existen dos bloques de operadores:
  + Operadores primarios:
    - Reproducción
    - Cruce
    - Mutación
  + Operadores secundarios
    - Permutación
    - Edición
    - Encapsulación
    - Decimación

# **2 Parte**

## Algoritmos evolutivos para la optimización continúa.

Los diseñadores de cada una de las técnicas de Computación Evolutiva vieron que sus problemas particulares podían ser resueltos por algoritmos de evolución.

Motivación: Centrarnos la atención en el problema de encontrar el óptimo global (u optimizar) de una función con variables reales que está caracterizada por:

* Múltiples mínimos
* No diferenciable.

## Estrategias de evolución.

Surgió un estado del arte con CMA-ES, una estrategia de evolución con covarianza de matriz adaptativa con una serie de características:

* Selección-mutación ES, se ejecta para n iteraciones
* Se registran los pasos exitosos
* Son analizados para encontrar direcciones básicas y puntos fuertes no correlacionados
* Requiere O(n3) para solucionar un problema de auto valores
* Rotación invariante.

## Evolución diferencial.

Modelo evolutivo que enfatiza la mutación, utiliza un operador de cruce/recombinación a posteriori de la mutación. Propuesto para optimización con parámetros reales.

* Inicialización: Una población de vectores de parámetros D-dimensionales, se genera aleatoriamente dentro de unos límites inferiores y superiores
* Generación de un vector de prueba: En la generación g-esima de una población, con N vectores D-dimensionales, a través de operadores de mutación y recombinación aplicados a la población actual
  + Mutación diferencial: con respecto a cada uno de los vectores en la población actual, llamado vector objetivo, se genera un vector mutado añadiendo un vector diferencia, escalado y aleatoriamente muestreado a un vector base aleatoriamente seleccionado de la población actual.
* Recombinación discreta: Con respecto a cada vector objetivo en la población actual, un nuevo vector se genera cruzando el vector objetivo con el correspondiente vector mutado con un ratio de cruce.
* Reemplazamiento: Si el vector generado tiene mejor valor que el vector objetivo, lo sustituye. Si no ocurre el vector objetivo inicial se mantiene.

Existen variantes de la evolución diferencial:

Mutación diferencial:

* Vector de una diferencia, de dos diferencias, factor de escalado de mutación G,etc.

Recombinación:

* Discreta en un punto, multipunto, exponencial, binomial, etc.
* Aritmética lineal, intermedia, intermedia extendida, etc.

Existen algunos trabajos clásicos en evolución diferencial como SaDE, que para alcanzar los mejores resultados en un problema propone que en vez de usar prueba y error, se propone ajustar las soluciones y los valores de los parámetros, SelfAdaptive DE.

## PSO. Algoritmos de nubes de partículas

La PSO es una metaheurística poblacional inspirada en el comportamiento social del vuelo de bandadas de aves y el movimiento de bancos de peces.

La población se compone de varias partículas (nubes/particle swan) que se mueven por el espacio de búsqueda durante la ejecución del algoritmo. Este movimiento de cada partícula p depende de:

* Su mejor posición desde que comenzó el algoritmo
* La mejor posición de las partículas de su entorno o de toda la nube desde que comenzó el algoritmo

En cada iteración se cambia aleatoriamente la velocidad de p para acercarla a las posiciones mejores del entorno y mejores globales.

Aplicación típica:

* Optimización continua (optimización de parámetros reales, numérica)

Características atribuidas:

* Asume un intercambio de información entre los agentes de búsqueda
* Idea Básica: Guardar información del mejor propio y global
* Implementación sencilla y con pocos parámetros
* Convergencia rápida a buenas soluciones.

**Funcionamiento Básico:**

* PSO simula el comportamiento de las bandadas de aves. Cada solución es un ave en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere
* Como las “aves” no saben dónde se encuentra su objetivo en un área determinada, la estrategia más eficaz es seguir a la que se encuentra más cerca de él. Es decir las partículas se mueven por el espacio de búsqueda y comunican y guardan la mejor solución (más cercana al objetivo) que han encontrado
* Cada una de las partículas tiene un fitness, una posición y un vector de velocidad que dirige el “vuelo”, el vuelo se guía por las partículas óptimas en el momento actual.

**Inicialización de la nube de partículas**

* La nube se inicia generando las posiciones y las velocidades iniciales de las partículas
* Las posiciones se pueden generar aleatoriamente en el espacio de búsqueda, de forma regular o combinando los dos métodos
* Las velocidades se generan aleatoriamente, en intervalo –Vmax,Vmax, si se ponen a 0 no se obtienen buenos resultados

**Movimiento de partículas**

* Las partículas se mueven añadiendo el vector de velocidad a su posición para obtener una nueva posición, el vector de velocidad está influenciado por la información de la partícula y la del entorno (tipos de algoritmo, cognitivo /social)
* Esta nueva posición se evalúa y si es mejor que la que tenía la partícula se actualiza

**Topologías de la nube de partículas**

* Las topologías definen el entorno de cada partícula individual, pueden ser de dos tipos, aunque los más usados son los sociales(en anillo):
  + Geográficos, se calcula la distancia de la partícula actual al resto y se toman las más cercanas
  + Sociales: Se define una lista de vecinas inicial independiente de la posición
* Hay que definir a parte el tamaño del entorno que entre 3 y 5 suele funcionar bien